Numerical Filter Stability, Fokker-Planck Equations and Infinite Dimensional Optimization with Deep Learning

A Thesis

Submitted to the
Tata Institute of Fundamental Research, Mumbai
for the degree of Doctor of Philosophy
in Subject Board of Mathematics

by

Pinak Mandal

International Centre for Theoretical Sciences, Bengaluru
Tata Institute of Fundamental Research

January, 2024
[Final Version Submitted in April, 2024]

I would like to dedicate this thesis to my loving family and all the kind people I've met so far.

Declaration

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgment of collaborative research and discussions.

The work was done under the guidance of Professor Amit Apte at the Tata Institute of Fundamental Research, Mumbai.

Pinak Mandal

[Candidate: Pinak Mandal]

In my capacity as the formal supervisor of record of the candidate's thesis, I certify that the above statements are true to the best of my knowledge.

[Research Supervisor: Vishal Vasan]

Date: 30 April 2024

Acknowledgements

This work was supported by the Department of Atomic Energy, Government of India, under project no. RTI4001, the Infosys-TIFR Leading Edge travel grant (2023) and the International Research Connections Fund (2022). Special thanks to Zhisong Qu, Matthew Hole and Robert Dewar for useful discussions on the topics of the last chapter. Also special thanks to Shashank Kumar Roy for working with me on topics of the first chapter.

I would like to thank all the amazing people I met at ICTS and NCBS, for being friends and teachers. In their company I experienced things I never thought I would and made memories for a lifetime. They are numerous and I offer them my sincerest apologies for not mentioning them by name. I would like to thank Vishal Vasan and Amit Apte for having faith in me and creating an environment where for better or worse I started seeing them as friends rather than professors. I would also like to thank Sreekar Vadlamani for always being encouraging.

List of figures

List of tables

List of symbols

$\hat{\pi}_n(\mu)$ Approximate filtering distribution at assimilation time n starting with the initial distribution μ

$\hat{\pi}_n^{\mathrm{EnKF}}(\mu), \hat{\pi}_n^{\mathrm{E}}(\mu)$ Approximate filtering distribution computed with particle filter at assimilation time n starting with the initial distribution μ

$\hat{\pi}_n^{\mathrm{PF}}(\mu), \hat{\pi}_n^{\mathrm{P}}(\mu)$ Approximate filtering distribution computed with particle filter at assimilation time n starting with the initial distribution μ

$\hat{\pi}_n^{X,*}(\mu)$ Limit of $\hat{\pi}_n^{X,N}(\mu)$ as $N \uparrow \infty$ assuming it exists

$\hat{\pi}_n^{X,N}(\mu)$ Approximate filtering distribution computed with algorithm X with N particles at assimilation time n starting with the initial distribution μ

$\mathcal{P}(\mathbb{R}^d)$ Set of probability measures on $\mathbb{R}^d$

$\mathcal{U}(\mathcal{D})$ Uniform distribution on $\mathcal{D}$

$\pi_n(\mu)$ True filtering distribution at assimilation time n starting with the initial distribution μ

$\xi(T, \mathcal{D}, \Omega)$ Average probability that a trajectory of the h-SDE stays inside Ω till time T after starting in $\mathcal{D}$

C^k Space of real-valued functions with k continuous derivatives

C_0^k Space of real-valued functions vanishing at infinity with k continuous derivatives

C_b Space of real-valued bounded continuous functions

$D_\varepsilon(\mu, \nu)$ Square root of Sinkhorn divergence between probability measures μ, ν with regularization parameter ε

$e_n(\mu)$ Scaled l_2 error or RMSE of a filtering algorithm at assimilation time n starting with the initial distribution μ with the algorithm identifier suppressed

L^2 Space of square integrable functions

$p_\infty + \text{FK}$ Algorithm for computing time dependent solutions to Fokker-Planck equations using the unnormalized steady state and the Feynman-Kac formula

$S_\varepsilon(\mu, \nu)$ Sinkhorn divergence between probability measures μ, ν with regularization parameter ε

$s_n(\mu)$ Square root of the trace of the sample covariance or uncertainty of a filtering algorithm at assimilation time n starting with the initial distribution μ with the algorithm identifier suppressed

$W^{k,p}$ Sobolev space of functions with k p-integrable weak derivatives

$W^{k,p}_{\text{loc}}$ Space of functions whose restrictions to compact sets are Sobolev

W_p The p-th Wasserstein metric

$\text{AL}^{\infty}_{\infty}$ Augmented Lagrangian algorithm for infinite dimensional problems when the output of the constraint function is a function itself

$\text{AL}^{\infty}_{\text{F}}$ Augmented Lagrangian algorithm for infinite dimensional problems when the output of the constraint function is finite dimensional

P^{∞} Penalty algorithm for infinite dimensional problems

Table of contents

Abstract

We tackle three different problems. Each problem is connected to the previous one in one or more ways, either directly or through the method of solution. Moreover, all three topics are comfortably enveloped by the overarching theme of numerical optimization. Our first problem is to evaluate stability of nonlinear filtering algorithms when the underlying dynamics is deterministic. We demonstrate that popular filtering algorithms are exponentially stable and discover a relationship between filter RMSE and filter stability. Our second problem is to devise an algorithm to solve Fokker-Planck equations in high dimensions. We use a deep learning algorithm to compute the non-trivial zeros of the Fokker-Planck operator. We show that combining these zeroes with an appropriate Feynman-Kac formula gives us the solution to the time-dependent Fokker-Planck equation. Lastly, we discuss two deep learning algorithms for solving constrained optimization problems in infinite dimensional Hilbert spaces. We test these algorithms on some toy problems inspired by calculus of variations and physics.

Introduction

One of the challenging problems in earth sciences is to incorporate the vast quantities of data that are constantly being collected world-wide into dynamical models for these systems, and is called the problem of data assimilation (DA). DA is a crucial ingredient for making meaningful real time predictions such as weather forecasts, hurricane tracking, and possibly even climate predictions. [7, 27] The Bayesian formulation of DA naturally leads to the problem of nonlinear filtering, which studies the conditional distribution, called the *filter* or the *posterior* distribution, of the state at any time conditioned on observations up to that time. [3, 106, 148]

A natural question is about the stability of the filter with respect to the initial condition, which is the probability distribution of the state at the initial time. This question has been studied extensively, e.g., [15, 33], but mostly in the context of stochastic systems. In many applications in the earth sciences, the models used are deterministic and only a few results about filter stability are known, see [147, 146] and the references therein. The main focus of the first part of this thesis is to illustrate numerically the stability of two commonly used filtering algorithms, namely the particle filter and the ensemble Kalman filter, in the case of deterministic dynamical systems. But our methods are in no way limited to deterministic systems alone and are readily applicable to stochastic systems as well. We also study the exponential nature of filter stability along with its dependence on two main parameters of the nonlinear filtering problem, the observation gap and the observation noise.

In order to investigate filter stability we need to compute distances between probability measures. Due to its nice geometric properties we use the Wasserstein metric as our choice of distance between probability measures. Computing the Wasserstein metric requires one to solve an optimization problem and this unveils

the overarching theme of this thesis, namely numerical optimization. Even though throughout the majority of this thesis we are interested in topics where optimization is not the end goal, it serves as an important tool in solving the problems at hand. Moreover, in the last chapter numerical optimization becomes the main subject matter.

In the Bayesian formulation, nonlinear filtering is usually done in two recursive steps. In the first step called "prediction", we evolve our knowledge of the state according to the underlying dynamical system. In the second step called "correction" we correct our knowledge with an Bayesian update using the current observation. In case the underlying dynamical system is stochastic, we can think of the prediction step as solving an appropriately formulated Fokker-Planck equation which indicates that algorithms for solving such equations can lead to new filtering algorithms. Other than that Fokker-Planck equations naturally arise as descriptions of evolution of probability densities everywhere [108], [93], [161] and are interesting in their own right. In 1996 Jordan et al showed that solutions to Fokker-Planck equations can be viewed as minimizers of a regularized version of the squared Wasserstein metric [87]. Therefore, these equations are intricately linked to the topics of the first chapter in more ways than one . The second and the third chapters of this thesis deal with Fokker-Planck equations. In particular, we are interested in ways to solve high dimensional Fokker-Planck equations. Since deep learning methods can be implemented in mesh-free ways they give us a viable paradigm for working with high dimensional problems. In the second chapter we develop a deep learning algorithm for learning zeros of high-dimensional Fokker-Planck operators. In the third chapter we use the Feynman-Kac formula with the stationary solutions of Fokker-Planck equations to solve time dependent Fokker-Planck equations.

The deep learning methods used in the second chapter to estimate zeros of Fokker-Planck operators can be extended to solve some constrained optimization problems which is the topic of the last chapter. Penalty and augmented Lagrangian are two popular methods for solving constrained optimization problems in finite dimensions [129]. And although their infinite dimensional analogues have been theoretically studied [81], [91], their practical implementations are few and far between. The goal of the last chapter of this thesis is to bridge this gap between theory and implementation. We devise deep learning versions of the penalty and augmented Lagrangian algorithms

for constrained optimization problems in infinite dimensional Hilbert spaces. We consider a few toy problems inspired by calculus of variations and physics to explore this topic.

Optimization techniques are the central to this thesis as problem solving tools. Even though they do not become the main topic of interest until chapter 4, they appear in various subproblems that help us tackle the original problems in the first two chapters. In chapter 1, in order to evaluate filter stability we need to compute distances in the space of probability measures. The Wasserstein distance between probability measures happens to be an ideal choice of distance for us which is defined in terms of an optimization problem. Thus our route to assessing filter stability passes through optimization as a subproblem. In chapter 2 our main objective is to compute the zeros of Fokker-Planck operators which have deep connections to data assimilation when the the underlying dynamical system is noisy. To do this we essentially recast the problem as an unconstrained infinite dimensional optimization problem. The method described in chapter 2 is also an integral part of the central algorithm in chapter 3. Finally, in chapter 4 we investigate the more general scenario when infinite dimensional optimization problems similar to the one in chapter 2 need to be solved in presence of constraints.

Outline of this thesis

Below we provide a brief outline of this thesis along with the source materials and necessary attributions of credit. The work appearing in the first three chapters was done under the supervision of Amit Apte.

Probing nonlinear filter stability

In this chapter we explore filter stability by directly assessing it with Sinkhorn divergence. The work in this chapter was done alongside Shashank Kumar Roy. The Kalman filter experiments appearing in this chapter were done by Shashank while the author was in charge of the particle filter experiments. The contents of this chapter are taken from the following papers.

- Mandal, P., Roy, S. K., and Apte, A. (2021). Stability of nonlinear filters - numerical explorations of particle and ensemble Kalman filters. In *2021 Seventh Indian Control Conference (ICC)*, pages 307–312. IEEE

- Mandal, P., Roy, S. K., and Apte, A. (2023). Probing robustness of nonlinear filter stability numerically using Sinkhorn divergence. *Physica D: Nonlinear Phenomena*, 451:133765

Learning zeros of Fokker-Planck operators

In this chapter we devise a deep learning algorithm to learn solutions to stationary Fokker-Planck equations. The contents of this chapter are taken from the following paper.

- Mandal, P. and Apte, A. (2023). Learning zeros of Fokker-Planck operators. *arXiv preprint arXiv:2306.07068*

Solving Fokker-Planck equations

In this chapter we solve time-dependent Fokker-Planck equations using a hybrid algorithm which uses the zeros of the Fokker-Planck operator and the Feynman-Kac formula. The contents of this chapter are taken from the following paper.

- Mandal, P. and Apte, A. (2024). Solving Fokker-Planck equations using the zeros of Fokker-Planck operators and the Feynman-Kac formula. *arXiv preprint arXiv:2401.01292*

Learning solutions to some toy constrained optimization problems

In this chapter we explore the penalty and augmented Lagrangian algorithms for infinite dimensional problems in a deep learning setting. The author is grateful to Zhisong Qu, Matthew Hole and Robert Dewar for helpful discussions on the topics of this chapter. The contents of this chapter are taken from the following paper.

- Mandal, P. (2024). Learning solutions to some toy constrained optimization problems in infinite dimensional Hilbert spaces. *arXiv preprint arXiv:2401.01306*

Chapter 1

Particle filters and ensemble Kalman filters are widely used in data assimilation but in the case of deterministic systems, which are quite commonly used in earth science applications, only a few theoretical results for their stability are available. Current numerical literature explores stability in terms of RMSE which, although practical, cannot represent the distance between probability measures, convergence of which is what defines filter stability. In this study, we explore the distance between filtering distributions starting from different initial distributions as a function of time using Wasserstein metric, thus directly assessing the stability of these filters. These experiments are conducted on the chaotic Lorenz-63 and Lorenz-96 models for various initial distributions for particle and ensemble Kalman filters. We show that even in cases when both these filters are stable, the filtering distributions given by each of them may be distinct. We explore the exponential nature and robustness of filter stability by varying two crucial parameters of the nonlinear filtering problem, namely the observation gap and the observation noise. We also establish numerically a relation between filter stability and filter convergence by showing that the Wasserstein distance between filters with two different initial conditions is proportional to the bias or the RMSE of the filter before stabilizing.

Chapter 1

Probing nonlinear filter stability

1.1 Introduction

The problem of predicting the state of a complex dynamical system is ubiquitous in many scientific and engineering fields. In the context of earth sciences, weather prediction and reanalysis of past climate [39, 105] are major examples of such state estimation problems, which have two main ingredients: (i) a dynamical model, usually deterministic, of the system and (ii) partial, usually sparse, and needless to say, noisy, observations. The process of combining these observations with the model to get an "optimal" state estimate is commonly called data assimilation - a term introduced in earth sciences [29, 57, 90].

The mathematical formulation of this problem in a Bayesian framework encapsulates the information from the model in terms of a prior distribution, and the observational likelihood is used to obtain a posterior distribution for the model state [3, 106]. For dynamical systems, this is precisely the problem of nonlinear filtering, where the posterior distribution itself changes with time and is conditioned on observations up to that time. This posterior is called the *filter* or the *filtering distribution* [152, 168, 7].

A crucial characteristic of the atmospheric and oceanic dynamics is their chaotic nature, manifested in the sensitivity to initial conditions. Thus a natural question is whether the filter is also sensitive to the choice to the initial distribution. In nonlinear filtering, this appears in the form of a question about the stability of the filter [34, 33, 42]. A filter is stable if two different initial distributions lead to the same filtering distribution asymptotically in time. This is a desirable property for a filter

since the choice of the initial distribution is arbitrary and we desire the filter to "forget" about this arbitrary choice. Thus filter stability is an extensively studied topic, but mainly in the context of stochastic dynamics with only limited results for deterministic dynamical systems [146, 132, 147, 31].

In practice, nonlinear filters need to be implemented numerically and we will focus on two of the most commonly used methods, namely, particle filters (PF) and ensemble Kalman filters (EnKF). The stability of particle filters has been extensively studied [35, 172] while very few theoretical results are known for the stability of EnKF [48], though some results related to filter divergence (which is quite distinct from an unstable filter) and accuracy for the EnKF are available [94, 107, 67]. We note that the assumptions used to prove stability of PF are not satisfied by a deterministic dynamical model and thus their stability in the context of data assimilation needs to be explored numerically. This is the main aim of the present chapter.

In order to assess filter stability directly using the definition (see definition (1.2.2)), we need to compute distances between probability distributions. This has been a challenging task, but recently proposed Sinkhorn algorithm provides an efficient method for this task [61, 62, 55, 5]. One of the novelties of this chapter is to demonstrate the use of the Sinkhorn algorithm in the context of data assimilation for directly studying stability. Even though we focus on deterministic dynamical systems here, our methods are readily applicable to stochastic systems as well.

Numerical studies of filter divergence, especially in the context of twin experiments where synthetic observations are generated using the model, have focused on assessing whether the filter remains bounded or whether the 'error' or 'bias' of the filter (commonly called RMSE, i.e. distance of the filter mean from the numerical trajectory which is used to generate the synthetic observations) remains bounded in time [94, 107]. But this does not provide a direct indication of stability as defined in (1.2.2). In this chapter, we demonstrate that there is a linear relationship between the filter error and the distance between two filters started with two different initial distributions, the latter giving a more appropriate measure of stability. This direct relation between filter stability and filter RMSE for particle and ensemble Kalman filters for deterministic dynamics is the other main contribution of this chapter.

The outline of this chapter is as follows. The next section introduces the mathematical setting for the problem of filter stability. This is followed by a description of the filtering algorithms (particle filter and ensemble Kalman filter), and the Sinkhorn algorithm for computing distances between probability distributions. After that we present our results in two parts. The first part looks at the numerical evidence for stability of the filtering algorithms considered here. In the second part we investigate the exponential nature and robustness of filter stability by varying two main parameters in the filtering problem, namely the observation gap and observation noise. We end this chapter with a summary of the main results along with some possible future directions of research.

1.2 Problem statement

1.2.1 The nonlinear filtering problem

In this chapter we work with a dynamical model given by a deterministic and chaotic ODE. Our model state is d-dimensional and the flow corresponding to the model is denoted by $\phi : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$. We observe the model every g units of time and call g the observation gap. So the model state x_k follows a discrete-time deterministic dynamical system $f_g \stackrel{\text{def}}{=} \phi(g, \cdot) : \mathbb{R}^d \to \mathbb{R}^d$. The observations $y_k \in \mathbb{R}^q$ is related to the model state by the observation operator (a linear projection throughout this chapter) $H : \mathbb{R}^d \to \mathbb{R}^q$ for $k = 0, 1, \ldots$, as follows:

$$x_{k+1} = f_g(x_k), \quad x_0 \sim \mu\,, \tag{1.1}$$

$$y_k = Hx_k + \eta_k\,, \tag{1.2}$$

where μ is the initial distribution of the model state x_0 at time 0, and $\eta_k \sim \mathcal{N}(0_q, \sigma^2 I_q)$ are *iid* Gaussian errors in the observation, and are assumed independent of μ. Given observations $y_0, y_1, \cdots, y_n$, the goal of filtering is to estimate the conditional distribution of the model state at time n conditioned on observations up to that time: $x_n | y_{0:n} \sim \pi_n(\mu)$, where the dependence on the distribution μ of the initial condition x_0 is made explicit since our focus will be on filter stability.

1.2.2 Filter stability

In practice we often do not know the initial distribution μ. In such a case, when a different initial condition ν is chosen, one obtains a different filter, denoted by $\pi_n(\nu)$, by using the same set of observations and using the same algorithm. A measure of robustness of a filtering algorithm is how well it is able to "forget" the initial distribution, which motivates the following definitions.

There are two different kinds of randomness that one needs to deal with in the setup above, the initial condition and the observation noise. Suppose $x_0 : \Omega \to \mathbb{R}^d$ is our random initial condition. Consider $x_0(\omega)$, a realization of this initial condition. Now that we have fixed a realization of x_0, $\pi_n(\nu)$ and $\pi_n(\mu)$ become random measures whose randomness is determined only by the observation noise. For $x_0(\omega)$ we can compute the following expectation with respect to the observation noise

$$\mathbb{E}\left|\int_{\mathbb{R}^d} h(x)\,\pi_n(\mu, dx) - \int_{\mathbb{R}^d} h(x)\,\pi_n(\nu, dx)\right| \tag{1.3}$$

for a bounded, continuous function h. If this expectation approaches 0 as $n \to \infty$ for any such h we can say that the filter is "pointwise" stable for the initial realization $x_0(\omega)$. And if the filter is "pointwise" stable for almost all realizations of x_0, we call the filter stable.

[146] explores the "true" filter stability for deterministic dynamics. Below we adapt the definition for numerical filter stability. Below $\hat{\pi}$ denotes the numerical approximation of the true filter π.

Definition 1.2.1 (Stability-RA [146]) *A numerical filter is stable if for any measure ν with $\mu \ll \nu$ we have,*

$$\lim_{n\to\infty} \mathbb{E}\left|\int_{\mathbb{R}^d} h(x)\,\hat{\pi}_n(\mu, dx) - \int_{\mathbb{R}^d} h(x)\,\hat{\pi}_n(\nu, dx)\right| = 0\,, \tag{1.4}$$

for any bounded and continuous h, μ-almost everywhere in the sense described above.

Note that the expectation above is taken with respect to the observation noise only. Although (1.4) captures the notion of filter stability quite well, from a computational perspective we can improve on it in the following aspects.

- Computing the expectation for every possible bounded and continuous function is infeasible.
- In real world applications we might not have access to μ and therefore an expectation independent of μ is preferable.

In order to overcome above difficulties and to assess filter stability numerically, we devise the following definition which can be proven to be a stronger version of definition 1.2.1 in an appropriate sense (see Theorem 1.6.7 in appendix).

Definition 1.2.2 (Stability-MRA [122]) *A numerical filter is said to be stable if for any two distributions ν_1, ν_2, the following holds,*

$$\lim_{n\to\infty} \mathbb{E}[D(\hat{\pi}_n(\nu_1), \hat{\pi}_n(\nu_2))] = 0\,, \tag{1.5}$$

μ-almost everywhere, where D is a distance on $\mathcal{P}(\mathbb{R}^d)$, the space of probability measures on $\mathbb{R}^d$.

Note that even with the modifications, definition 1.2.2 remains hard to compute in the following aspects.

- Computing the limit for every possible pair ν_1, ν_2 is infeasible.
- Computing the limit for every possible initial realization $x_0(\omega)$ is infeasible.

The last two difficulties also arise in definition 1.2.1 and are unavoidable in some sense in a complete definition of filter stability. But even with these difficulties we can explore numerical filter stability in a meaningful albeit slightly limited way. Although we demonstrate results for a single realization $x_0(\omega)$ here, this realization was generated randomly and different initial realizations yield qualitatively similar results which are consistent with the stability definition 1.2.2 and hence are not included in the chapter to avoid repetition.

The main aim of this chapter is to study the stability of two popular filtering algorithms, namely the particle filter (PF) and the ensemble Kalman filter (EnKF) by studying the limit in (1.5), where we choose the Wasserstein metric W_2 as our distance D on $\mathcal{P}(\mathbb{R}^d)$. Thus we study $\mathbb{E}[D(\hat{\pi}_n(\nu_1), \hat{\pi}_n(\nu_2))]$ as a function of time n for PF and EnKF algorithms. We also study the rate of convergence of the expectation in (1.5)

and how it varies with respect to the time between the observations denoted by g and the observational uncertainty or the error variance σ^2. In the following discussion we sometimes abuse the notation and use π to mean $\hat{\pi}^{\text{PF}}$ or $\hat{\pi}^{\text{EnKF}}$ with clear context. We now describe these numerical filtering algorithms, followed in section 1.2.6 by a description of the Sinkhorn algorithm for computing distances between probability distributions.

1.2.3 Ensemble Kalman Filters

Kalman filters provide the closed form solutions to the Bayesian filtering equations in the scenario when the dynamic and measurement models are linear Gaussian or if the state equation (1.1) looks like

$$x_{k+1} = A_k x_k + \alpha_k \,, \tag{1.6}$$

where $\alpha_k \sim \mathcal{N}(\mathbf{0}, Q_k)$. The filtering distribution in this special case turns out to be Gaussian. The mean and covariance of this distribution is computed recursively in two steps, a prediction step where the effect of the hidden dynamics is captured and $p(x_k|y_{1:k-1})$ is computed and an update step where the observation y_k is taken into account to give the filtering distribution $p(x_k|y_{1:k})$ using Baye's rule and well-known properties of the multivariate Gaussian distribution.

Ensemble Kalman filters can be thought of as an approximation of the original Kalman filter where the filtering distribution is represented by a collection of particles, as is the norm in Monte Carlo-based methods. The ensemble representation is akin to dimension reduction which leads to computational feasibility for systems with large state space dimension d [53]. Localization, which is the process of weeding out long range spurious correlations, has made EnKF more applicable as well as wildly popular in high-dimensional data assimilation problems for spatially extended systems. For a discussion about localization see [27]. We use Gaspari-Cohn function as our choice of localization function with radius set to 2. The details of the exact implementation that we use can be found in algorithm 1.

Algorithm 1 EnKF with covariance localization in state-space. $\circ$ denotes Hadamard product.

Initialize N particles $\{x_0^i\}_{i=1}^N$ according to the initial distribution and set $x_0^{i,a} = x_0^i$
Set ρ as the Gaspari-Cohn localization matrix [27].
for $k = 1, \cdots, n$ **do**
  for $i = 1, \cdots, N$ **do**
    $x_k^{i,f} \leftarrow f_g(x_{k-1}^{i,a})$
  end
  $m_k^f \leftarrow \frac{1}{N} \sum_i x_k^{i,f}$
  $P_k^f \leftarrow \rho \circ \frac{\sum_i \left(x_k^{i,f} - m_k^f\right)\left(x_k^{i,f} - m_k^f\right)^\top}{N-1}$
  $K \leftarrow P_k^f H^T \left[HP_k^f H^T + R_k\right]^{-1}$
  for $i = 1, \cdots, N$ **do**
    Sample $\eta_k^i \sim \mathcal{N}(0_q, \sigma^2 I_q)$
    $y_k^i \leftarrow y_k + \eta_k^i$
    $x_k^{i,a} \leftarrow x_k^{i,f} + K\left[y_k^i - Hx_k^{i,f}\right]$
  end
  $\hat{\pi}_k \leftarrow \frac{1}{N} \sum_{i=1}^N \delta_{x_k^{i,a}}$
end

1.2.4 Particle Filters

Particle-filters are also Monte Carlo-based filters that recursively compute importance sampling approximations of the filtering distribution $p(x_k|y_{1:k})$. PFs also follow the Bayesian paradigm of two-step recursion with prediction and update steps. The filtering distribution is represented as a collection of weighted particles. In the prediction step the particles are evolved in time according to (1.1) which gives us the prior for the next Bayesian update step where the weights are adjusted appropriately to account for the observation. For an excellent overview of the PF algorithm see [50]. PFs do not rely on linearity or Gaussianity of dynamic of observation models which make them powerful but unless the number of particles scale exponentially with d, PFs experience weight degeneracy and provide poor estimates [11]. In order to combat weight degeneracy, a resampling step is performed after the Bayesian update where particles with negligible weights are replaced with particles with higher weights. Many variants of the standard or the bootstrap PF have been proposed and the interested reader can see [54] for a discussion. However, applying PFs on problems with significantly high dimensions still remains a challenge.

Algorithm 2 BPF with offspring-based resampling

Initialize N particles $\{x_0^i\}_{i=1}^N$ according to the initial distribution with equal weights $\left\{w_0^i = \frac{1}{N}\right\}_{i=1}^N$. Set $\tilde{\sigma}$. Below $S[i]$ denotes i-th element of S.

for $k = 0, \cdots, n$ **do**
  if $k > 0$ **then**
    for $i = 1, \cdots, N$ **do**
      $x_k^i \leftarrow f_g(S[i])$
    end
  end
  Sample $u \sim \mathcal{U}\left(0, \frac{1}{N}\right)$
  for $i = 1, \cdots, N$ **do**
    $w_k^i \leftarrow p(y_k | x_k^i)$
    $U_i \leftarrow u + \frac{i-1}{N}$
  end
  $W \leftarrow \sum_{i=1}^N w_k^i$
  for $i = 1, \cdots, N$ **do**
    if $|\{U_j : \sum_{l=1}^{i-1} w_k^l \leq W U_j \leq \sum_{l=1}^{i} w_k^l\}| > 0$ **then**
      tag x_k^i as significant
    end
  end
  Set $S \leftarrow \{x_k^{i_1}, x_k^{i_2}, \cdots, x_k^{i_m}\}$ as the set of significant particles and compute $N_j \propto w_k^{i_j}$: $\sum_{j=1}^m N_j = N$.
  for $j = 1, \cdots, m$ **do**
    $S \leftarrow S \cup \{N_j - 1$ samples from $\mathcal{N}(x_k^{i_j}, \tilde{\sigma}^2 I_d)\}$
  end
  $\hat{\pi}_k \leftarrow \frac{1}{N} \sum_{i=1}^N \delta_{S[i]}$
end

We use the bootstrap particle filter for our experiments with a stochastic resampling step where we place a Gaussian distribution with a pre-determined, small covariance $\tilde{\sigma}^2$ around the best-performing particles and sample new particles according to the weights. The details of the exact algorithm can be found in algorithm 2. In algorithm 2, we use the convention that $\sum_{l=1}^0 w_k^l = 0$. It should be noted that larger values of $\tilde{\sigma}$ are needed to prevent filter collapse when working with fewer number of particles. In our experiments for Lorenz 63, $\tilde{\sigma}^2 = 0.1$ and for Lorenz 96, $\tilde{\sigma}^2 = 0.5$.

1.2.5 Choice of distance D

Although the stability definitions are independent of any kind of specific distance and can be computed with other choices of D we choose D to be the 2nd Wasserstein distance W_2. We justify our choice with the reasons below.

- An efficient algorithm exists for approximating W_p which is described below.
- W_p metrizes the convergence in law unlike some other distances and distance-substitutes e.g. the total variation distance and the KL-divergence, see Introduction and Appendix B.5 in [55] for a discussion. This makes W_p a more *intuitive* distance to work with, see example 1 in section 2 of [5] for an illustration.
- Moreover, W_p does not require the notion of absolute continuity unlike KL-divergence or Hellinger distance which is useful for comparing empirical distributions.

For a comparison of these distances the interested reader may see [5] where example 1 (learning parallel lines) depicts how the output of W_p can often be intuitive because Wasserstein distances lift the standard metrics on $\mathbb{R}^d$ to the probability space $\mathcal{P}(\mathbb{R}^d)$ unlike KL-divergence or total variation. Lastly, we use $p = 2$ for no reason other than the familiarity of the 2-norm on Euclidean spaces.

1.2.6 Sinkhorn divergence

The p-th Wasserstein distance (W_p) between probability measures with p-th finite moment on metric spaces have many desirable geometric features which stems form the fact that its definition extends the distance function on the metric space to a distance on the space of probability measures on the metric space. For a discussion see [55, 5]. W_1 or the earth mover's distance has been used in various problems e.g. comparing colour histograms, solving resource allocation problems etc. When applied to two sampling distributions with both having sample size k, computing W_1 is equivalent to solving a constrained linear programming problem in $n = k^2$ variables. Since LPPs take $O(n^3)$ time to solve a problem with n variables, computing W_1 takes $O(k^6)$ time which is prohibitively expensive.

In recent years it has been noted that by regularizing the optimization problem that defines the Wasserstein distance, one can attempt to solve the dual to the regularized problem which is akin to solving a convex optimization problem. For a comprehensive discussion see [61]. The dual problem can be solved using a variant of the Sinkhorn-Knopp algorithm for finding a doubly-stochastic matrix given a square matrix with positive entries. The solution to the regularized problem is known as the Sinkhorn divergence since it fails to satisfy the triangle inequality and is not an exact distance on the space of probability measures.

Here we focus on the case $p = 2$. For two probability measures μ and ν on $\mathbb{R}^d$ with finite first and second moments, the Sinkhorn divergence S_ε is defined as follows [55].

$$\mathrm{OT}_\varepsilon(\mu,\nu) \stackrel{\text{def}}{=} \min_{\pi\in\mathrm{S}} \left[\int \|x-y\|_2^2 \, d\pi(x,y) + \varepsilon \mathrm{KL}(\pi|\mu\otimes\nu) \right], \tag{1.7}$$

$$\mathrm{S}_\varepsilon(\mu,\nu) \stackrel{\text{def}}{=} \mathrm{OT}_\varepsilon(\mu,\nu) - \frac{1}{2}\mathrm{OT}_\varepsilon(\mu,\mu) - \frac{1}{2}\mathrm{OT}_\varepsilon(\nu,\nu), \tag{1.8}$$

where the minimisation is over the set S of distributions π with the first and second marginals being μ and ν respectively and KL is the Kullback–Leibler divergence. Moreover, it turns out [55] that

$$\lim_{\varepsilon\to 0} \sqrt{S_\varepsilon(\mu,\nu)} = W_2(\mu,\nu), \tag{1.9}$$

and therefore for small enough ε we obtain a good approximation of W_2. We use the following notation for this approximation: $D_\varepsilon = \sqrt{S_\varepsilon}$.

In our experiments we compute $S_\varepsilon(\mu,\nu)$ for sampling distributions $\mu = \sum_{i=1}^N \mu_i \delta_{x_i}$ and $\nu = \sum_{j=1}^M \nu_j \delta_{y_j}$ with $\varepsilon = 0.01$ where $\{x_i\}_{i=1}^N$ and $\{y_j\}_{j=1}^M$ are points in $\mathbb{R}^d$. A detailed justification of this choice of ε is in appendix 1.6.3. The Sinkhorn divergence algorithm being a fixed point iteration is extremely fast. The exact procedure is given in algorithm 3. The authors of [55] show that the algorithm is parallelizable with respect to sample-size. The dimension dependence of the algorithm is only explicitly apparent while calculating the distance matrix and consequently the algorithm itself scales only linearly in d. But to accurately represent the underlying distribution with increasing dimension one would require to compute S_ε with increasing sample size.

Algorithm 3 Computation of S_ε

Input: $\{\mu_i\}_{i=1}^N, \{x_i\}_{i=1}^N, \{\nu_j\}_{j=1}^M, \{y_j\}_{j=1}^M$

Output: $S_\varepsilon\left(\sum_{i=1}^N \mu_i \delta_{x_i}, \sum_{j=1}^M \nu_j \delta_{y_j}\right)$

Note the definition, $\mathrm{LSE}_{k=1}^L V_k \stackrel{\text{def}}{=} \log \sum_{k=1}^L \exp(V_k)$.
Initialize $a_i \leftarrow 0 \; \forall \; i = 1, \cdots, N$ and $b_j \leftarrow 0, \; \forall \; j = 1, \cdots, M$.
iteration $\leftarrow 0$
while min{L_1 *relative errors in a and b*} $> 0.1\%$ **do**
  for $i = 1, \cdots, N$ **do**
    $a_i \leftarrow -\varepsilon \mathrm{LSE}_{k=1}^M \left(\log \nu_k + \frac{1}{\varepsilon} b_k - \frac{1}{\varepsilon}\|x_i - y_k\|_2^2\right)$
  end
  for $j = 1, \cdots, M$ **do**
    $b_j \leftarrow -\varepsilon \mathrm{LSE}_{k=1}^N \left(\log \mu_k + \frac{1}{\varepsilon} a_k - \frac{1}{\varepsilon}\|x_k - y_j\|_2^2\right)$
  end
  iteration $\leftarrow$ iteration + 1
end
$\mathrm{OT}_{\mu,\nu} \leftarrow \sum_{i=1}^N \mu_i a_i + \sum_{j=1}^M \nu_j b_j$
Initialize $a_i \leftarrow 0 \; \forall \; i = 1, \cdots, N$ and $b_j \leftarrow 0, \; \forall \; j = 1, \cdots, M$.
while L_1 *relative error in* $a > 0.1\%$ **do**
  for $i = 1, \cdots, N$ **do**
    $a_i \leftarrow \frac{1}{2}\left[a_i - \varepsilon \mathrm{LSE}_{k=1}^N \left(\log \mu_k + \frac{1}{\varepsilon} a_k - \frac{1}{\varepsilon}\|x_i - x_k\|_2^2\right)\right]$
  end
end
while L_1 *relative error in* $b > 0.1\%$ **do**
  for $j = 1, \cdots, M$ **do**
    $b_j \leftarrow \frac{1}{2}\left[b_j - \varepsilon \mathrm{LSE}_{k=1}^M \left(\log \nu_k + \frac{1}{\varepsilon} b_k - \frac{1}{\varepsilon}\|y_j - y_k\|_2^2\right)\right]$
  end
end
$S_\varepsilon \leftarrow \mathrm{OT}_{\mu,\nu} - \sum_{i=1}^N \mu_i a_i - \sum_{j=1}^M \nu_j b_j$

For a detailed discussion of sample complexity of the Sinkhorn divergence see chapter 3 of [61].

1.3 Numerical Evidence for Filter Stability

In this section we directly calculate the distances between filtering distributions to assess the stability of particle filter and ensemble Kalman filter. We describe the experimental setup and the results and also explore the *zero of the Sinkhorn algorithm* since it is a fundamental part of the numerical evidence for filter stability presented here.

1.3.1 Experimental Setup

Models

Proposed first by Edward N. Lorenz, the Lorenz equations are sets of autonomous equations said to be mimicking the circulation of the earth's atmosphere in an over-simplified manner. Although simple, they have had significant impact on the development of the dynamical systems theory, especially because of their chaotic nature in arbitrary dimensions. Since an important application of data assimilation is numerical weather prediction, the Lorenz systems are a natural first choice for experiments. Since their inception they have been extensively used in data assimilation literature. We use two chaotic models in this section: (i) Lorenz-63 [72, Chapter 14] with parameters $\rho = 28, \alpha = 10, \beta = \frac{8}{3}$ and (ii) 10 and 40-dimensional Lorenz-96 [114, 167] with forcing constant $F = 10$ and $F = 8$ respectively. These ODE systems are defined in (1.10) and (1.11) respectively. Note that, in (1.10) and (1.11) the subscripts denote coordinates and the state x defined in (1.1) is a discretized version of these ODEs depending on the observation gap.

$$\begin{aligned} \frac{dX_1}{dt} &= \alpha(X_2 - X_1) \\ \frac{dX_2}{dt} &= X_1(\rho - X_3) - X_2 \\ \frac{dX_3}{dt} &= X_1 X_2 - \beta X_3 \end{aligned} \tag{1.10}$$

$$\frac{dX_j}{dt} = (X_{j+1} - X_{j-2})X_{j-1} - X_j + F, \quad j = 1, 2, \cdots, d \tag{1.11}$$

We observe the system every 0.1 units of time which fixes the evolution function f_g. We observe alternate coordinates starting from the first coordinate, so

$$y_{k,j} = x_{k,2j-1} + \eta_{k,j} \tag{1.12}$$

for $j = 1, 2, \cdots, q = \left\lceil \frac{d}{2} \right\rceil$ and $\eta_{k,j} \sim \mathcal{N}(0, \sigma^2)$. Throughout this section, we choose $\sigma^2 = 0.1$ or $\sigma^2 = 1.0$.

1.3.2 Data generation

Lorenz systems are known to have attracting sets. In this chapter, we focus on the special case when the filtering distributions are expected to be supported on the attractor. Not only does it mimic real world scenarios, it also lets us make use of the theory optimal transport distances, outlined in [55], when the probability measures are supported on a compact domain, since the Lorenz attractors are bounded sets. So we begin by finding a point on the attractor by randomly generating an initial point and evolving it according to f_g for 10^5 iterations. Starting from this point x_0^{true} on the attractor, we generate a true trajectory according to (1.1) and then generate 10 different observation realizations for the same trajectory according to (1.14) in order to compute the expectation over observational noise, as in (1.5). For a justification of why 10 observation realizations suffice for our study, see appendix 1.6.4.

Initial distributions

We use three initial conditions:

$$\begin{aligned}\mu_1 &= \mathcal{N}(x_0^{\text{true}}, 0.1 \times I_d)\,,\\ \mu_2 &= \mathcal{N}(x_0^{\text{true}} + 2 \times 1_d, 0.5 \times I_d)\,,\\ \mu_3 &= \mathcal{N}(x_0^{\text{true}} + 4 \times 1_d, I_d)\,,\end{aligned} \tag{1.13}$$

where 1_d is a d-dimensional vector with all entries 1. With this notation x_0^{true} corresponds to $x_0(\omega)$ in subsection 1.2.2. Note that different realizations of x_0 produce similar results as shown here.

1.3.3 Zero of the Sinkhorn algorithm

In sections 1.3.4, 1.3.5 we discuss the stability of PF and EnKF by calculating $\mathbb{E}[D_\varepsilon(\hat{\pi}_n(\mu_i), \hat{\pi}_n(\mu_j))]$ $(i \neq j)$ as a function of time n for initial conditions μ_i from (1.13), with the expectation taken by averaging over 10 observation realizations. For clarity, this quantity is shown at every 4-th assimilation step in figures 1.2–1.5. In order to understand the convergence to 0 of this expectation [see (1.5)], we first discuss how close to zero D_ε can approach numerically. In figure 1.1 we see the average $D_\varepsilon(\alpha_m^d, \beta_m^d)$

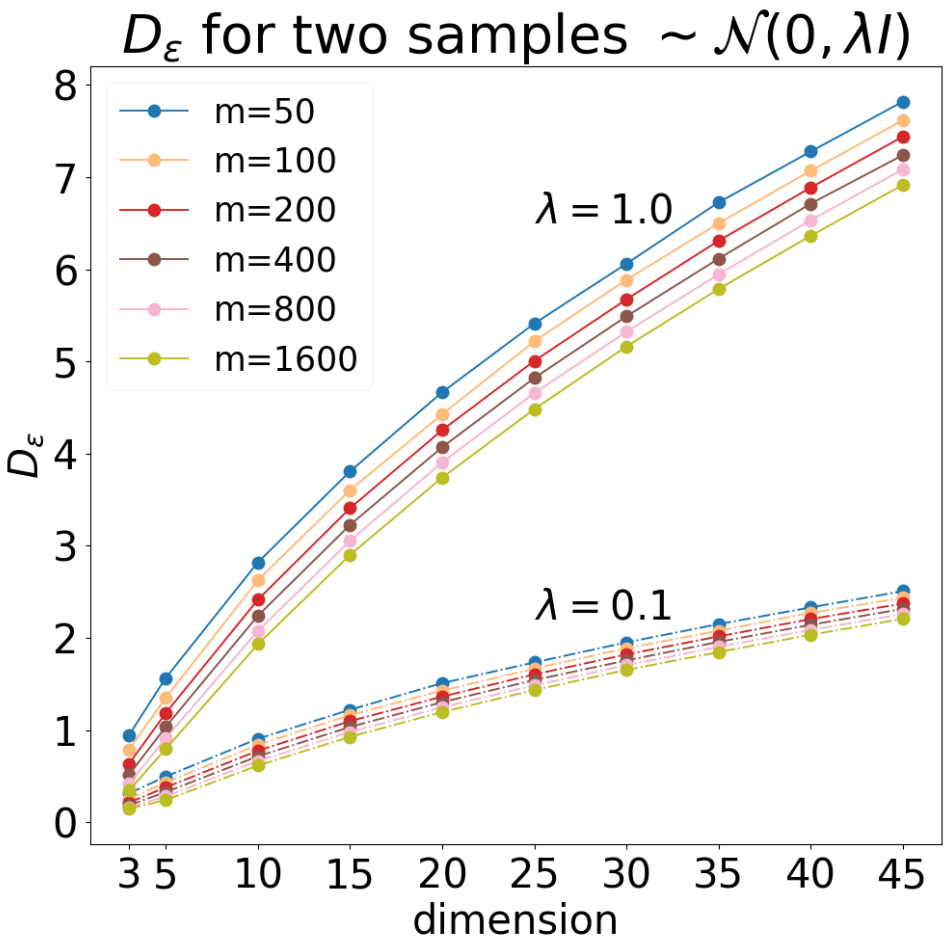

Fig. 1.1 Average $D_\varepsilon(\alpha_m^d, \beta_m^d)$ (over 20 realizations) where α_m^d, β_m^d are two different sampling distributions with the same sample size m for the same underlying d-dimensional Gaussian $\mathcal{N}(0_d, \lambda I_d)$

where $\alpha_m^d = \frac{1}{m}\sum_{i=1}^m \delta_{x_i^{m,d}}$ and $\beta_m^d = \frac{1}{m}\sum_{i=1}^m \delta_{y_i^{m,d}}$, with $\{x_i^{m,d}\}$ and $\{y_i^{m,d}\}$ both samples from the same underlying d-dimensional Gaussian distribution $\mathcal{N}_d^\lambda := \mathcal{N}(0_d, \lambda I_d)$.

For "small" λ, we can expect D_ε to behave in a similar fashion as if $\mathcal{N}_d^\lambda$ were supported on a compact set. With that in mind, we relate the numerical results shown in figure 1.1 to the results 1.6.1–1.6.3 in the appendix 1.6.1 by noting the following key points:

Drop with increase in sample size

Theorem 1.6.3 explains the monotone drop in average D_ε for a fixed dimension while increasing the sample size.

Rise with increase in dimension

As the dimension increases, larger sample sizes are required to accurately estimate $\mathcal{N}_d^\lambda$. Consequently, $D_\varepsilon(\alpha_m^d, \beta_m^d)$ grows with d for fixed m since α_m^d, β_m^d become poorer estimators of $\mathcal{N}_d^\lambda$ as d increases.

Drop with decrease in covariance

Decreasing the covariance λ has the opposite effect since, for fixed dimension d and sample size m, smaller covariance leads to a better estimation of the underlying distribution, i.e., α_m^d, β_m^d become better estimators of $\mathcal{N}_d^\lambda$ as λ decreases.

Support of our distributions

Since the true trajectories for both systems (L63, L96) lie on bounded attractors, we can assume that true filtering distributions are supported on a compact set. Consequently, in the filtering experiments shown later, the zero of the Sinkhorn algorithm shows qualitatively similar behavior (e.g., in figure 1.2) with respect to dimension as seen in figure 1.1.

1.3.4 Stability of Particle Filter

Here we use the notation $\pi_n^{P,N}$ for $\hat{\pi}_n$ obtained by alogithm 2 with N particles (omitting N for brevity when value of N is clear from context). Figure 1.2 shows $\mathbb{E}[D_\varepsilon(\hat{\pi}_n(\mu_i), \hat{\pi}_n(\mu_j))]$, $i \neq j$ as a function of n. We note some important conclusions.

BPF quickly forgets the initial distribution

From the insets in figure 1.2 we can see that for every pair (μ_i, μ_j) of initial distributions, $\mathbb{E}\left[D_\varepsilon(\pi_n^P(\mu_i), \pi_n^P(\mu_j))\right]$ stabilizes in the first few assimilation steps. In fact, this behavior is consistent with exponential stability of particle filters [34].

Dependence on the number of particles

$\mathbb{E}\left[D_\varepsilon(\pi_n^{P,N}(\mu_i), \pi_n^{P,N}(\mu_j))\right]$ for a fixed n decreases monotonically with increasing N for both L63 and L96 and for both observation covariances for all pairs $i \neq j$.

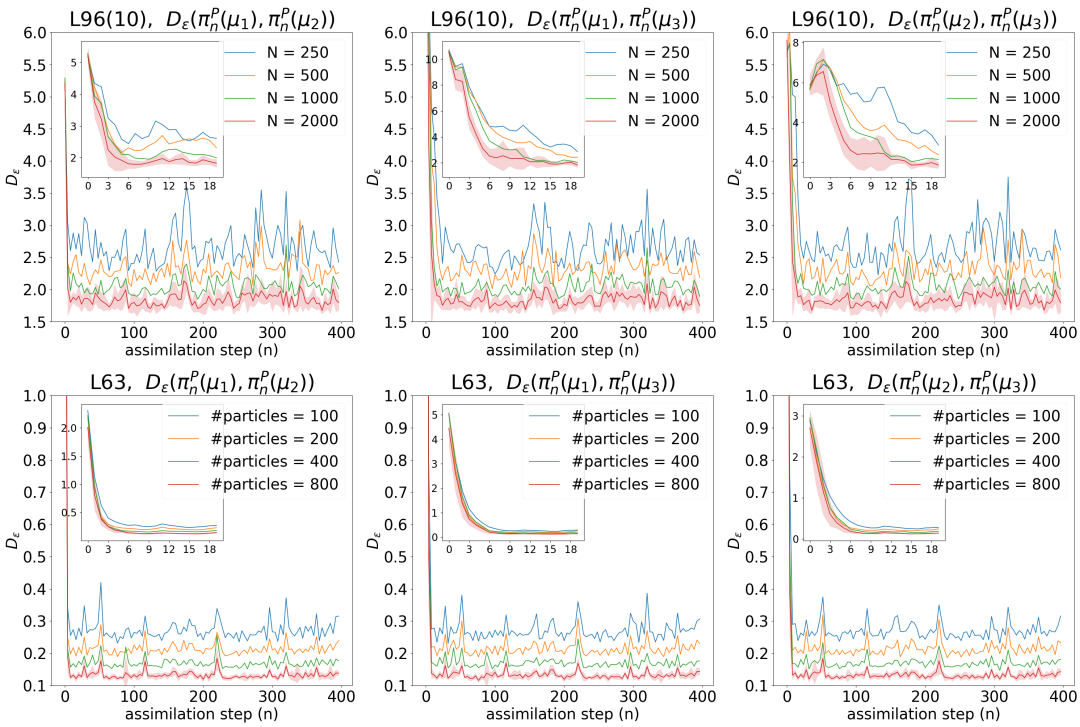

Fig. 1.2 D_ε (averaged over 10 observation realizations) for BPF for 10-dimensional $L96$ (row 1) and $L63$ (row 2) systems with observation covariance $\sigma^2 = 0.1$, for pairs of initial distributions in (1.13), with varying sample size. The line for $N = 2000$ has a band showing one standard deviation. The inset shows the drop in average D_ε during the first few assimilation steps.

Stability

Suppose the best possible filtering distribution that can be computed by the particle filter is $\pi_n^{P,*} = \lim_{N\to\infty} \pi_n^{P,N}$. Figure 1.2 is consistent with the condition

$$\lim_{n\to\infty} \liminf_{N\to\infty} \mathbb{E}[D_\varepsilon(\pi_n^{P,N}(\mu_i), \pi_n^{P,N}(\mu_j))] = 0 \;\forall\; i \neq j$$

since fixing n and increasing N results in a steady drop in D_ε averaged over observation realizations. By Theorem 1.6.4 this condition is sufficient for concluding

$$\lim_{n\to\infty} \mathbb{E}[D_\varepsilon(\pi_n^{P,*}(\mu_i), \pi_n^{P,*}(\mu_j))] = 0 \;\forall\; i \neq j$$

Dependence on observation covariance

All plots in figure 1.2 correspond to observation covariance $\sigma^2 = 0.1$. The other case $\sigma^2 = 1.0$ mentioned in 1.3.1 can be seen in figure 1.3. As we can see figure 1.3 is qualitatively similar to figure 1.2 and in our experiments, stability of particle filter

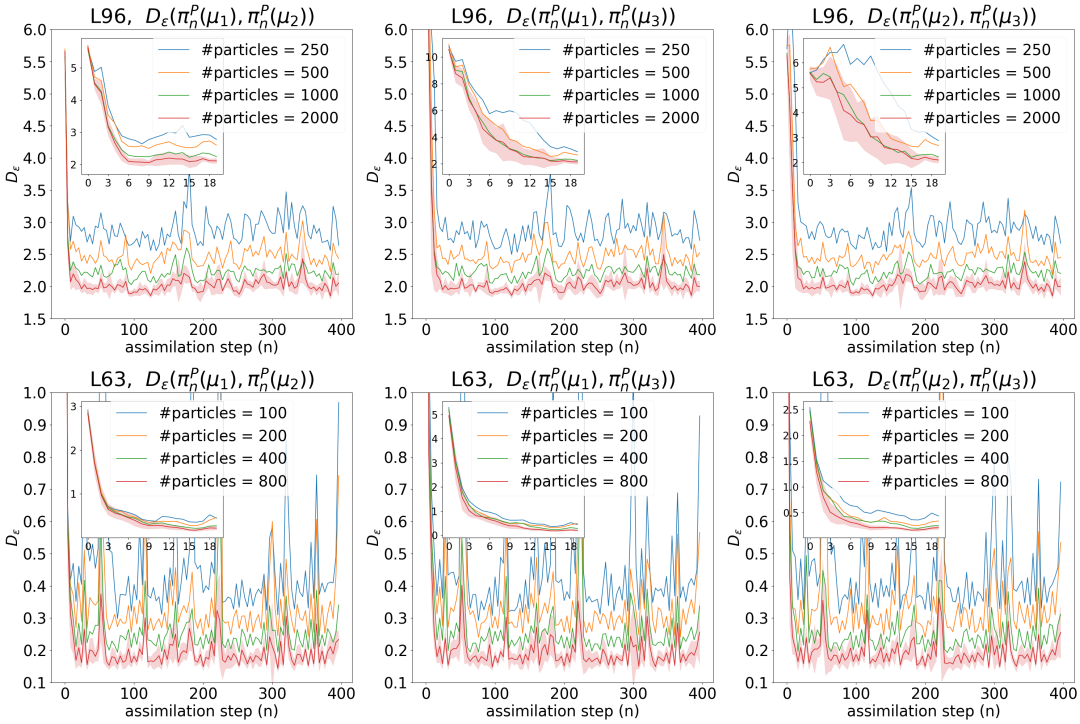

Fig. 1.3 D_ε (averaged over 10 observation realizations) for BPF for 10-dimensional $L96$ (row 1) and $L63$ (row 2) systems with observation covariance $\sigma^2 = 1.0$, for pairs of initial distributions in (1.13), with varying sample size. The line for $N = 2000$ has a band showing one standard deviation. The inset shows the drop in average D_ε during the first few assimilation steps.

was not seen to be affected by observation covariance but higher noise levels in observations naturally causes the level where the Sinkhorn divergence stabilizes, to be more noisy.

1.3.5 Stability of EnKF

Here we use the notation $\pi_n^{E,N}$ for $\hat{\pi}_n$ obtained by alogithm 1 with ensemble size N. We might omit N for brevity.

Drop in D_ε over time

From figure 1.4, we see that for every pair (μ_i, μ_j) of initial distributions $D_\varepsilon(\pi_n^E(\mu_i), \pi_n^E(\mu_j))$ decreases with time rapidly within the first 50 assimilation steps and beyond 100 assimilation steps, the observation average of D_ε for filters with different pairs initial distributions are similar and have very little variance.

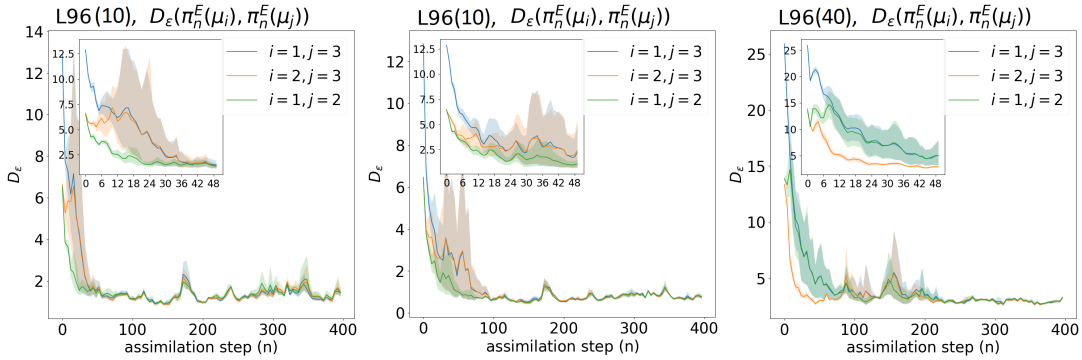

Fig. 1.4 D_ε (averaged over 10 observation realizations, with one standard deviation confidence band) for EnKF for 10-dimensional L96 with $N = 50$ with localization (left), $N = 200$ without localization (middle) for observation covariance $\sigma^2 = 0.1$, and for 40-dimensional L96 with $N = 50$ with localization (right) with observation covariance $\sigma^2 = 1.0$ for pairs of initial distributions in 1.13. The inset shows the drop in D_ε for the first 50 assimilation steps.

Variation with respect to observation realization

We see that the variation of D_ε for different observation realizations (shown by the shaded bands in left two panels in figure 1.4 and in top row in figure 1.2) is larger for the case of EnKF when compared to the particle filter for initial times (e.g. $n < 100$ for the 10-dimensional L96 model). On the other hand, for larger times (approx. $n > 100$), the variation for EnKF is significantly smaller than the particle filter.

Effect of localization

EnKF with small ensemble size needs localization which, however, is an ad-hoc procedure to prevent filter divergence and may not approximate the true filter. Figure 1.4 for 10-dimensional L96 (left panel) and for 40-dimensional L96 (right panel) shows that for $N = 50$ with localization length 4, the EnKF is stable, whereas the middle panel shows the stability (with the same configuration as the left panel) for 10-dimensional L96 without localization, but with larger ensemble size $N = 200$. This indicates that that localization does not affect EnKF's stability properties.

1.3.6 BPF vs EnKF

We now compare the BPF and EnKF for the case of 10-dimensional L96 with $\sigma^2 = 1.0$ with the same true trajectory and observation realizations. This is shown in figure 1.5. In the following discussion we assume BPF with 2000 particles to be a

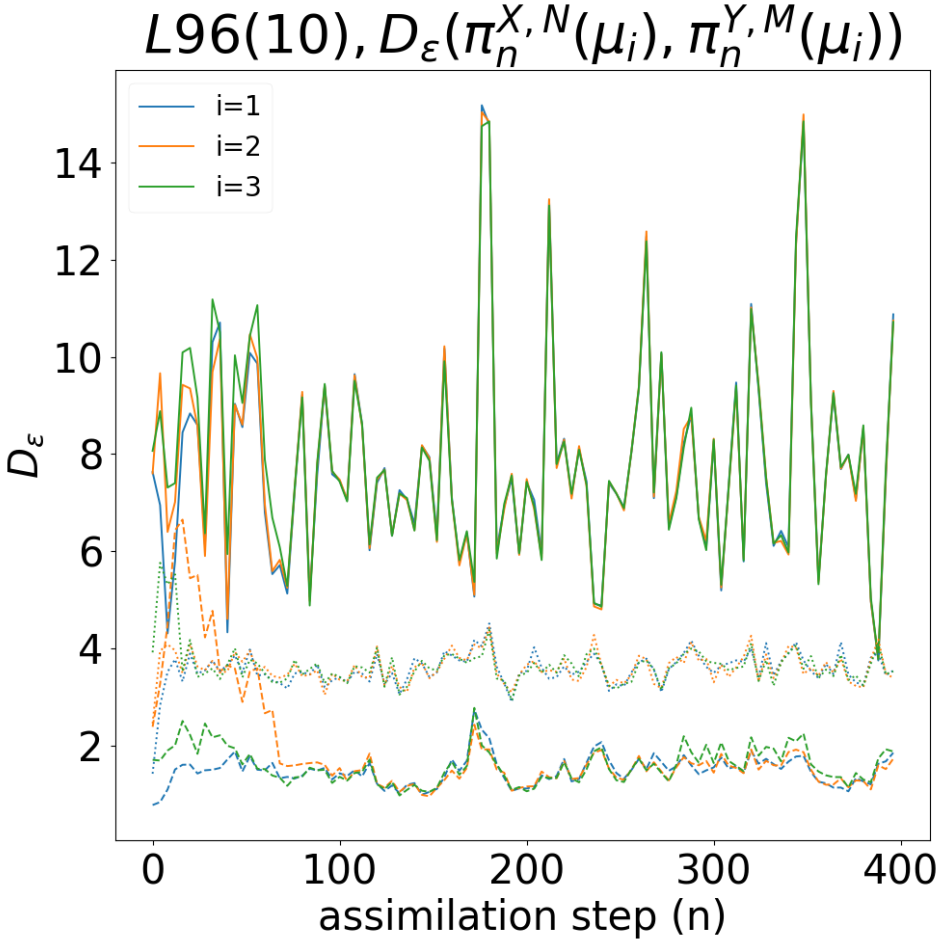

Fig. 1.5 Comparison between filters for 10-dimensional L96 and $\sigma^2 = 1.0$. The solid lines on the top show average D_ε between EnKF without localization with $N = 200$ and BPF with $M = 2000$. The dotted lines in the middle show average D_ε between BPF with $N = 250$ and BPF with $M = 2000$. The dashed lines at the bottom show average D_ε between EnKF without localization with $N = 200$ and EnKF with localization with $M = 50$. In each case, different colors are for different initial conditions from (1.13).

decent approximation for the true filter and refer to them interchangably. We note a few important points.

Poor approximation of the true filter by EnKF

The three lines towards the top show the distance between EnKF and BPF, for three different initial conditions. We see that EnKF produces distributions that are significantly different from the true filter, for all the initial distributions. But recall that for this setup, the EnKF is stable (as is the BPF too), i.e., the distance between EnKF with different initial conditions is smaller (as seen in figures 1.2-1.4) than the distance between the BPF and EnKF

BPF is closer to the true filter than is EnKF

The three lines in the middle show the distance between BPF with $N = 250$ and $N = 2000$ (putatively true filter). We see that in comparison with EnKF with $N = 200$ particles, BPF with similar ensemble size (250 particles) is much closer to the true filter.

EnKF with different ensemble size are very similar

The bottom three lines in figure 1.5 show D_ε between EnKF with different ensemble size, which shows that the EnKF is quite stable with respect to changes in the ensemble size, even though it is not very close to the true filter – thus EnKF is stable but biased, whereas BPF is stable and unbiased. A more detailed study of the reasons for this behaviour will be taken up in the future.

1.4 Robustness of Filter Stability

In this section we probe the robustness of filter stability by varying the observation gap g and the observation noise standard deviation σ. We also explore the nature of filter stability by fitting exponential curves to the expectation data. The numerical difficulty in computing filter stability has led to the adoption of RMSE error of the filter as a substitute for the expectation in (1.5) in practical settings. We, therefore, also investigate the relationship between filter RMSE and filter stability. We first describe

experimental setup, then go through the results for varying observation gap while keeping the observation noise constant and vice versa.

1.4.1 Experimental Setup

Model and Data generation

Here we use $d = 10$ dimensional Lorenz-96 [114, 167] with forcing constant $F = 10$, described in (1.11). We observe alternate coordinates starting from the first coordinate, so

$$y_{k,j} = x_{k,2j-1} + \eta_{k,j} \tag{1.14}$$

for $j = 1, 2, \cdots, q = \left\lceil \frac{d}{2} \right\rceil$ and $\eta_{k,j} \sim \mathcal{N}(0, \sigma^2)$. Throughout this section, we use $\sigma^2 = 0.2, 0.4, 0.8, 1.6$ and $g = 0.01, 0.03, 0.05, 0.07, 0.09$. The choice of the dimension $d = 10$ makes sure that we get reasonable performances from both the EnKF and the particle filter. We generate our data as we did in section 1.3.2.

Initial distributions

We use two Gaussian initial conditions. The first one μ_0 is centered at the true state with a small variance representing the case when our guess for the initial distribution is unbiased and precise. Thus we expect the filter to continue to have those properties upto some time. The second one μ_b is centered away from the true state with a significantly larger variance representing the case when our guess for the initial distribution is biased and imprecise. They are given by,

$$\begin{aligned} \mu_0 &= \mathcal{N}(x_0^{\text{true}}, 0.1 \times I_d)\,, \\ \mu_b &= \mathcal{N}(x_0^{\text{true}} + 4 \times 1_d, I_d)\,, \end{aligned} \tag{1.15}$$

where 1_d is a d-dimensional vector with all entries 1.

Metrics for probing filter stability

To probe filter stability directly using the definition 1.2.2, we study the Sinkhorn divergence $\mathbb{E}[D_\varepsilon(\pi_n(\mu_0), \pi_n(\mu_b))]$ as a function of time. It has been well-known

that in nonlinear filtering problems where the dynamic model is stochastic, under suitable additional conditions, the filter is exponentially stable and from an incorrect initial condition, it reaches stability in an exponential fashion (see, e.g., chapter 3 of [166]). Although such results are not available for the case of deterministic dynamics, exponential decay is a natural or at least desirable behaviour for the temporal behaviour of the distance between two filters starting from different initial distributions. To explore this qualitatively, we fit a curve of the following form

$$\mathbb{E}[D_\varepsilon(\pi_n(\mu_0), \pi_n(\mu_b))] = a \exp(-\lambda t) + c\,, \tag{1.16}$$

where time t = assimilation step $\times$ observation gap = ng. One of the motivation is to understand whether the exponent λ is related the dynamical quantities such as the Lyapunov exponents of the chaotic dynamical system under consideration.

In addition to stability, we also explore its relationship to the convergence of the filter mean toward the true signal as well as the uncertainty of the mean estimate. Motivated by the results about bounds on the former of these two in [94, Theorem 4.4] and [107, Theorem 4.6], we define the following two quantities.

The first quantity aims to capture the bias of the filter and is the scaled l_2 error denoted by $e_n(\nu)$.

$$\begin{aligned} e_n(\nu) &\stackrel{\text{def}}{=} \frac{1}{\sqrt{d}} \left\| \mathbb{E}_{\hat{\pi}_n(\nu)}[x_n] - \phi\left(ng, x_0^{\text{true}}\right) \right\|_2, \\ &= \left[\frac{1}{d} \sum_{i=1}^{d} \left(\frac{1}{N} \sum_{\alpha=1}^{N} x_n^{\alpha,i} - x_n^{\text{true},i} \right)^2 \right]^{1/2}, \end{aligned} \tag{1.17}$$

where $x_n^{\alpha,i}$ denotes the i-th coordinate of the α-th member of the ensemble representing the filtering distribution at time n. Thus, $e_n(\nu)$ is the distance between the true state and the filter mean divided by square root of the state space dimension d, with n denoting the assimilation step and ν the initial distribution of the filter. Note that from the results [94, Theorem 4.4] and [107, Theorem 4.6] mentioned earlier, we expect $\mathbb{E}[e_n^2(\nu)] \sim \sigma^2$ asymptotically in time.

The second quantity $s_n(\nu)$ captures the uncertainty of the filter estimate.

$$s_n(\nu) \overset{\text{def}}{=} \left[\frac{1}{d}\,\mathrm{tr}\left[\mathbb{E}_{\hat{\pi}_n(\nu)}[\left(x_n - \mathbb{E}_{\hat{\pi}_n(\nu)}[x_n]\right)\left(x_n - \mathbb{E}_{\hat{\pi}_n(\nu)}[x_n]\right)^t\right]^{1/2}\right],$$
$$= \left[\frac{1}{d}\sum_{i=1}^{d}\frac{1}{N-1}\sum_{\alpha=1}^{N}\left(x_n^{\alpha,i} - \sum_{\beta=1}^{N} x_n^{\beta,i}\right)^2\right]^{1/2}, \tag{1.18}$$

Thus, $s_n(\nu)$ is the square root of the trace of the sample covarianace of the filter. We note that we are not aware of any theoretical results that give any indication about the asymptotic in time limit of this quantity but it is reasonable to explore their relation with the observational uncertainty.

We now present the numerical results on stability for PF and EnKF, as well as the dependence of the exponential rates on the observation gap g and the observation noise strength σ. In the following figures 1.6-1.7, in top row in each of the figures, the dots represent distance $D_\varepsilon(\pi_n(\mu_0), \pi_n(\mu_b))$ between the posterior distributions at time $t = ng$ with the initial distribution μ_0 and μ_b mentioned in (1.15) versus time for 10 realizations of the observational noise. We also plot the mean $\mathbb{E}[D_\varepsilon]$ averaged over these 10 observational realizations, and the best-fit curve (1.16) in the same figures. Rows 2 and 3 contain, respectively, the expectation value of scaled l_2 error $\mathbb{E}[e_n^2]$ and uncertainty $\mathbb{E}[s_n^2]$ defined in (1.17) and (1.18) versus time. Row 4 contains scatter plot of RMSE, defined as the square root of the expected value of e_n^2, averaged over the 10 observational realizations, for the filter with biased initial distribution μ_b versus the mean D_ε between posteriors from the biased and the unbiased initial distributions.

1.4.2 Dependence on observation gap

We first discuss the results of assimilation with both PF and EnKF for a fixed observation covariance $\sigma^2 = 0.4I$ with varying observation gap. As shown in figure 1.6, the mean D_ε falls exponentially over time until reaching a stationary value for both the filters. Table 1.1 shows the values of the coefficients of the best-fit of mean D_ε versus time, according to (1.16), for different observation gaps g.

For PF, we see that the rate λ decreases with increasing observation gap g but for EnKF, the rates are not significantly affected by the change in g. The highest Lyapunov

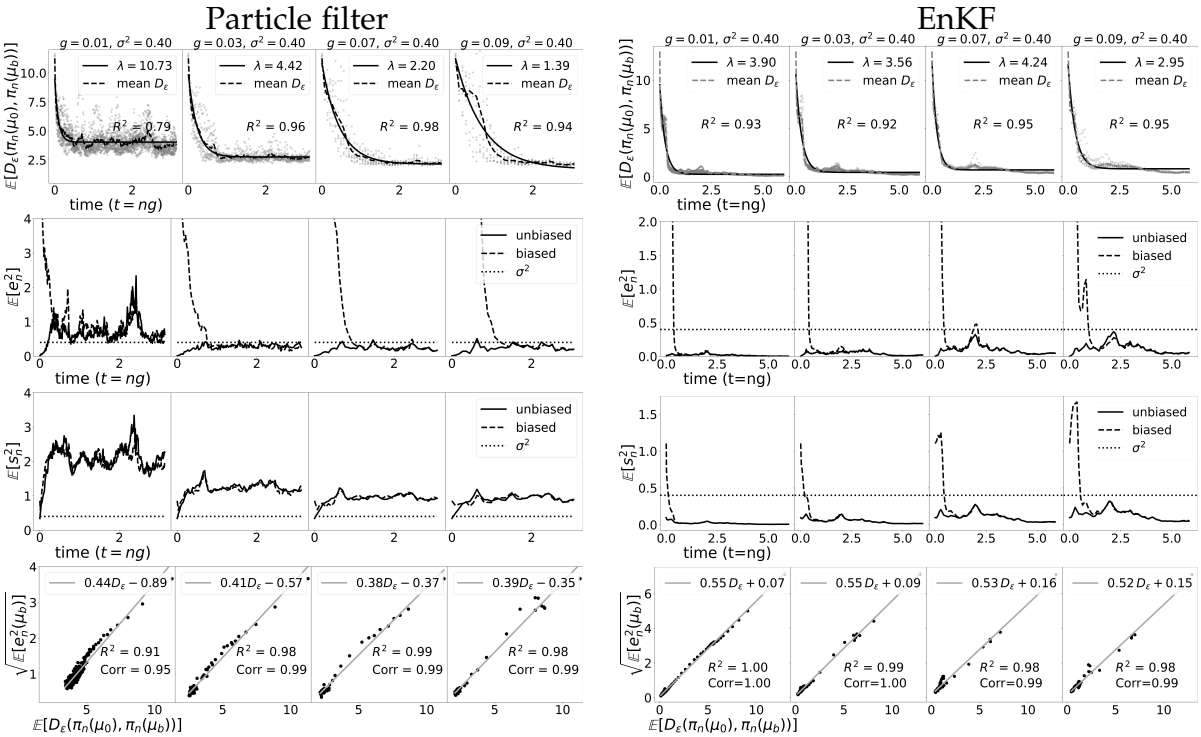

Fig. 1.6 The left and right panels show the results for PF and EnKF respectively with fixed observational error variance $\sigma^2 = 0.4$, and each column contains the results for different time between observations $g = 0.01, 0.03, 0.07, 0.09$. Row 1: Mean D_ε versus time. The dots represent 10 different realisations. The solid line is the exponential best-fit line for the mean D_ε as in (1.16). Row 2: Mean scaled l_2 error from (1.17) versus time for the two initial distributions. Row 3: Mean uncertainty from (1.18) versus time for the two initial distributions. The constant dotted line in rows 2 and 3 shows the observational error variance σ^2 for reference. Row 4: RMSE versus mean D_ε. Pearson correlation coefficient between these two quantities is depicted alongside the goodness of fit for the best-fit line.

g		**0.01**	**0.03**	**0.07**	**0.09**
a	**PF**	5.367 ± 0.080	7.077 ± 0.055	8.672 ± 0.084	9.54 ± 0.40
	EnKF	9.28 ± 0.13	10.08 ± 0.27	11.11 ± 0.32	10.76 ± 0.37
λ	**PF**	10.73 ± 0.76	4.423 ± 0.058	2.203 ± 0.021	1.392 ± 0.052
	EnKF	3.904 ± 0.085	3.56 ± 0.15	4.24 ± 0.21	2.95 ± 0.17
c	**PF**	4.03425 ± 0.00083	2.7524 ± 0.0018	2.1362 ± 0.0093	1.69 ± 0.14
	EnKF	0.258 ± 0.016	0.459 ± 0.036	0.711 ± 0.046	0.827 ± 0.064

Table 1.1 Parameters of the best-fit for the mean D_ε versus time as in (1.16) with associated confidence intervals for fixed observation covariance $\sigma^2 = 0.4$ and different observation gap g shown in the top row.

exponent for the model (with the chosen parameter value) is approximately $\lambda_{\max} = 1.7$ whereas the exponential rate λ for EnKF is seen to be in the range of $(3.0, 4.2)$, close to $2\lambda_{\max}$, indicating a possible close relation between the dynamics and the EnKF that

could be explored further. The exponential rate for the PF does not seem to show such a relation.

Another difference between PF and EnKF may also be noted: with increasing observation gap, the stationary value c for the PF decreases whereas it increases for the EnKF. Also, the stationary values c of the D_ε for EnKF are significantly lower compared to corresponding values for PF. These asymptotic values of D_ε over time for both the filters can be explained by their mean posterior covariance using the following argument.

A characteristic of the numerical distance D_ε is that for two different *i.i.d.* samples drawn from the same probability distribution, D_ε has a nonzero positive value. Statistically, D_ε between two empirical measures approach this value at which they are essentially representing the same distribution and cannot be distinguished. For a fixed dimension d and sample size N, this value increases with increasing covariance of the distribution, see section 1.3.3 for an in-depth discussion.

The mean posterior covariance trace is directly proportional to the s_n^2. With increasing observation gap g, the mean uncertainty decreases for PF while it increases for EnKF. Hence the asymptotic value of D_ε decreases with increasing observation gap for PF, while for the latter, it increases. We note that the previous paragraph explains the asymptotic value of D_ε, but the difference in the behaviour of the filter uncertainty s_n for PF and EnKF as a function of observation gap needs to be explored further.

The scaled l_2 errors also reach an asymptotically constant value around the same time the corresponding filters stabilize in D_ε. The scatter plots for the RMSE against the D_ε shows strong correlation between them. This suggests that we can use the RMSE over time as a good indicator for the time when the filter stabilizes. Note that the methods in this chapter give us a direct way to check whether a numerical filter is stable for a given dynamical and observational model, and the relation between the filter stability and the l_2 error or bias e_n implies that a stable filter may be expected to be an accurate one.

We note that in the plots in the bottom row, the cluster at the bottom left corresponds to the time after which both RMSE and the D_ε have reached their stationary values. Even for two different biased initial distributions for the filter, there is a finite transient growth after which the D_ε falls exponentially [122]. Although not shown

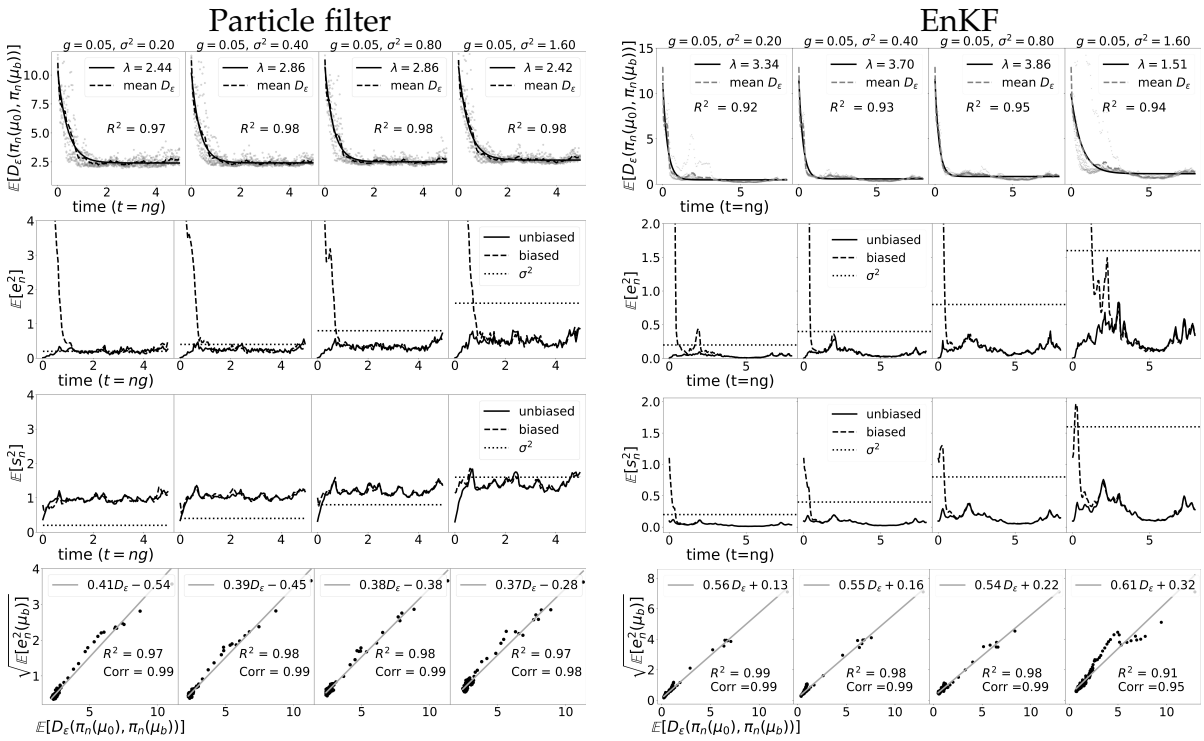

Fig. 1.7 Same as in figure 1.6 with the left and right panels showing the results for PF and EnKF respectively, but with fixed time between observations $g = 0.05$, and each column containing the results for different observational error variances of $\sigma^2 = 0.2, 0.4, 0.8, 1.6$.

here, the linear regime is still present in the scatter plot of the RMSE of either one of them versus the D_ε in those cases.

1.4.3 Dependence on observation noise

We now discuss the results of numerical experiments with fixed observation gap $g = 0.05$, with varying observation covariance $\sigma^2 = 0.2, 0.4, 0.8, 1.6$. In figure 1.7, we again note the exponential decrease of the distance D_ε over time until it reaches a stationary value c. The parameter values obtained for the best-fit for different observation covariance are shown in table 1.2.

In contrast with the case of varying observational gap, the exponential rates for the PF stability are not affected by the change in observational uncertainty. While the rates for EnKF are again close to twice the Lyapunov exponent, the rates for PF are smaller.

The scaled l_2 error and the D_ε achieve their stationary value around the same time as in the former case of fixed observation. As expected, this asymptotic value c as

σ^2		**0.2**	**0.4**	**0.8**	**1.6**
a	**PF**	7.842 ± 0.058	7.730 ± 0.046	8.153 ± 0.044	8.038 ± 0.048
	EnKF	10.61 ± 0.32	10.84 ± 0.30	10.69 ± 0.23	8.75 ± 0.22
λ	**PF**	2.442 ± 0.016	2.858 ± 0.017	2.859 ± 0.015	2.416 ± 0.012
	EnKF	3.34 ± 0.16	3.70 ± 0.16	3.86 ± 0.14	1.507 ± 0.062
c	**PF**	2.4050 ± 0.0022	2.4144 ± 0.0015	2.5051 ± 0.0014	2.6554 ± 0.0019
	EnKF	0.470 ± 0.039	0.579 ± 0.035	0.838 ± 0.027	1.148 ± 0.041

Table 1.2 Parameters of the best-fit for the mean D_ε versus time as in (1.16) with associated confidence intervals for fixed observation gap $g = 0.05$ and different observational error covariance σ^2 shown in the top row.

well as the asymptotic values of the uncertainty s_n and the bias e_n all increase with increasing σ^2 for both PF and EnKF.

We also see near perfect correlation in the scatter plots for the RMSE versus mean D_ε as for both PF and EnKF, we get Pearson correlation coefficient very close to 1. We remark that in our numerical experiments, either with varying observational time gap or with varying observational covariance, we did not notice any relation between stability and posterior uncertainty or precision, i.e., there did not seem to be any relation between D_ε and s_n.

1.5 Summary and Future Work

The main focus of this chapter is the numerical study of nonlinear filter stability. For this purpose, we use the recently developed Sinkhorn algorithm to calculate an approximation of the Wasserstein distance between Monte Carlo samples from probability distributions. This allows us to directly assess the stability by computing the expected value, averaged over multiple observation realizations, of the distance between filtering distributions as function of time. We also studied extensively the dependence of stability properties on two main parameters, the time between observations and the observational error covariance.

Our results provide strong numerical indication of exponential stability of PF and EnKF for deterministic chaotic dynamical systems. For a fixed observational covariance, as the gap between the observations increases, the exponential rate of decay of the distance D_ε between two filters decreases for the particle filter whereas it remains approximately constant and close to twice the Lyapunov exponent for

the ensemble Kalman filter. Further exploration of relationships between the chaotic properties of the system and the exponential stability is an interesting direction for future work. Mathematical proofs of stability of these numerical algorithms is another interesting open area of further research.

Further, with increasing observational gap, the filter uncertainty and bias decrease for PF but increases for the EnKF. In general, the EnKF has significantly smaller uncertainty and bias as compared to the PF. More extensive numerical explorations and associated theoretical studies to understand the differences between these two important numerical filters - the particle filter and the ensemble Kalman filter - is an important direction for further investigations.

The other main focus was to establish a relation between the filter stability and filter convergence. We show that for a wide variety of parameters, the distance between two filters started with different initial conditions is directly proportional to the l_2 error or the RMSE between the filter mean and the true underlying trajectory. The significance of this result is that the techniques developed in this chapter provide tools to check filter stability, with synthetic or real observations for a given dynamical and observational model, even before starting to use a filter with actual observations, while the results connecting stability with convergence of the l_2 error can then be used to give indication of whether the filter is also accurate. This is particularly useful in real-world scenarios where one does not have access to the true trajectory of the dynamical system under consideration and hence cannot compute the RMSE. In these cases one can still perform assimilation with two different initial conditions and compute the distance between the resulting filtering distributions and how this distance changes in time can indicate how the accuracy of the filter changes with time since these two quantities are strongly correlated. The mechanism behind this correlation is an important direction for future research.

1.6 Appendix

1.6.1 Properties of Sinkhorn divergence

Lemma 1.6.1 *If α, β, α_m are probability measures on a compact set $\chi \subset \mathbb{R}^d$ then*

$$0 = S_\varepsilon(\beta, \beta) \leq S_\varepsilon(\alpha, \beta) \tag{1.19}$$

$$\alpha = \beta \iff S_\varepsilon(\alpha, \beta) = 0 \tag{1.20}$$

$$\alpha_m \xrightarrow{weak^*} \alpha \iff S_\varepsilon(\alpha_m, \alpha) \to 0 \tag{1.21}$$

Proof. See Theorem 1 in [55]. □

Lemma 1.6.2 *If $\alpha_m, \beta_m, \alpha, \beta$ are probability measures supported on a compact set $\chi \subset \mathbb{R}^d$ such that $\alpha_m \xrightarrow{weak^*} \alpha$ and $\beta_m \xrightarrow{weak^*} \beta$ then $\lim_{m\to\infty} S_\varepsilon(\alpha_m, \beta_m) = S_\varepsilon(\alpha, \beta)$.*

Proof. Direct consequence of Proposition 13 in [55]. □

Theorem 1.6.3 *If $\alpha_m = \frac{1}{m}\sum_{i=1}^m \delta_{x_i^m}$ and $\beta_m = \frac{1}{m}\sum_{i=1}^m \delta_{y_i^m}$ are sampling distributions for the same underlying probability distribution μ which is supported on a compact set $\chi \subset \mathbb{R}^d$ then $\lim_{m\to\infty} S_\varepsilon(\alpha_m, \beta_m) = 0$.*

Proof. Direct consequence of Lemmas 1.6.1 and 1.6.2. □

Theorem 1.6.4 *If $\alpha_{m,n}, \beta_{m,n}, \alpha_n, \beta_n$ are random probability measures supported on a compact set $\chi \subset \mathbb{R}^d$ such that $\alpha_{m,n} \xrightarrow{weak^*} \alpha_n$ and $\beta_{m,n} \xrightarrow{weak^*} \beta_n$ as $m \to \infty$ and*

$$\lim_{n\to\infty} \liminf_{m\to\infty} \mathbb{E}[D_\varepsilon(\alpha_{m,n}, \beta_{m,n})] = 0$$

then, $\lim_{n\to\infty} \mathbb{E}[D_\varepsilon(\alpha_n, \beta_n)] = 0$.

Proof.

$$\begin{aligned}
&\lim_{n\to\infty} \mathbb{E}[D_\varepsilon(\alpha_n, \beta_n)] \\
&= \lim_{n\to\infty} \mathbb{E}[\lim_{m\to\infty} D_\varepsilon(\alpha_{m,n}, \beta_{m,n})] \quad \text{by Lemma 1.6.2} \\
&\leq \lim_{n\to\infty} \liminf_{m\to\infty} \mathbb{E}[D_\varepsilon(\alpha_{m,n}, \beta_{m,n})] = 0 \quad \text{by Fatou's lemma}
\end{aligned}$$

□

1.6.2 Definition 1.2.2 implies definition 1.2.1

We have introduced two different definitions of filter stability in section 1.2.2. Definition 1.2.2 is used for the numerical computations in this chapter while the other definition 1.2.1 is a commonly used definition using the notion of weak convergence. In this appendix, we prove that 1.2.2 is stronger in the sense that it implies the other.

Theorem 1.6.5 *Let (M, d) be a compact metric space and $\{\alpha_n\}, \{\beta_n\}$ be sequences of random probability measures on M with finite first and second moments and*

$$\lim_{n\to\infty} \mathbb{E}[D(\alpha_n, \beta_n)] = 0\,, \tag{1.22}$$

where $D = W_1$ or W_2. Then for all globally Lipschitz functions $f : M \to \mathbb{R}$ we have

$$\lim_{n\to\infty} \mathbb{E}\left[\left|\int_M f\, d\alpha_n - \int_M f\, d\beta_n\right|\right] = 0\,. \tag{1.23}$$

Proof. Let $S = \{f : M \to \mathbb{R} : \mathrm{Lip(f)} \leq 1\}$ be the set of Lipschitz functions with Lipschitz constant not greater than 1. If $f \in S$, by Kantorovich-Rubinstein duality we have,

$$\mathbb{E}\left[\left|\int_M f\, d\alpha_n - \int_M f\, d\beta_n\right|\right] \leq \mathbb{E}\left[\sup_{g\in S}\left|\int_M g\, d\alpha_n - \int_M g\, d\beta_n\right|\right] = \mathbb{E}[W_1(\alpha_n, \beta_n)]\,, \tag{1.24}$$

which proves the assertion for the case $D = W_1$ for $f \in S$. For $D = W_2$, recall that for two measures μ, ν on M with finite first and second moments,

$$W_p(\mu, \nu)^p = \inf_{\mathrm{S}} \mathbb{E}[d(X, Y)^p]\,, \tag{1.25}$$

where $\mathbb{S} = \{\pi : \pi \text{ is the joint distribution of } X, Y \text{ with marginals } \mu, \nu \text{ respectively}\}$. Therefore,

$$W_1(\mu,\nu)^2 = \left(\inf_{\mathbb{S}} \mathbb{E}[d(X,Y)]\right)^2 = \inf_{\mathbb{S}} \mathbb{E}[d(X,Y)]^2 \leq \inf_{\mathbb{S}} \mathbb{E}[d(X,Y)^2], \quad \text{[by Jensen's inequality]} \tag{1.26}$$

$$\leq W_2(\mu,\nu)^2, \quad \text{[by (1.25)]} \tag{1.27}$$

Combining (1.24) and (1.27), we immediately see that the assertion is true for the case $D = W_2$ for $f \in S$.

Now pick a function $\tilde{f} : M \to \mathbb{R}$ with $\mathrm{Lip}(\tilde{f}) \leq K$ for some $K > 0$. Define $f = \frac{\tilde{f}}{K}$. Clearly, $f \in S$ and therefore,

$$\lim_{n\to\infty} \mathbb{E}\left[\left|\int_M \tilde{f}\, d\alpha_n - \int_M \tilde{f}\, d\beta_n\right|\right] = K \lim_{n\to\infty} \mathbb{E}\left[\left|\int_M f\, d\alpha_n - \int_M f\, d\beta_n\right|\right] = 0\,, \tag{1.28}$$

□

Theorem 1.6.5 immediately yields the following stronger statement.

Corollary 1.6.6 *Under identical conditions,* (1.23) *holds for any continuous* $f : M \to \mathbb{R}$.

Proof. Note that real-valued, continuous functions on M are uniform limits of locally Lipschitz functions on M. Since M is compact and locally Lipschitz functions on compact metric spaces are globally Lipschitz, real-valued, continuous functions on M are uniform limits of globally Lipschitz functions on M.

Let $\alpha_n - \beta_n = \gamma_n$ for brevity and let f be a real-valued, continuous function on M.

$$\left|\int_M f\, d\alpha_n - \int_M f\, d\beta_n\right| = \left|\int_M f\, d\gamma_n\right| = \left|\int_M f^+\, d\gamma_n^+ + \int_M f^-\, d\gamma_n^- - \int_M f^-\, d\gamma_n^+ - \int_M f^+\, d\gamma_n^-\right|, \tag{1.29}$$

where $+, -$ denote the positive and negative parts of f and γ_n. We now consider the case when the last sum of four terms is positive. The other case when it is negative is very similar.

Pick sequences of non-negative globally Lipschitz functions $f_m^+ \uparrow f^+$ and $f_m^- \uparrow f^-$. Then by using dominated convergence theorem and Fatou's lemma, we see that

$$\mathbb{E}\left[\int_M f^+ \, d\gamma_n^+\right] = \mathbb{E}\left[\lim_{m\to\infty} \int_M f_m^+ \, d\gamma_n^+\right] \leq \mathbb{E}\left[\liminf_{m\to\infty} \int_M f_m^+ \, d\gamma_n^+\right] \leq \liminf_{m\to\infty} \mathbb{E}\left[\int_M f_m^+ \, d\gamma_n^+\right]. \tag{1.30}$$

Using very similar arguments for each of the four terms and adding the four inequalities, we see that

$$\mathbb{E}\left[\left|\int_M f \, d\alpha_n - \int_M f \, d\beta_n\right|\right] \leq \liminf_{m\to\infty} \mathbb{E}\left[\int_M f_m^+ \, d\gamma_n^+ \int_M f_m^- \, d\gamma_n^- - \int_M f_m^- \, d\gamma_n^+ - \int_M f_m^+ \, d\gamma_n^-\right]. \tag{1.31}$$

Define, $f_m = f_m^+ - f_m^-$ which is globally Lipschitz since sum of globally Lipschitz functions are globally Lipschitz. Since the RHS of (1.31) is non-negative, it must be equal to $\liminf_{m\to\infty} \mathbb{E}\left[\left|\int_M f_m \, d\gamma_n\right|\right]$. Therefore,

$$\mathbb{E}\left[\left|\int_M f \, d\alpha_n - \int_M f \, d\beta_n\right|\right] \leq \liminf_{m\to\infty} \mathbb{E}\left[\left|\int_M f_m \, d\alpha_n - \int_M f_m \, d\beta_n\right|\right]. \tag{1.32}$$

Hence, given $\varepsilon > 0$, $\exists\, m_0$ such that $\forall\, m > m_0$,

$$\mathbb{E}\left[\left|\int_M f_m \, d\alpha_n - \int_M f_m \, d\beta_n\right|\right] > \mathbb{E}\left[\left|\int_M f \, d\alpha_n - \int_M f \, d\beta_n\right|\right] - \varepsilon. \tag{1.33}$$

Using Theorem 1.6.5, we see that the limit $n \to \infty$ of LHS is 0 which gives

$$0 \geq \lim_{n\to\infty} \mathbb{E}\left[\left|\int_M f \, d\alpha_n - \int_M f \, d\beta_n\right|\right] - \varepsilon. \tag{1.34}$$

Since ε can be chosen arbitrarily, we have

$$\lim_{n\to\infty} \mathbb{E}\left[\left|\int_M f \, d\alpha_n - \int_M f \, d\beta_n\right|\right] = 0. \tag{1.35}$$

□

Theorem 1.6.7 *Assume the filtering distributions $\hat{\pi}_n(\mu), \hat{\pi}_n(\nu)$ are supported on a compact subset of $\mathbb{R}^d$ and $D = W_2$ or W_1. If (1.5) holds then so does (1.4).*

Proof. Direct consequence of Corollary 1.6.6. □

1.6.3 Convergence in D_ε and W_2

Theorem 1.6.8 *Suppose* $\exists$ *a sequence* $p_k : \lim_{k\to\infty} p_k = 0$ *and* $a_{kn} := \mathbb{E}[D_{p_k}(\alpha_n, \beta_n)]$ *is monotone decreasing in* k *or,* $a_{kn} \geq a_{k+1,n}\ \forall\ k, n$ *where* $\{\alpha_n\}, \{\beta_n\}$ *are sequences of random measures. If for some* j

$$\limsup_{n\to\infty} \mathbb{E}[D_{p_j}(\alpha_n, \beta_n)] \leq \delta, \tag{1.36}$$

then

$$\limsup_{n\to\infty} \mathbb{E}[W_2(\alpha_n, \beta_n)] \leq \delta \tag{1.37}$$

Proof. Recall that $\lim_{k\to\infty} D_{p_k}(\alpha_n, \beta_n) = W_2(\alpha_n, \beta_n)$, see for example, Theorem 1 in [62]. By Fatou's lemma,

$$\limsup_{n\to\infty} \mathbb{E}[W_2(\alpha_n, \beta_n) \leq \limsup_{n\to\infty} \liminf_{k\to\infty} \mathbb{E}[D_{p_k}(\alpha_n, \beta_n))] \leq \limsup_{n\to\infty} \mathbb{E}[D_{p_j}(\alpha_n, \beta_n))] \leq \delta \tag{1.38}$$

□

Theorem 1.6.8 is helpful in discussing the convergence of measures in Wasserstein metric by looking at convergence in D_ε for a fixed ε. The assumption about existence of a monotone decreasing subsequence is consistent with numerical experiments as shown in figure 1.8. Although Sinkhorn divergence is zero when we are comparing two identical distributions, while comparing two different samples from the same distribution we would expect a non-zero divergence which is why Theorem 1.6.8 is stated in terms of a non-zero bound δ. We note that a more detailed discussion about the behaviour of this non-zero divergence can be found in section 1.3.3. With increasing sample size this non-zero divergence approaches zero. For a detailed look at the theoretical sample-complexity of Sinkhorn divergence see chapter 3 of [61].

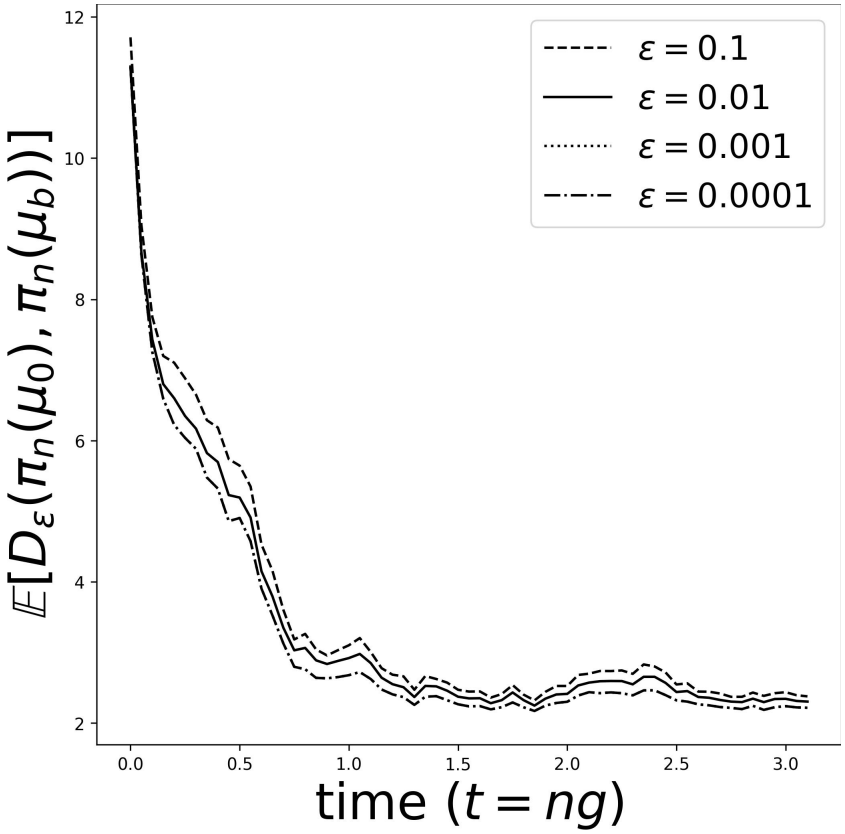

Fig. 1.8 Change in mean D_ε with ε. The smallest two ε values produce identical mean lines. The filtering distributions are generated by particle filter for observation gap $= 0.05$ and observation covariance $= 0.4$.

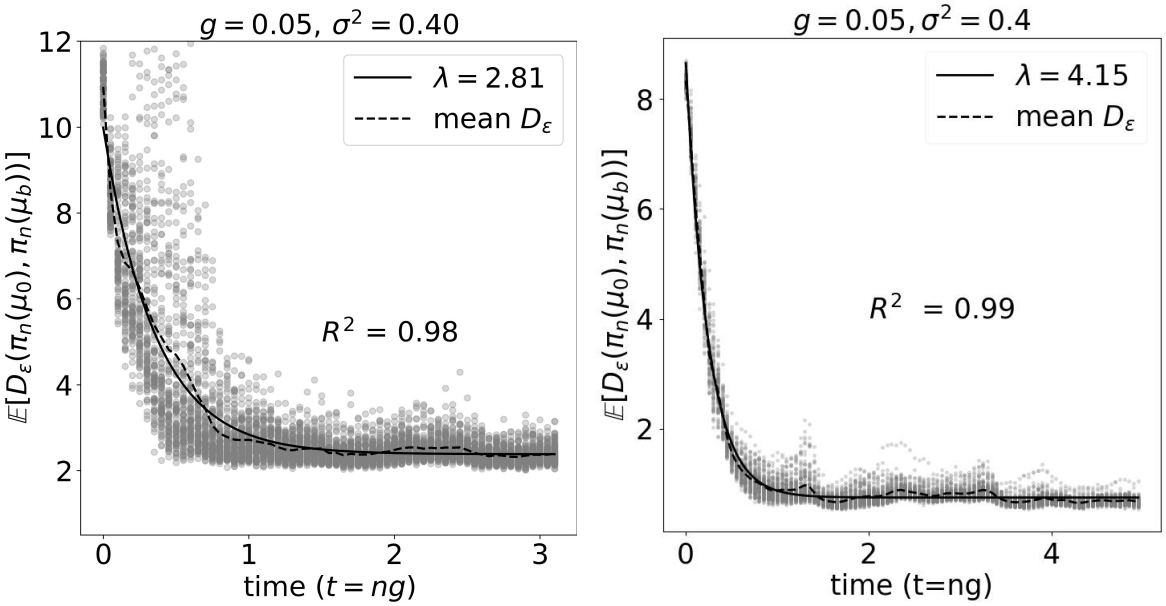

Fig. 1.9 Mean D_ε for 100 observations for PF (left panel) and EnKF (right panel) with observation gap $= 0.05$ and observation covariance $= 0.4$.

1.6.4 Effect of varying the sample-size

We find that averaging over 10 observation realizations (for the expectation in (1.5)) is sufficient for this study. To illustrate this, we repeated a representative experiment for a larger sample size of 100 and the resulting plots of the distance versus time for observation gap $g = 0.05$ and $\sigma^2 = 0.4$ for the two filters are shown in figure 1.9. Comparing with the plots in first row, second column of the particle filter and EnKF results in figure 1.7, we see that the results are qualitatively identical and quantitatively near-identical. Thus the choice of averaging over 10 realizations suffices to capture the statistical features of the quantities we are studying.

Chapter 2

Stochastic dynamical systems are intimately related to Fokker-Planck equations. The prediction step in the filtering algorithm can be thought of as roughly solving a Fokker-Planck equation when the underlying dynamics is noisy. These equations are therefore of great interest when it comes to developing new filtering algorithms. Apart from filtering, their sheer ubiquity makes studying them worthwhile. In this chapter we devise a deep learning algorithm to find non-trivial zeros of Fokker-Planck operators when the drift is non-solenoidal. We demonstrate the efficacy of our algorithm for problem dimensions ranging from 2 to 10. Our method scales linearly with dimension in memory usage. We show that this method produces better approximations compared to Monte Carlo methods, for the same overall sample sizes, even in low dimensions. Unlike the Monte Carlo methods, our method gives a functional form of the solution. We also demonstrate that the associated loss function is strongly correlated with the distance from the true solution, thus providing a strong numerical justification for the algorithm. Moreover, this relation seems to be linear asymptotically for small values of the loss function.

Chapter 2

Learning zeros of Fokker-Planck operators

2.1 Introduction

Many real world problems can be modelled as the response of nonlinear systems to random excitations and such systems have been a topic of interest for a long time. Stochastic differential equations (SDE) provide the natural language for describing these systems. Although SDEs have their origins in the study of Brownian motion by Einstein and Smoluchowski, it was Itô who first developed the mathematical theory. Since then SDEs have extensively appeared in physics [108], [161], [82], biology [2], mathematical finance [49], [93], and many other fields [131], [60]. The probability density associated with an Itô SDE evolves in time according to a Fokker-Planck equation (FPE) or Kolmogorov forward equation. A stationary FPE (SFPE) can be solved analytically when the corresponding Itô SDE has a drift term that can be represented as the gradient of some potential [149]. But the same is not true when the drift is not of the aforementioned form. And the time-dependent FPE does not admit a closed form solution in general even when the drift is integrable. Since the solution of an FPE is a probability density, the boundary condition is replaced by an integral condition which is extremely hard to implement in dimensions larger than 2.

In recent times deep learning has been successfully used to solve high-dimensional PDEs [159], [70], [144]. Although universal approximation theorems [139], [115], [46], [101] guarantee existence of neural networks that approximate the true solution well,

due to the non-convex nature of loss functions one cannot guarantee convergence of neural networks to the true solution during training in many instances [103], [9]. Moreover, these methods are almost always used for PDEs with simple boundary conditions not containing integral terms which makes applying them for FPEs challenging. But even though deep learning solutions to PDEs is fraught with challenges, it is a worthwhile paradigm to work in while dealing with high-dimensional PDEs for the following reasons. Most deep learning methods are mesh-free [16] and can deal with the curse of dimensionality much better than classical methods [37]. Moreover, some of them focus on computing pointwise solutions to PDEs [70] which albeit non-standard, might be the only practical and efficient approach in high dimensions.

The goal of this chapter is to devise a reliable, mesh-free deep learning algorithm to find non-trivial zeros of high-dimensional Fokker-Planck operators. In a sequel we will devise a method for solving high-dimensional time-dependent FPEs using these zeros. Although our algorithm is capable of handling any non-solenoidal drift function, as examples we focus on problems where the underlying ODE system possesses a global attractor. These systems are often used to make simple models in the earth sciences and provide ideal test cases for non-linear filtering algorithms [28]. We solve 2,3,4,6,8 and 10 dimensional problems with our method. We compare our method with Monte Carlo for $d = 2$, investigate how the loss function and the distance from the true solution are related to each other and explore how our method scales with dimension.

2.2 Problem statement

In this chapter we are interested in the stationary Fokker-Planck equation

$$\begin{aligned} \mathcal{L}p &\stackrel{\text{def}}{=} -\sum_{i=1}^{d} \frac{\partial(\mu_i p)}{\partial x_i} + \sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial(D_{ij} p)}{\partial x_i \partial x_j} = 0, \quad \mathbf{x} \in \mathbb{R}^d \\ \int_{\mathbb{R}^d} p(\mathbf{x})\, d\mathbf{x} &= 1, \quad p(\mathbf{x}) \geq 0 \ \ \forall\, \mathbf{x} \in \mathbb{R}^d \end{aligned} \tag{2.1}$$

Here $D = \frac{1}{2}\sigma\sigma^\top$, $\mu \in C^1(\mathbb{R}^d; \mathbb{R}^d)$ is a non-solenoidal vector field i.e. $\nabla \cdot \mu \not\equiv 0$ in $\mathbb{R}^d$ and $\sigma \in C(\mathbb{R}^d; \mathbb{R}^{d\times d})$ is a matrix-valued function such that D is positive-definite. The operator $\mathcal{L}$ is known as the *Fokker-Planck operator* (FPO). The goal of this work is to devise an algorithm to find a non-trivial zero of $\mathcal{L}$ in a mesh-free manner that works

well in dimensions that are challenging for classical PDE-solvers. Note that when μ is solenoidal The motivations behind choosing to find a non-trivial zero of $\mathcal{L}$ rather than solving (2.1) are as follows.

- Numerical integration suffers from the curse of dimensionality [71] and consequently the normalization constraint is extremely challenging to implement in high dimensions.
- The end goal is to devise an algorithm to solve time-dependent FPEs with unique solutions which we describe in the sequel. It turns out, knowing a non-trivial zero of $\mathcal{L}$, even if unnormalized, is enough to find the normalized solution to the time-dependent FPE.
- When μ is solenoidal every constant function is a zero of $\mathcal{L}$. In this special case if the corresponding time-dependent FPE has a unique solution, we would not require a non-trivial zero of $\mathcal{L}$ to calculate it, as we will see in the sequel.
- Lastly, rather than trying to force normalization during the computation of a non-trivial zero, it is much more economical to integrate the zero at the end to find the normalization constant at a one-time cost. Quasi Monte Carlo [109] or deep learning methods like i-flow [59] can be used for this purpose.

Although our method is perfectly valid for any matrix-valued σ that gives rise to a positive definite D, in the demonstrations we use the form $\sigma = aI_d$ where a is a positive constant and I_d is the $d \times d$ identity matrix. This allows us to abuse notation and use σ and D as scalar quantities. With this simplification our equation becomes,

$$\mathcal{L}p = -\nabla \cdot (\mu p) + D\Delta p = 0 \tag{2.2}$$

where Δ is the Laplacian operator.

Since we approximate a solution to (2.2) with a neural network, it is sensible to consider strong solutions. We therefore restrict our search space of functions to $W^{1,2}_{\text{loc}}(\mathbb{R}^d) \cap C^2(\mathbb{R}^d)$. Since the superscripts for Sobolev spaces have been used interchangeably in literature, to avoid confusion we define $W^{1,2}(\mathbb{R}^d)$ as

$$W^{1,2}(\mathbb{R}^d) \stackrel{\text{def}}{=} \{f \in L^2(\mathbb{R}^d) : \|\nabla f\|_2 \in L^2(\mathbb{R}^d)\} \tag{2.3}$$

Sobolev spaces are frequently encountered while studying elliptic PDEs and therefore are very well-studied [23], [96]. This choice of function space enables us to prove uniqueness of solutions to the SFPEs that we will encounter in this chapter, see appendix 2.9.1 for details. Moreover, density of arbitrary-size neural networks in the space of continuous functions [139] and non-closedness of fixed-size neural networks in Sobolev spaces [117] are good justifications for our algorithm, see section 2.5.2, making $W^{1,2}_{\text{loc}}(\mathbb{R}^d) \cap C^2(\mathbb{R}^d)$ an ideal function space to work with.

2.3 Examples

From an algorithmist's perspective it is important to have access to a class of equations on which our algorithm can be validated easily. Since classical methods do not work satisfactorily for our problem dimensions, the only other way is to use those equations, for which the analytical solutions are known, as the validating examples.

2.3.1 Gradient systems

To that end a very convenient class of equations is where the drift μ can be written as the gradient of a potential function,

$$\mu = -\nabla V \tag{2.4}$$

To see why this structure of μ leads to analytical solutions of SFPEs, note that if $p = e^f$ is a solution to the SFPE then according to (2.2) we have

$$-\nabla \cdot (e^f \mu) + D\Delta e^f = 0 \tag{2.5}$$

$$\implies -\nabla \cdot \mu - \mu \cdot \nabla f + D\left(\|\nabla f\|_2^2 + \Delta f\right) = 0 \tag{2.6}$$

$$\implies (\nabla + \nabla f) \cdot (D\nabla f - \mu) = 0 \tag{2.7}$$

$$\implies (\nabla + \nabla f) \cdot (\nabla(Df + V)) = 0 \tag{2.8}$$

so we can find one solution by simply setting the second term in the RHS of (2.8) to be zero which gives us

$$\nabla(Df + V) = \mathbf{0} \tag{2.9}$$

$$\implies f = \ln c - \frac{V}{D} \tag{2.10}$$

$$\implies p = c \exp\left(-\frac{V}{D}\right) \tag{2.11}$$

where c is the normalizing constant. So a solution in this special case is already known up to the normalizing constant. We refer to a system satisfying (2.4) as a *gradient system*. In this chapter we use the following gradient systems to validate our algorithm in high dimensions.

2D ring system

For $d = 2$, $V = (x^2 + y^2 - 1)^2$ and $\mu = -\nabla V$ we get the following SFPE,

$$4(x^2 + y^2 - 1)\left(x\frac{\partial p}{\partial x} + y\frac{\partial p}{\partial y}\right) + 8(2x^2 + 2y^2 - 1)p + D\Delta p = 0 \tag{2.12}$$

This system possesses a unique solution concentrated on around the unit circle. The proof of uniqueness using the method of Lyapunov functions is given in the appendix 2.9.1. The corresponding ODE system has the unit circle as a global attractor. This is a recurring theme in all of our example problems. Such systems with attractors are of great interest in the study of dynamical systems [133] as well as filtering theory [100]. We solve this system for $D = 1$.

2nD ring system

We can daisy-chain the previous system to build decoupled systems in higher dimensions. In this case the potential is given by

$$V(\mathbf{x}) = \sum_{i=0}^{\frac{d}{2}-1} (x_{2i}^2 + x_{2i+1}^2 - 1)^2, \qquad d = 2n \tag{2.13}$$

Since our algorithm does not differentiate between coupled and decoupled systems, this example serves as a great high-dimensional test case. In a sequel we show

how to solve the time-dependent FPEs which is intimately related to the method presented here and being decoupled, this system presents a great way to verify the time-dependent algorithm. This is important since analytical solutions for time-dependent FPEs are not known in general even for gradient systems. Uniqueness of solution for the 2nD ring system directly follows from the uniqueness of solution for the 2D ring system, again thanks to its decoupled nature. Here we solve this system for $n = 1, 2, 3, 4, 5$ and $D = 1$.

2.3.2 Non-gradient Systems

Not all μ's can however be represented as the gradient of a potential. We call the systems belonging to this complementary class, *non-gradient systems*. Analytic solutions for these systems are not known in general.

Noisy Lorenz-63 system

One such example is the famous Lorenz-63 system, first proposed by Edward Lorenz [113] as an oversimplified model for atmospheric convection. This system and its variants like Lorenz-96 have since become staple test problems in the field of data assimilation [28], [178]. We use the standard parameters to define the drift and solve the system for $D = 50$. This exact system also appears as a test case in [32]. The famous butterfly attractor associated with the corresponding ODE is shown in figure 2.1. This problem has a unique solution, for a proof see appendix 2.9.1.

$$\mu = [\alpha(y - x),\ x(\rho - z) - y,\ xy - \beta z]^\top \tag{2.14}$$

$$\alpha = 10\,,\ \beta = \frac{8}{3}\,,\ \rho = 28 \tag{2.15}$$

Noisy Thomas system

Another example of a non-gradient system that we study is one for which the deterministic version was proposed by René Thomas [164]. It is a 3-dimensional system with cyclical symmetry in x, y, z and the corresponding ODE system has a strange attractor which is depicted in figure 2.1. We solve this system for $D = 1$. This problem

also has a unique solution, for a proof see appendix 2.9.1.

$$\mu = [\sin y - bx, \sin z - by, \sin x - by]^\top \tag{2.16}$$

$$b = 0.2 \tag{2.17}$$

Since analytic solutions for non-gradient systems are not known, we stick to $d = 3$ in this case. This is a dimension that can be reliably tackled with Monte Carlo simulations for comparison at a low computation cost. See 2.9.2 for a description of the Monte Carlo algorithm.

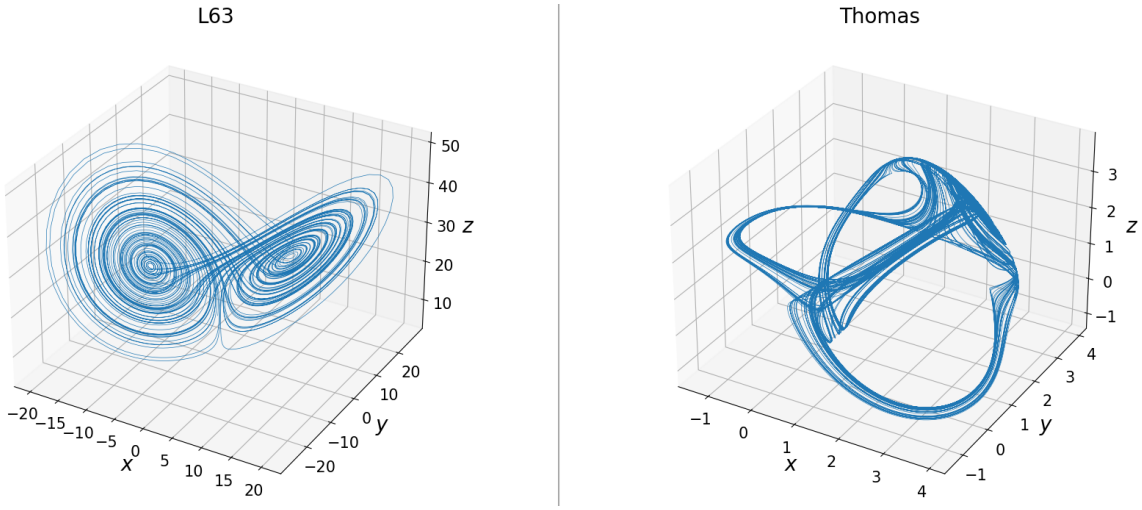

Fig. 2.1 Attractors for non-gradient examples

2.4 Previous works

An extensive amount of work has been done on the topic of numerically solving Fokker-Planck equations. A large amount of these works are based on traditional PDE solving techniques like finite difference [12], [173], [157] and finite element [128], [124] methods. For a comparison of these traditional methods the reader can look at this comparative study [138] by Pitcher et al where the methods have been applied to 2 and 3 dimensional examples.

In recent times efforts have been made to devise methods that are applicable in dimensions higher than 3. Tensor decomposition methods [69], [99] are an important toolkit while dealing with high-dimensional problems and they are proving to be

useful in designing numerical solvers for PDEs [8], [102]. For stationary Fokker-Planck equations Sun and Kumar proposed a tensor decomposition and Chebyshev spectral differentiation based method [163] in 2014. In this method drift functions are approximated with a sum of functions that are separable in spatial variables, an well-established paradigm for solving PDEs. The differential operator for the stationary FPE is then discretized and finally a least sqaures problem is solved to find the final solution. The normalization is enforced via addition of a penalty term in the optimization problem. The high-dimensional integral for the normalization constraint in this method is replaced with products of one dimensional integrals and therefore becomes computable.

In 2017 Chen and Majda proposed another hybrid method [32] that utilizes both kernel and sample based density approximation to solve FPEs that originate from a specific type of SDE referred to as a conditional Gaussian model,

$$\begin{aligned} d\mathbf{u_I} &= [A_0(t,\mathbf{u_I}) + A_1(t,\mathbf{u_I})\mathbf{u_{II}}]\,dt + \Sigma_I(t,\mathbf{u_I})\,dW_I(t) \\ d\mathbf{u_{II}} &= [a_0(t,\mathbf{u_I}) + a_1(t,\mathbf{u_I})\mathbf{u_{II}}]\,dt + \Sigma_{II}(t,\mathbf{u_I})\,dW_{II}(t) \end{aligned} \tag{2.18}$$

This special structure of the SDE allows one to approximate $p(\mathbf{u_{II}}(t))$ as a Gaussian mixture with parameters that satisfy auxiliary SDEs. $p(\mathbf{u_I}(t))$ is approximated with a non-parametric kernel based method. Finally the joint distribution $p(\mathbf{u_I}(t), \mathbf{u_{II}}(t))$ is computed with a hybrid expression. Using this method Chen and Majda computed the solution to a 6 dimensional conceptual model for turbulence. Note that, among our examples only L63 falls under this special structure.

In recent years machine learning has also been applied to solve SFPEs. In 2019 Xu et al solved 2 and 3 dimensional stationary FPEs with deep learning [176]. Their method enforced normalization via a penalty term in the loss function that represented a Monte-Carlo estimate of the solution integrated over $\mathbb{R}^d$. Although simple and effective in lower dimensions, this normalization strategy loses effectiveness in higher dimensions. Zhai et al [180] have proposed a combination of deep learning and Monte-Carlo method to solve stationary FPEs. The normalization constraint here is replaced with a regularizing term in the loss function which tries to make sure the final solution is close to a pre-computed Monte-Carlo solution. This strategy is

more effective than having to approximate high-dimensional integrals and the authors successfully apply their method on Chen and Majda's 6 dimensional example.

2.5 Overview of deep learning

In this section we describe the general process of *learning* a solution to a partial differential equation. The strategy described here will be an integral part of the final algorithm. In what follows next, we see how to see solve a generic PDE independent of time on a bounded domain with a Dirichlet boundary condition in a *physics-informed* manner. The interested reader can see [144], [16], [159] for more discussions. In the next few subsections we keep simplifying our PDE problem until it finally becomes solvable on a computer.

2.5.1 From PDE to optimization problem

In the context of machine learning, *learning* refers to solving an optimization problem. So to solve our PDE with deep learning we first transform it into an optimization problem. Suppose our 2nd order PDE looks like,

$$\begin{aligned} \mathcal{L}f(\mathbf{x}) &= 0, \quad \mathbf{x} \in \Omega\,, \\ f(\mathbf{x}) &= g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega\,, \end{aligned} \tag{2.19}$$

and just like before we are interested in finding a solution in $W^{1,2}_{\text{loc}}(\Omega) \cap C^2(\Omega)$. Instead of trying to solve (2.19) a popular strategy is to try to solve the following problem (see for example [159]),

$$\underset{f \in W^{1,2}_{\text{loc}}(\Omega) \cap C^2(\Omega)}{\arg\inf} \left[\int_\Omega (\mathcal{L}f)^2 + \int_{\partial\Omega} (f-g)^2 \right] \tag{2.20}$$

The choice of function space ensures one-to-one correspondence between the solutions of the PDE and the optimization problem.

2.5.2 From infinite-dimensional search space to finite-dimensional search space

To solve a problem on a machine with finite resources we need to finitize the infinite aspects of the problem. We then solve the finitized problem which preferably approximates the original problem well to get an approximate solution to the orginal problem. For (2.20) our search space $W^{1,2}_{\text{loc}}(\Omega) \cap C^2(\Omega)$ is infinite dimensional which we need to replace with a finite dimensional search space. In order to finitize the dimension of the search space we appeal to universal approximation theorems that say neural networks of even the simplest architectures are dense in continuous functions, see for example Theorem 3.2 in [95] or Proposition 3.7 in [139]. Universal approximation theorems typically allow networks to have either arbitrary depth or arbitrary width in order to achieve density [139], [46]. But the sets of neural networks with arbitrary depth or width are still infinite dimensional and therefore are infeasible to work with. In practice, we fix an architecture $\mathcal{A}$ with a fixed number of layers and trainable parameters and work with the following set instead.

$$S_{\mathcal{A}} \stackrel{\text{def}}{=} \{n^{\mathcal{A}}_{\theta} : \theta \in \mathbb{R}^C\} \tag{2.21}$$

Here $n^{\mathcal{A}}_{\theta}$ is a network with architecture $\mathcal{A}$ with trainable parameters θ and C is the total number of trainable parameters or the size of θ. Since C is fixed, $S_{\mathcal{A}}$ has a one-to-one correspondence with $\mathbb{R}^C$ and therefore is finite-dimensional. Even though we lose the density argument while working with θ of fixed size, in recent times it has been shown that sets like $S_{\mathcal{A}}$ are not closed in $W^{1,2}(\Omega)$ and $n^{\mathcal{A}}_{\theta}$ can be used as a good function approximator, see [117] for a detailed discussion. In the following discussion we suppress the architecture and use $n^{\mathcal{A}}_{\theta}$ and n_{θ} interchangably for notational convenience. After restricting our search space to (2.21), (2.20) becomes,

$$\inf_{\theta \in \mathbb{R}^C} \left[\int_{\Omega} (\mathcal{L} n_{\theta})^2 + \int_{\partial\Omega} (n_{\theta} - g)^2 \right] \tag{2.22}$$

2.5.3 From integrals to sums

The domain Ω in our examples will often be of such a dimension that will make computing the integrals in (2.22) extremely challenging. To deal with this we will replace the integrals in (2.22) with Monte-Carlo sums as follows,

$$\arg\inf_{\theta\in\mathbb{R}^C}\left[\frac{1}{N}\sum_{j=1}^{N}(\mathcal{L}n_\theta(\mathbf{x}_j))^2+\frac{1}{M}\sum_{j=1}^{M}(n_\theta(\mathbf{y}_j)-g(\mathbf{y}_j))^2\right] \tag{2.23}$$

where $\{\mathbf{x}_j\}_{j=1}^N$, $\{\mathbf{y}_j\}_{j=1}^M$ are uniform samples from Ω and $\partial\Omega$ respectively.

2.5.4 Finding the optimal parameters

We simply perform gradient descent with respect to θ to find the optimal network for the problem (2.23). The Monte-Carlo sample sizes should be dictated by the hardware available to the practitioner. In our experiments $N = M = 1000$. In higher dimensions these choices are not enough to capture the original integrals entirely in one go. In that case (2.23) can interpreted as trying to find a network that satisfies the original problem (2.19) at the specified points $\{\mathbf{x}_j\}_{j=1}^N$, $\{\mathbf{y}_j\}_{j=1}^M$, which we can refer to as *collocation points*. But in our experiments we try to the learn the solution on the entire domain as thoroughly as possible and so we resample the domain every few training iterations. So even though we are limited in sample-size by our hardware, we can shift the burden on space or memory to time or number of training iterations and adequately sample the entire domain. This principle of space-time trade-off is ubiquitous in machine learning [25] and comes in many different flavours like mini-batch gradient descent, stochastic gradient descent etc. Even though in this paradigm we are not training our network with typical input-output pairs, our method can be thought of as a variant of the mini-batch gradient descent.

2.5.5 Why deep learning

Deep learning in this context refers to learning an approximate solution to (2.19) with the outlined method with an architecture $\mathcal{A}$ that is *deep* or has many hidden layers. Deep networks are more efficient as approximators than shallow networks in the sense

that they require far fewer number of trainable parameters to achieve the same level of approximation, for a discussion see section 5 of [73] or section 4 of [116]. Now that we have described the general procedure of *deep learning* a solution to a PDE, we will pause briefly to point out some benefits and demerits of this approach. Deep learning has, like any other method some disadvantages.

- Deep learning is slower and less accurate for lower dimensional problems for which standard solvers exist and have been in consistent development for many decades.
- Most modern GPUs are optimized for computation with single-precision or float32 numbers. This is efficient for rendering polygons or other image processing tasks which are the primary reasons GPUs were invented [136] but float32 might be not accurate enough for scientific computing. Although not ideal, this problem will most likely disappear in the future.
- The objective or *loss* function used in a typical problem might not be convex and hence difficult to deal with [103], [9].

But even with these disadvantages, the benefits of deep learning make it a worthwhile tool for solving PDEs.

- Since we don't need to deal with meshes or grids in this method, we can mitigate the curse of dimensionality in memory. It will be clear from our experiments that the size of the network or C does not need to grow exponentially with the dimensions. This method lets one compute the solution at collocation points but if one wants to compute the solution over the entire domain, one needs to sample the entire domain thoroughly which can be done in a sequential manner without requiring more memory as discussed in 2.5.4.
- All derivatives are computed with automatic differentiation and therefore are accurate upto floating point errors. Moreover, finite difference schemes do not satisfy some fundamental properties of differentiation e.g. the product rule [145]. With automatic differentiation one does not have to deal with such problems.
- If one computes the solution over the entire domain, the solution is obtained in a functional form which can be differentiated, integrated etc.

- Other than a method for sampling no modifications are required for accommodating different domains.

2.6 The algorithm

In this section we outline the algorithm for learning zeros of FPOs. But before that we go through the primary challenges and ways to mitigate them.

2.6.1 Unboundedness of the problem domain

We can try the same procedure as outlined in section 2.5 to solve find a non-trivial zero of $\mathcal{L}$. But since computationally we can only deal with a bounded domain, we focus on a compact domain which contains most of the mass of the solution to (2.1). We refer to this domain as the *domain of interest* Ω_I in the following discussion.

2.6.2 Existence of the trivial solution

$\mathcal{L}$, being a linear operator, $\mathcal{L}0 = 0$. Since we want to find a non-trivial zero of $\mathcal{L}$, we would like avoid the learning the zero function during the training of the network. To deal with this problem [180] added a regularization term that required solving (2.1) with Monte-Carlo first. Here we propose a method that does not require a priori knowing an approximate solution. Consider the operator $\mathcal{L}_{\log}$ instead.

$$\mathcal{L}_{\log} f \stackrel{\text{def}}{=} e^{-f} \mathcal{L} e^{f} \tag{2.24}$$

Recalling (2.6) we see that,

$$\mathcal{L}_{\log} f = -\nabla \cdot \mu - \mu \cdot \nabla f + D \left(\|\nabla f\|_2^2 + \Delta f \right) \tag{2.25}$$

Since $\nabla \cdot \mu \not\equiv 0$, any constant function cannot be a zero of $\mathcal{L}_{\log}$. p is a zero of $\mathcal{L}$ iff either $p \equiv 0$ or $\log p$ is a zero of $\mathcal{L}_{\log}$. So we can look for a zero of $\mathcal{L}_{\log}$ to find a non-trivial zero of $\mathcal{L}$.

2.6.3 The steady state algorithm

The procedure outlined in section 2.5 together with the modifications in sections 2.6.1, 2.6.2 immediately yields the following loss function.

$$L_{\log} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\log}(n_\theta(\mathbf{x}_i))^2 \tag{2.26}$$

where $\{\mathbf{x}_i\}_{i=1}^N$ is a uniform sample from Ω_I. Accordingly, the final procedure for finding a non-trivial zero of $\mathcal{L}$ is given in algorithm 4. In the following sections we

Algorithm 4 The steady state algorithm

Sample $\{\mathbf{x}_i\}_{i=1}^N$ from Ω_I, the domain of interest.
Select the desired architecture for n_θ.
Select resampling interval τ.
Select an adaptive learning rate $\delta(k)$ and the number of training iterations E.
for $k = 1, 2 \cdots, E$ **do**
  Compute $\nabla_\theta L_{\log} = \frac{1}{N} \sum_{i=1}^N \nabla_\theta (\mathcal{L}_{\log}(n_\theta(\mathbf{x}_i))^2)$
  where $\mathcal{L}_{\log} f = -\nabla \cdot \mu - \mu \cdot \nabla f + D \left(\|\nabla f\|_2^2 + \Delta f \right)$
  Update $\theta \leftarrow \theta - \delta(k) \nabla_\theta L_{\log}$
  if *k is divisible by* τ **then**
    Resample $\{\mathbf{x}_i\}_{i=1}^N$ from Ω_I
  end
end
$e^{n_\theta(\mathbf{x})}$ is a non-trivial zero of $\mathcal{L}$.
Optional: Approximate $Z \leftarrow \int_{\mathbb{R}^d} e^{n_\theta(\mathbf{x})} \, d\mathbf{x}$. $\frac{1}{Z} e^{n_\theta(\mathbf{x})}$ is the learned, normalized steady state.

describe in detail the network architecture and optimizer used in our experiments. Since the final solution is represented as $e^{n_\theta(\mathbf{x})}$, we have automatically secured positivity of the solution. Note that we can use other stopping criterion such as the norm of the gradient of the loss function falling below a certain threshold but in this study we simply stop the algorithm after a certain number of iterations.

2.6.4 Architecture

We choose the widely used LSTM [158], [169] architecture described below for our experiments. This type of architecture rose to prominence in deep learning because of their ability to deal with the vanishing gradient problem, see section IV of [158], section 2.2 of [169]. A variant of this architecture has also been used to solve PDEs

[159]. This kind of architectures have been shown to be universal approximators [155]. We choose this architecture simply because of how *expressive* they are. By expressivity of an architecture we imply its ability to approximate a wide range of functions and experts have attempted to formalize this notion in different ways in recent times [116], [142], [143]. There are architectures that are probability densities by design i.e. the normalization constraint in (2.1) is automatically satisfied for them, see for example [165], [135]. But our experiments suggest these architectures are not expressive enough to learn solutions to PDEs efficiently since the normalization constraint makes their structure too rigid. Other than the difficulty in implementing the normalization constraint numerically, this is another reason why we choose to focus on learning a non-trivial zero of $\mathcal{L}$ rather than solving (2.1). LSTM networks on the other hand are expressive enough to solve all the problems listed in section 2.3.

Below we define the our architecture in detail. Here $\mathbf{0}_k$ implies a zero vector of dimension k and $\odot$ implies the Hadamard product.

$$i \in \{1, 2, \cdots, L\} \tag{2.27}$$

$$\mathtt{c}_0(\mathbf{x}) \stackrel{\text{def}}{=} \mathbf{0}_m \tag{2.28}$$

$$\mathtt{h}_0(\mathbf{x}) \stackrel{\text{def}}{=} \mathbf{0}_d \tag{2.29}$$

$$\mathtt{f}_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{A}(\mathtt{W}_f^{(i)}\mathbf{x} + \mathtt{U}_f^{(i)} h_{i-1}(\mathbf{x}) + \mathtt{b}_f^{(i)}) \tag{2.30}$$

$$\mathtt{g}_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{A}(\mathtt{W}_g^{(i)}\mathbf{x} + \mathtt{U}_g^{(i)} h_{i-1}(\mathbf{x}) + \mathtt{b}_g^{(i)}) \tag{2.31}$$

$$\mathtt{r}_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{A}(\mathtt{W}_r^{(i)}\mathbf{x} + \mathtt{U}_r^{(i)} h_{i-1}(\mathbf{x}) + \mathtt{b}_r^{(i)}) \tag{2.32}$$

$$\mathtt{s}_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{A}(\mathtt{W}_s^{(i)}\mathbf{x} + \mathtt{U}_s^{(i)} h_{i-1}(\mathbf{x}) + \mathtt{b}_s^{(i)}) \tag{2.33}$$

$$\mathtt{c}_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{f}_i(\mathbf{x}) \odot \mathtt{c}_{i-1}(\mathbf{x}) + \mathtt{g}_i(\mathbf{x}) \odot s_i(\mathbf{x}) \tag{2.34}$$

$$\mathtt{h}_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{r}_i(\mathbf{x}) \odot \mathtt{A}(\mathtt{c}_i(\mathbf{x})) \tag{2.35}$$

$$\mathtt{d}_L(\mathbf{y}) \stackrel{\text{def}}{=} \mathtt{W}^\top \mathbf{y} + \mathtt{b}, \qquad \mathbf{y} \in \mathbb{R}^m \tag{2.36}$$

$$n_\theta^{\text{LSTM}} \stackrel{\text{def}}{=} \mathtt{d}_L \circ \mathtt{h}_L \tag{2.37}$$

Here $\{\mathtt{f}_i, \mathtt{g}_i, \mathtt{r}_i, \mathtt{s}_i, \mathtt{c}_i, \mathtt{h}_i : i = 1, \cdots, L\} \cup \{\mathtt{d}_L\}$ are the hidden layers and

$$\theta = \{\mathtt{W}_f^{(i)}, \mathtt{U}_f^{(i)}, \mathtt{b}_f^{(i)}, \mathtt{W}_g^{(i)}, \mathtt{U}_g^{(i)}, \mathtt{b}_g^{(i)}, \mathtt{W}_r^{(i)}, \mathtt{U}_r^{(i)}, \mathtt{b}_r^{(i)}, \mathtt{W}_s^{(i)}, \mathtt{U}_s^{(i)}, \mathtt{b}_s^{(i)} : i = 1, \cdots, L\} \cup \{\mathtt{W}, \mathtt{b}\} \tag{2.38}$$

is the set of the trainable parameters. The dimensions of these parameters are given below,

$$\mathtt{W}_f^{(i)}, \mathtt{W}_g^{(i)}, \mathtt{W}_r^{(i)}, \mathtt{W}_s^{(i)} \in \mathbb{R}^{m \times d} \tag{2.39}$$

$$\mathtt{U}_f^{(i)}, \mathtt{U}_g^{(i)}, \mathtt{U}_r^{(i)}, \mathtt{U}_s^{(i)} \in \begin{cases} \mathbb{R}^{m \times d}, & \text{if } i = 1 \\ \mathbb{R}^{m \times m}, & \text{otherwise} \end{cases} \tag{2.40}$$

$$\mathtt{b}_f^{(i)}, \mathtt{b}_g^{(i)}, \mathtt{b}_r^{(i)}, \mathtt{b}_s^{(i)} \in \mathbb{R}^m \tag{2.41}$$

$$\mathtt{W} \in \mathbb{R}^m, \mathtt{b} \in \mathbb{R} \tag{2.42}$$

which implies the size of the network or cardinality of θ is

$$C = 4m[d(L+1) + m(L-1)] + 5m + 1 \tag{2.43}$$

Note that (2.43) implies the size of the network grows only linearly with dimension which is an important factor for mitigating the curse of dimensionality. We use elementwise tanh as our activation function,

$$\mathtt{A} = \tanh \tag{2.44}$$

We use $m = 50$ and $L = 3$ for our experiments which implies our network has $6L + 1 = 19$ hidden layers. We use the very popular Xavier or Glorot initialization [63], [45] to initialize θ. With that, the description of our architecture is complete.

2.6.5 Optimization

In our experiments we use the ubiquitous Adam optimizer [97] which is often used in the PDE solving literature [70], [180], [159]. We use a piece-wise linear decaying

learning rate. Below k denotes the training iteration and $\delta(k)$ is the learning rate.

$$\delta(k) = \begin{cases} 5 \times 10^{-3}, & \text{if } k < 1000 \\ 1 \times 10^{-3}, & \text{if } 1000 \leq k < 2000 \\ 5 \times 10^{-4}, & \text{if } 2000 \leq k < 10000 \\ 1 \times 10^{-4}, & \text{if } k \geq 10000 \end{cases} \tag{2.45}$$

We stop training after reaching a certain number of iterations E which varies depending on the problem. In all our experiments we use $N = 1000$ as the sample size and $\tau = 10$ as the resampling interval for algorithm 4.

2.7 Results

We are now ready to describe the results of our experiments. Next few sections parallel the examples in section 2.3 and contain problem-specific details about algorithm 4 e.g. Ω_I, E etc. All computations were done with float32 numbers.

2.7.1 2D ring system

Figure 2.2 shows the learned and true solutions for the 2D ring system. Note that algorithm 4 produces an unnormalized zero of $\mathcal{L}$ but on the left panel the learned solution has been normalized for easier visualization. In this case we use $\Omega_I = [-2, 2]^2$ and $E = 8 \times 10^5$ iterations.

Comparison with Monte Carlo

Since the network was trained with domain resampling every 10 steps and a mini-batch size of $N = 1000$, during the entire training procedure 8×10^7 points were sampled from the domain. We compute the steady state with Monte Carlo with 8×10^7 particles to compare errors produced by both methods. Here the SDE trajectories were generated till time 10 with time-steps of 0.01. Since in this case we know the analytic solution we can compute and compare absolute errors. As we can see in

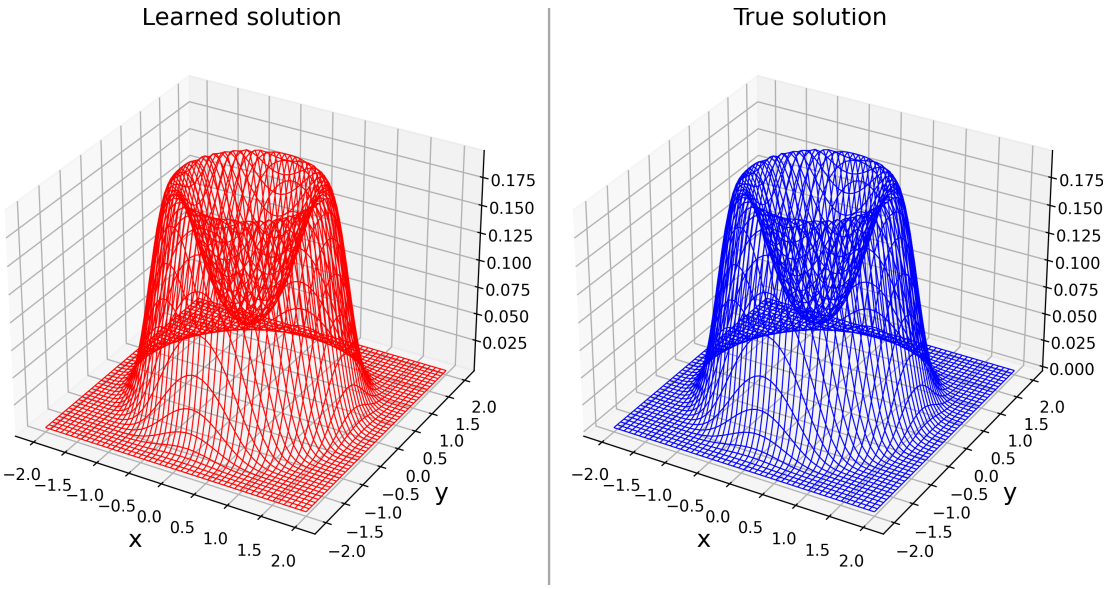

Fig. 2.2 Solution for the 2D ring system

figure 2.3, for the same number of overall sampled points, Monte Carlo error is an order of magnitude larger than deep learning error.

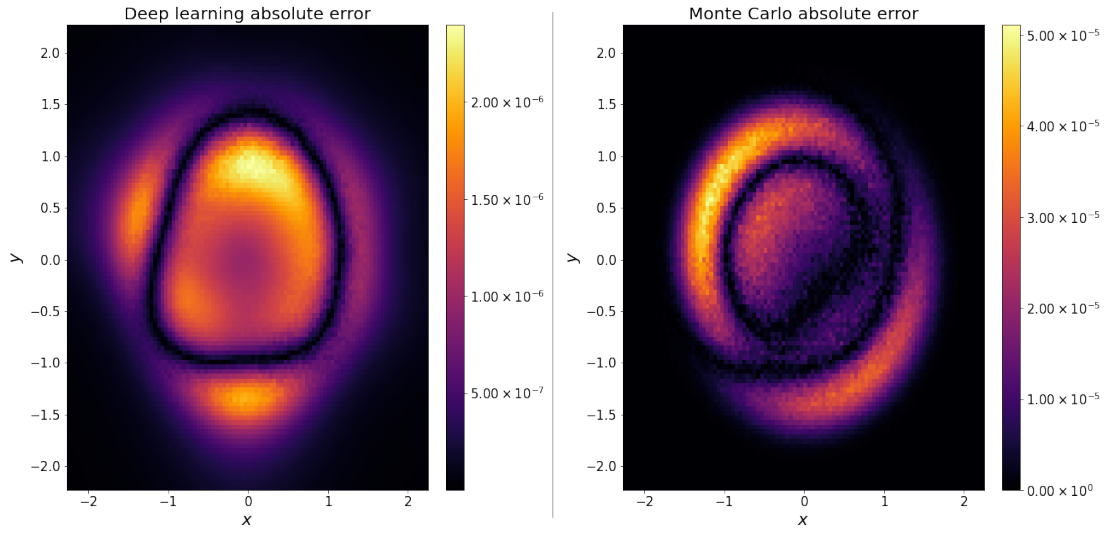

Fig. 2.3 Comparison of absolute errors for deep learning and Monte Carlo solutions for the 2D ring system

2.7.2 2nD ring system

Although we solve this system for $n = 1, 2, 3, 4, 5$, in this section we only produce the results for $n = 5$ or $d = 10$ to avoid repetition. Figure 2.4 shows the solutions for

the 10D ring system for $\Omega_I = [-2,2]^{10}$ and $E = 4.6 \times 10^6$. In order to visualize the solution we focus on the quantity $p(0,0,0,0,x_4,x_5,0,0,0,0)$. For a visual comparison with the true solution normalization is desirable. But rather than trying to compute a 10-dimensional integral which is a non-trivial problem in itself we can normalize $p(0,0,0,0,x_4,x_5,0,0,0,0)$ which is much easier to do and due to the decoupled nature of this problem we can expect an identical result as in figure 2.2 which is what we see in figure 2.4. In both of the panels the solutions have been normalized in a way such that,

$$\int_{\mathbb{R}} \int_{\mathbb{R}} p(0,0,0,0,x_4,x_5,0,0,0,0)\, dx_4\, dx_5 = 1$$

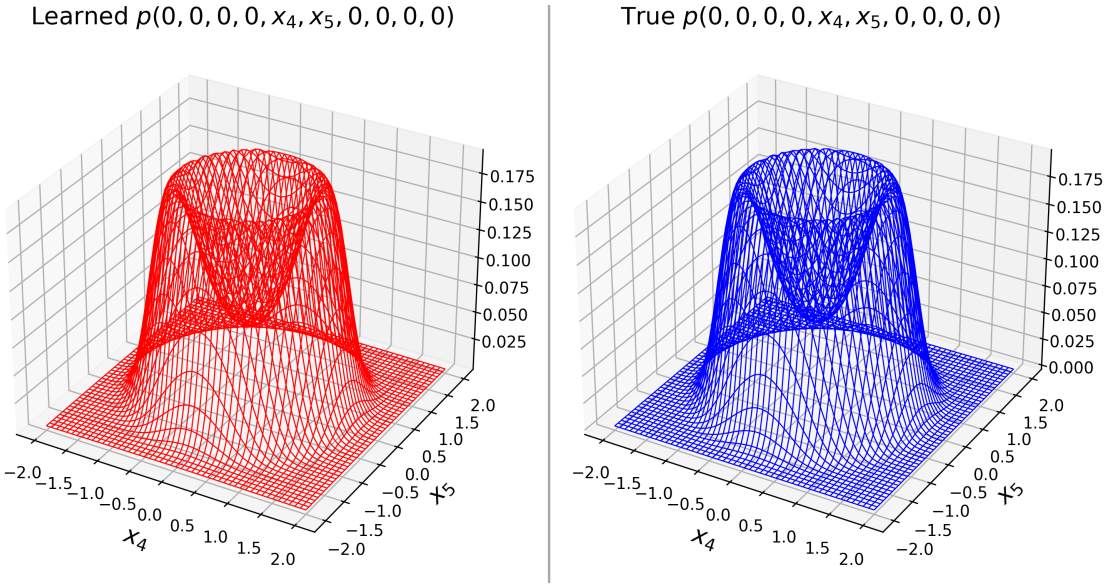

Fig. 2.4 Solutions for the 10D ring system. Both solutions have been normalized such that $\int_{\mathbb{R}} \int_{\mathbb{R}} p(0,0,0,0,x_4,x_5,0,0,0,0)\, dx_4\, dx_5 = 1$

The error in the learned solution can be seen in figure 2.5.

2.7.3 Noisy Lorenz-63 system

Figure 2.6 shows the results for the L63 system for $\Omega_I = [-30,30] \times [-40,40] \times [0,70]$ and $E = 10^6$. For ease of visualization the solutions have been normalized and in each row one of the dimensions has been integrated over the relevant interval to produce 2D marginals. In order to integrate out one dimension we use a composite Gauss-Legendre quadrature rule. We subdivide the relevant interval into 240 subintervals and use 10-point Gauss-Legendre rule to compute the integral over every subinterval. Note that since n_θ is a smooth function, our integrand is always a smooth function.

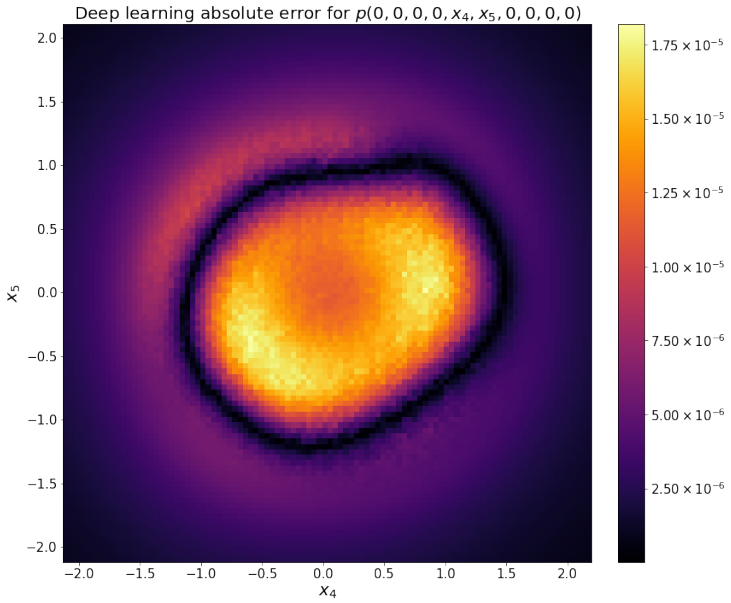

Fig. 2.5 Absolute error in the learned solution for the 10D ring system

The largest possible subinterval is of length $\frac{40-(-40)}{240} = \frac{1}{3}$ so assuming absolute value of the 20-th derivative of the integrand is upper-bounded by M everywhere, the integration error on each subinterval is upper-bounded by $\frac{2M}{20!}\left(\frac{1}{6}\right)^{20} \leq 2.25M \times 10^{-34}$, see appendix 2.9.3 for more details on this estimate. To produce the Monte Carlo solution, SDE trajectories were generated till time 10 with time-steps of 10^{-2}. Since Monte Carlo produces lower-accuracy solutions even in lower dimensions as we saw in section 2.7.1 and an analytic solution is unavailable in this case, we refrain from producing error plots.

2.7.4 Noisy Thomas system

Figure 2.7 shows the results for the Thomas system for $\Omega_I = [-10, 10]^3$ and $E = 4 \times 10^5$. Due to the inherent symmetry of this problem it suffices to compute only the 2D marginal $p(x, y)$. To integrate out the z dimension we use 8-point composite Gauss-Legendre quadrature rule with 165 subintervals. Assuming absolute value of the 16-th derivative of the integrand is upper-bounded by M everywhere, the integration error on each subinterval is upper-bounded by $\frac{2M}{16!}\left(\frac{10}{165}\right)^{16} \leq 3.17M \times 10^{-33}$, see appendix 2.9.3 for more details on this error estimate. To produce the Monte Carlo solution, SDE trajectories were generated till time 10 with time-steps of 10^{-2}. Even though we have solved a lower dimensional problem here, Thomas system turns out

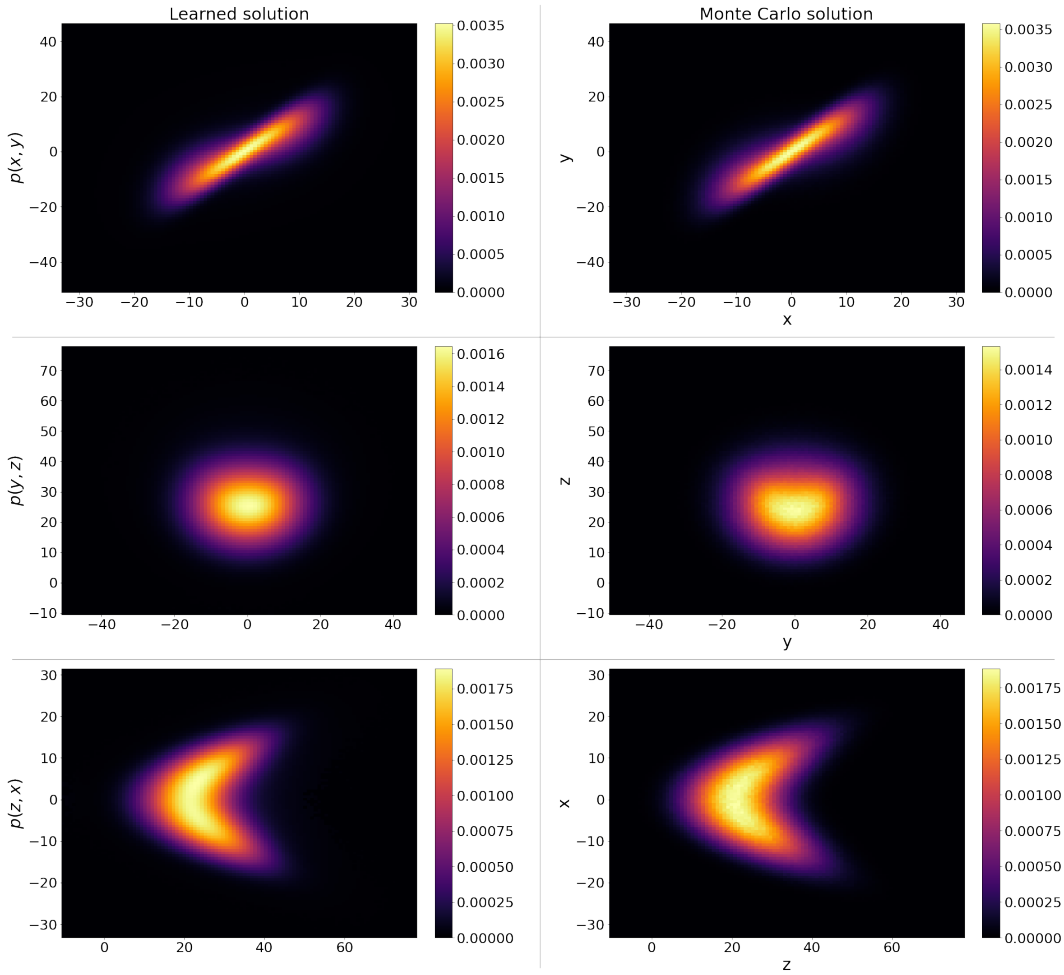

Fig. 2.6 Solutions for the noisy Lorenz-63 system

to be the *easiest* i.e. algorithm 4 converges faster for this system compared to the other ones as we will see in the next section.

2.7.5 Dimension dependence

In this section we explore the dimension dependence of algorithm 4. In the left panel of figure 2.8 we have plotted the loss given by (2.26) against training iterations for all the of the systems above in a semi-log manner starting from iteration 100. We often encounter spikes in the loss curve for the following reasons

- the loss curves are single realizations of algorithm 4 instead of being an average

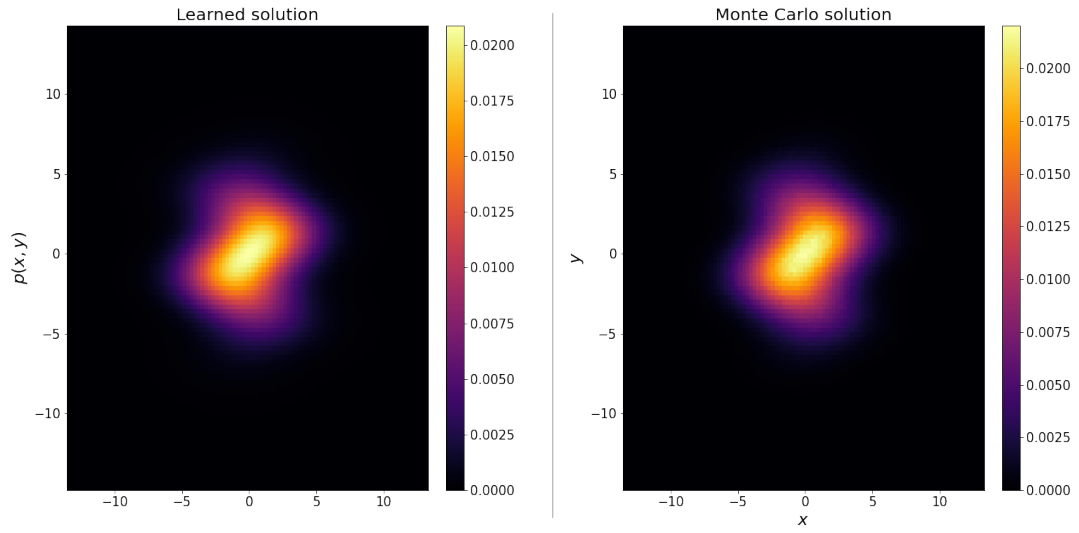

Fig. 2.7 Solutions for the noisy Thomas system

- we resample the domain every 10 iterations and if the new points belong to a previously unexplored region in Ω_I, the loss might increase.

But the general trend of loss diminishing with iterations is true for every system. We also see that loss is system-dependent and the *hardness* of these problems or how quickly algorithm 4 converges depends on the nature of μ as much as the dimension. This is easily seen by noting that the two 3D systems sandwich the 2D and the 4D ring systems in the left panel of figure 2.8. The loss for Thomas system drops very quickly compared to the rest of the systems due to the simplicity and global Lipschitzness of the corresponding drift function. We also see from the right panel of figure 2.8 that time taken per training iteration grows near-linearly with dimension. Since it is hard to estimate the amount of iterations to run before the loss drops below a pre-determined level, we refrain from plotting the total runtime of algorithm 4 against dimension. But it's interesting to note that the number of total training iterations E varies from 4×10^5 to 4.6×10^6 across all the problems. Since the data shown in the right panel of figure 2.8 is very much hardware dependent, at this point we disclose that all of the experiments were done using the cloud service provided by Google Colab. This service automatically assigns runtimes to different hardware depending on availability at the time of computation which might explain why the 8D and 10D ring systems take nearly the same amount of time per iteration in figure 2.8.

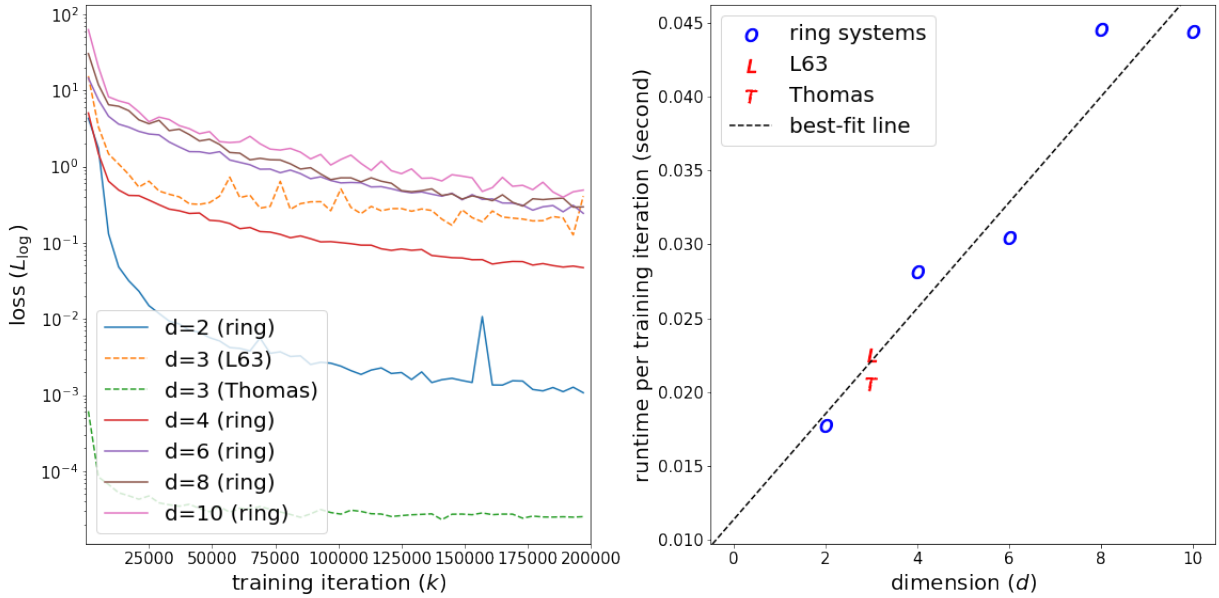

Fig. 2.8 Left panel: Loss vs training iteration starting from iteration 100. Right panel: time taken per training iteration vs dimension.

2.7.6 Comparison of loss and distance from truth

In this section we explore the relationship between the loss given by (2.26) and the distance from truth. In spite of being structurally completely different, both are measures of goodness for a computed solution. In most cases we only have access to the loss and therefore it is an important question if going a decreasing loss implies getting closer to the truth for algorithm 4. We define the distance of the learned zero from the true solution as follows,

$$\|\phi\|_* \stackrel{\text{def}}{=} \sup_{\mathbf{x}\in\Omega_I} |c\phi(\mathbf{x}) - p^{\text{true}}(\mathbf{x})|, \qquad c\int_{\mathbb{R}^d}\phi = 1 \tag{2.46}$$

where p^{true} is the true solution to (2.1). (2.46) is not easy to compute in arbitrary dimensions but can be computed for the 2D ring system without too much effort since p^{true} is known and the problem is low-dimensional. Figure 2.9 shows the results for the 2D ring system. The right panel of figure 2.9 shows that loss and distance from truth are strongly correlated for algorithm 4. Moreover, asymptotically for small values of the loss function they are linearly related with a Pearson correlation coefficient $R = 0.98$ as can be seen from the inset in the right panel which depicts the data from training iteration 10000 to 50000. The best-fit line is also shown in the inset. On the left panel we see that the distance from truth monotonically decreases with training

iteration and is extremely well approximated by a curve of the form $a_0 e^{-a_1 k} + a_2$. Both panels contain data from training iteration 5000 to 50000. We omit the first few iterations to filter out the effects of the random initialization of the trainable parameters. Figure 2.9 serves as a good justification for algorithm 4 since it shows that minimizing the loss is akin to getting closer to a true non-trivial zero of $\mathcal{L}$.

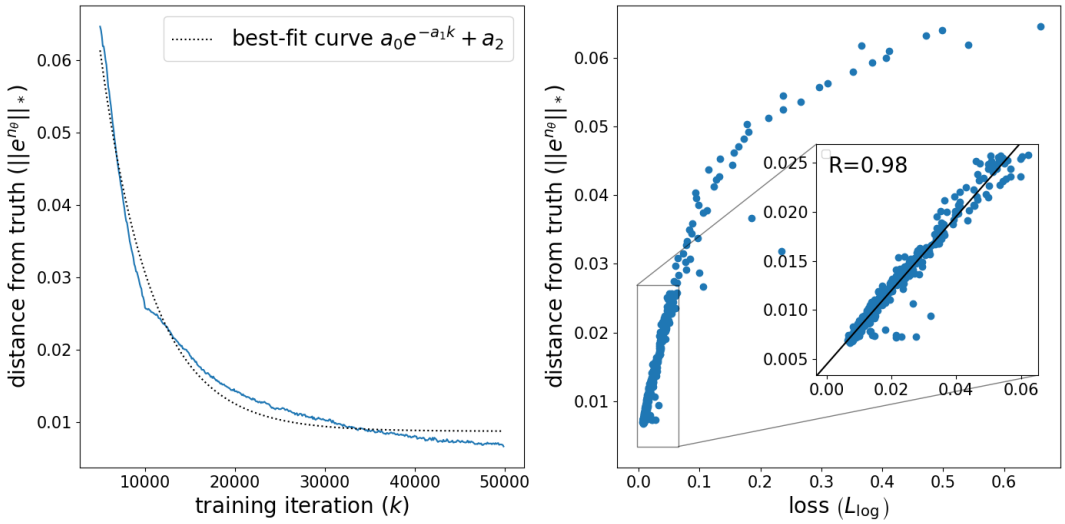

Fig. 2.9 Left panel: Distance from truth vs training iteration every 100 iterations, starting from iteration 5000 and ending at iteration 50000 for the 2D ring system. Right panel: Scatter plot for loss vs distance from truth for the 2D ring system. The inset shows that asymptotically loss and the distance from the truth are linearly related. The inset depicts the data from training iteration 10000 to 50000.

2.8 Summary and Future Work

In this work we devise a deep learning algorithm for finding non-trivial zeros of $\mathcal{L}$ when the corresponding drift is non-solenoidal. We can summarize our results as follows.

1. Our choice of architecture is capable of learning zeros of Fokker-Planck operators across many different problems while scaling only linearly with dimension.

2. Time taken per training iteration grows near-linearly with problem dimension.

3. Apart from being able to produce solutions in a functional form which Monte Carlo is incapable of, for the same overall sample-size algorithm 4 produces more accurate solutions compared to Monte Carlo.

4. How quickly algorithm 4 converges to a zero depends as much on the dimension as it does on the nature of the problem or the structure of μ.

5. By minimizing the loss we also get closer to a true non-trivial zero of the Fokker-Planck operator which justifies algorithm 4. The loss and the distance from a true zero of the Fokker-Planck operator, even though structurally completely different, are strongly correlated. Moreover, they can be asymptotically linearly related for small values of the loss function.

In a sequel we will show how we can solve time-dependent FPEs by using the zeros learned by algorithm 4. The landscape of the loss defined in (4) poses many interesting geometric questions. For example, in case the nullspace of $\mathcal{L}$ is 1-dimensional do all the minima of $L_{\log}$ lie on a connected manifold of dimension 1 or are they disconnected from each other? Such questions provide possible avenues for future research.

2.9 Appendix

2.9.1 Existence and uniqueness of solutions to example problems

In this section we prove that the example problems used here have a unique weak solution in $W^{1,2}_{\text{loc}}(\mathbb{R}^d)$. We employ the method of Lyapunov function as described in [76] to arrive at existence and uniqueness. First we begin with the prerequisites for this approach.

Lyapunov functions

Definition 2.9.1 *Let $U \in C(\mathcal{U})$ be a non-negative function and denote $\rho_M = \sup_{\mathbf{x}\in\mathcal{U}} U(\mathbf{x})$, called the essential upper bound of U. U is said to be a compact function in U if*

$$i)\ U(\mathbf{x}) < \rho_M, \quad \mathbf{x} \in U \tag{2.47}$$

and

$$ii)\ \lim_{\mathbf{x}\to\partial U} U(\mathbf{x}) = \rho_M \tag{2.48}$$

This definition of a compact function appears as definition 2.2 in [76].

Proposition 2.9.2 *An unbounded, non-negative function $U \in C(\mathbb{R}^d)$ is compact iff*

$$\lim_{\|x\|_2 \to +\infty} U(\mathbf{x}) = +\infty \tag{2.49}$$

This proposition appears as Proposition 2.1 in [76].

Definition 2.9.3 *Let U be a compact function in $C^2(\mathcal{U})$ with essential upper bound ρ_M. U is called a Lyapunov function in $\mathcal{U}$ with respect to $\mathcal{L}^*$ is $\exists\, \rho_m \in (0, \rho_M)$ and a constant $\gamma > 0$ such that*

$$\mathcal{L}^* U(\mathbf{x}) \leq -\gamma, \qquad \forall \mathbf{x} \in \mathcal{U} \setminus \overline{\{\mathbf{x} \in \mathcal{U} : U(\mathbf{x}) < \rho_m\}} \tag{2.50}$$

where $\mathcal{L}^*$ is the adjoint Fokker-Planck operator given by

$$\mathcal{L}^* f = \mu \cdot \nabla f + D \odot \nabla^2 f \tag{2.51}$$

This definition appears as definition 2.4 in [76]. Now we are ready to state the main theorem that will help us prove uniqueness for our example problems.

Theorem 2.9.4 *If the components of μ are in $L^2_{\text{loc}}(\mathcal{U})$ and there exists a Lyapunov function with respect to $\mathcal{L}^*$ in $C^2(\mathcal{U})$ then (2.1) has a positive weak solution in the space $W^{1,2}_{\text{loc}}(\mathcal{U})$. If, in addition, the Lyapunov function is unbounded, the solution is unique in $\mathcal{U}$.*

This theorem appears as Theorem A in [76]. Since the components of μ are locally integrable for our example problems, all we need to do is find an unbounded Lyapunov function U for proving existence and uniqueness in $W^{1,2}_{\text{loc}}(\mathbb{R}^d)$.

Existence and uniqueness of solution for 2D ring system

Setting

$$\mathcal{U} = \mathbb{R}^2 \tag{2.52}$$

$$U(x,y) = x^2 + y^2 \tag{2.53}$$

$$\rho_m = \frac{1}{2} + \sqrt{D+1} \tag{2.54}$$

$$\gamma = 4D + 6 \tag{2.55}$$

$$\tag{2.56}$$

we see that,

$$\mathcal{L}^* U + \gamma = -8\left(x^2 + y^2 - \frac{1}{2}\right)^2 + 8(D+1) \tag{2.57}$$

and,

$$\mathcal{U} \setminus \overline{\{\mathbf{x} \in \mathcal{U} : U(\mathbf{x}) < \rho_m\}} = \left\{(x,y) \in \mathbb{R} : x^2 + y^2 > \rho_m\right\} \tag{2.58}$$

In $\{(x,y) \in \mathbb{R} : x^2 + y^2 > \rho_m\}$,

$$\mathcal{L}^* U + \gamma \leq 0 \tag{2.59}$$

and therefore U is an unbounded Lyapunov function for the 2D ring system which guarantees uniqueness of solution (2.11).

Existence and uniqueness of solution for L63 system

Setting,

$$U(x,y,z) = \rho x^2 + \alpha y^2 + \alpha(z - 2\rho)^2 \tag{2.60}$$

we see that

$$\mathcal{L}^*U = -2\alpha\rho x^2 - 2\alpha y^2 - 2\alpha\beta z^2 + 4\alpha\beta\rho z + 2D(2\alpha+\rho) \tag{2.61}$$
$$= -2\alpha\rho x^2 - 2\alpha y^2 - \alpha\beta z^2 - \alpha\beta(z-2\rho)^2 + 4\alpha\beta\rho^2 + 2D(2\alpha+\rho) \tag{2.62}$$
$$\leq -\rho x^2 - \alpha y^2 - \alpha(z-2\rho)^2 + 4\alpha\beta\rho^2 + 2D(2\alpha+\rho) \tag{2.63}$$
$$= -U(x,y,z) + 4\alpha\beta\rho^2 + 2D(2\alpha+\rho) \tag{2.64}$$

(2.63) is a consequence of $\alpha, \beta, \rho > 1$. Now setting,

$$\gamma = 1, \tag{2.65}$$
$$\rho_m = 4\alpha\beta\rho^2 + 2D(2\alpha+\rho) + 1 \tag{2.66}$$

we see that in $\{U > \rho_m\}$,

$$\mathcal{L}^*U + \gamma \leq 0 \tag{2.67}$$

So U is an unbounded Lyapunov function for this system and we have a unique solution.

Existence and uniqueness of solution for Thomas system

Setting,

$$U(x,y,z) = x^2 + y^2 + z^2 \tag{2.68}$$

we see that

$$\mathcal{L}^*U = x\sin y + y\sin z + z\sin x - b(x^2+y^2+z^2) + 6D \tag{2.69}$$
$$\leq \sqrt{3U} - bU + 6D \tag{2.70}$$
$$= -b\left(\sqrt{U} - \frac{\sqrt{3}}{2b}\right)^2 + \frac{3}{4b} + 6D \tag{2.71}$$

(2.70) follows from Cauchy Schwarz inequality. Setting,

$$\gamma = \frac{1}{4b}, \tag{2.72}$$

$$\rho_m = \left(\frac{\sqrt{3}}{2b} + \frac{\sqrt{1+6bD}}{b} \right)^2 \tag{2.73}$$

we see that in $\{U > \rho_m\}$,

$$\mathcal{L}^* U + \gamma \leq 0 \tag{2.74}$$

So U is an unbounded Lyapunov function for this system and we have a unique solution.

2.9.2 Monte Carlo steady state algorithm

The time-dependent FPE given by

$$\begin{aligned} &\frac{\partial p(t,\mathbf{x})}{\partial t} = \mathcal{L} p(t,\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^d,\ t \geq 0 \\ &p(0,\mathbf{x}) = p_0(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^d \\ &\int_{\mathbb{R}^d} p(t,\mathbf{x})\, d\mathbf{x} = 1, \qquad \forall\ t \geq 0 \end{aligned} \tag{2.75}$$

gives us the probability density of the random process X_t which is governed by the SDE,

$$\begin{aligned} dX_t &= \mu\, dt + \sigma\, dW_t \\ X_0 &\sim p_0 \end{aligned} \tag{2.76}$$

where $\{W_t\}$ is the standard Wiener process, see for example chapters 4, 5 of [60]. We can evolve (4) up to sufficiently long time using Euler-Maruyama method [98] to approximate the steady state solution of (2.75) or the solution of (2.1) as follows. Here $\mathcal{N}$ denotes the multivariate normal distribution. Note that in case of a unique solution of (2.1), many choices of p_0 can lead to the stationary solution. In all our examples, it suffices to choose p_0 to be the standard d-dimensional normal distribution.

Algorithm 5 Monte Carlo steady state algorithm

Sample $\{X_0^{(i)}\}_{i=1}^N \sim p_0$.
Set the time-step h.
Set the number of steps S.
for $k = 1, 2 \cdots, S$ **do**
  Sample $w_k^i \sim \mathcal{N}(\mathbf{0}_d, hI_d) \ \ \forall\ i$
  $X_k^{(i)} \leftarrow X_{k-1}^{(i)} + \mu\left(X_{k-1}^{(i)}\right) h + \sigma w_k^i \ \ \forall\ i$
end
Subdivide the domain of interest Ω_I into d-dimensional boxes.
Count the number of $X_S^{(i)}$ that are in a box to estimate the stationary density at the center of the box.

2.9.3 Integration error for n-point Gauss-Legendre rule

Suppose we are trying to integrate a smooth function $f(x)$ over $\left[a - \frac{h}{2}, a + \frac{h}{2}\right]$ with n-point Gauss-Legendre rule where $h \in (0, 1]$. Let us denote $I[f]$ to be the Gauss-Legendre approximation of $\int_{a-\frac{h}{2}}^{a+\frac{h}{2}} f(x)\, dx$. Recalling that n-point Gauss-Legendre gives us exact integrals for polynomial of degree $\leq 2n-1$ and using the Lagrange form of Taylor remainder we see that,

$$\left| I[f] - \int_{a-\frac{h}{2}}^{a+\frac{h}{2}} f(x)\, dx \right| \leq MI\left[\frac{(x-a)^{2n}}{(2n)!}\right] + M\int_{a-\frac{h}{2}}^{a+\frac{h}{2}} \frac{(x-a)^{2n}}{(2n)!}\, dx \tag{2.77}$$

where $|f^{(2n)}(x)| \leq M \ \ \forall \ \ x \in \left[a - \frac{h}{2}, a + \frac{h}{2}\right]$. To bound the first term on the RHS of (2.77) we can use the fact that if

$$I[f] = \sum_{i=1}^{n} w_i f(x_i) \tag{2.78}$$

then,

$$I[1] = \int_{a-\frac{h}{2}}^{a+\frac{h}{2}} 1\, dx = h \tag{2.79}$$

$$\implies \sum_{i=1}^{n} w_i = h \leq 1 \tag{2.80}$$

$$\implies I\left[\frac{(x-a)^{2n}}{(2n)!}\right] \leq \frac{1}{(2n)!}\left(\frac{h}{2}\right)^{2n} \tag{2.81}$$

Therefore,

$$\left| I[f] - \int_{a-\frac{h}{2}}^{a+\frac{h}{2}} f(x)\,dx \right| \leq \frac{M}{(2n)!}\left(\frac{h}{2}\right)^{2n} + \frac{2M}{(2n+1)!}\left(\frac{h}{2}\right)^{2n+1} \leq \frac{2M}{(2n)!}\left(\frac{h}{2}\right)^{2n} \tag{2.82}$$

Chapter 3

In the last chapter we computed unnormalized solutions to the stationary Fokker-Planck equation using deep learning. In this chapter we are interested in the time-dependent Fokker-Planck equation. Even though deep learning seems to effective for elliptic partial differential equations, the same is not true for parabolic partial differential equations. In this chapter first we show that physics-informed neural networks are not suitable for a large class of parabolic partial differential equations including the Fokker-Planck equation. Then we devise an algorithm to compute solutions of the Fokker-Planck equation using the zeros of Fokker-Planck operator and the Feynman-Kac formula. The resulting algorithm is mesh-free, highly parallelizable and able to compute solutions pointwise, thus mitigating the curse of dimensionality in a practical sense. We analyze various nuances of this algorithm that are determined by the drift term in the Fokker-Planck equation. We work with problems ranging in dimensions from 2 to 10. We demonstrate that this algorithm requires orders of magnitude fewer trajectories for each point in space when compared to Monte-Carlo. We also prove that under suitable conditions the error caused by letting some trajectories associated with the Feynman-Kac expectation, escape our domain of knowledge is proportional to the fraction of trajectories that escape. As an application we calculate the one step filtering density for a noisy dynamical system.

Chapter 3

Solving Fokker-Planck equations

3.1 Introduction

From the motion of a particle suspended in a fluid [92], enzyme kinetics [2] to dynamics of a stock price [93], [49] evolving systems in the real worlds are often modelled as systems of ordinary differential equations propagating under the influence of additive noise. These models known as stochastic differential equations (SDE) [131], [60], [161] are directly linked to Fokker-Planck equations (FPE) [149] or Kolmogorov forward equations that describe the evolution of probability density of the state vector. In the prequel we developed a deep learning algorithm to compute non-trivial zeros of high-dimensional Fokker-Planck operators in a mesh-free manner. In this chapter we will devise an algorithm to compute solutions of high-dimensional time-dependent FPEs in a mesh-free manner. We will begin by noting an algorithm similar to the one used in chapter 2 to solve stationary FPEs (SFPE) fails for time-dependent FPEs. A widely adopted strategy for solving high-dimensional PDEs is to appeal to Feynman-Kac type formulae [47], [84], since they allow pointwise calculation of solutions without requiring a mesh thus mitigating at least one aspect of the curse of dimensionality. For example, Kakutani's solution of Dirichlet problem for the Laplace operator [88], [89], Muller's walk-on-spheres method for Dirichlet problems [127] and an analogous method called walk-on-stars for Neumann problems [154], multi-level Picard iteration method for solving semilinear heat equations [77] are all based on Feynman-Kac type formulae. In recent times deep learning methods have been combined with the Feynman-Kac formula to solve high-dimensional PDEs [70], [16]. Even though

FPEs are semilinear, parabolic PDEs whose solutions are probability densities and Deep-BSDE method proposed in [70] deals with semilinear, parabolic PDEs, generic FPEs pose many challenges that make them unapproachable for deep-BSDE. Non-Lipschitzness of drift functions leading to blow-up of SDE trajectories [36], [111] and unboundedness of the divergence of drift functions causing FPEs to dissatisfy one of the requirements for the Feynman-Kac formula are foremost amongst these challenges. In this chapter we apply the Feynman-Kac formula on an auxiliary equation and combine the solution with the zero of the Fokker-Planck operator obtained through the method described in chapter 2 to produce the normalized solution to the time-dependent Fokker-Planck equation. We will apply our method for problem dimensions ranging from 2 to 10 to verify its effectiveness in high dimensions.

As we have noted, real world systems are often modelled as SDEs and we often observe such systems partially due to limited resources and are tasked with determining the distribution of the state vector at a certain time given all the observations up to that time. This is known as the filtering problem in the field of data assimilation and is useful for a variety of topics - global positioning system, target tracking, monitoring infectious diseases, to name a few [153]. FPEs are naturally connected to data assimilation when the underlying dynamics is stochastic. We will show how this method can be used to to calculate the one-step filtering density in the nonlinear filtering problem. To that end we will focus on systems with attractors since such systems are often used as important test cases in the field of data assimilation [28].

3.2 Problem Statement

3.2.1 Time-dependent FPEs

Consider the Itô SDE with coefficients $\mu \in C^1(\mathbb{R}^d; \mathbb{R}^d)$ and $\sigma \in C(\mathbb{R}^d; \mathbb{R}^{d\times l})$ that are independent of time,

$$\begin{aligned} dX_t &= \mu(X_t)\, dt + \sigma(X_t)\, dW_t \\ X_0 &\sim p_0 \end{aligned} \tag{3.1}$$

such that $D = \frac{1}{2}\sigma\sigma^\top$ is a positive definite matrix for all inputs. Such an SDE is directly related to the Fokker-Planck equation describing the evolution of the probability

density of X_t [149], [18], [24],

$$\begin{aligned}
&\frac{\partial p}{\partial t} = \mathcal{L}p \stackrel{\text{def}}{=} -\nabla \cdot (\mu p) + \text{tr}(D \odot \nabla^2 p) = 0, \qquad \mathbf{x} \in \mathbb{R}^d,\ t \in (0, T] \\
&p(0, \mathbf{x}) = p_0(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^d \\
&\int_{\mathbb{R}^d} p(t, \mathbf{x})\, d\mathbf{x} = 1, \qquad t \in [0, T]
\end{aligned} \tag{3.2}$$

where $\odot$ is the Hadamard product and ∇^2 denotes Hessian. The operator $\mathcal{L}$ is known as the Fokker-Planck operator (FPO). Our goal is to solve (3.2) in a mesh-free way in high-dimensions. In our examples we stick to $\sigma \equiv cI_d$ for some $c > 0$ which lets us abuse notation and use σ and D as scalar quantities, note that however our final method can be easily applied to problems where σ is indeed a function of space or non-diagonal with minor modifications. With this simplification (3.2) becomes,

$$\frac{\partial p}{\partial t} = -\nabla \cdot (\mu p) + D\Delta p = 0 \tag{3.3}$$

where Δ is the Laplacian operator. An extensive amount of work has been done to find numerical solutions to FPEs over the years, for a brief overview the reader can see section 2.4 of chapter 2.

3.2.2 One step filtering problem

Suppose we partially observe X_t at discrete times $t = g, 2g, 3g, \cdots$ with observation gap g. These observations are given by,

$$y_k = Hx_k + \eta_k, \qquad k = 1, 2, \cdots \tag{3.4}$$

where $x_k = X_{kg}$, $H : \mathbb{R}^d \to \mathbb{R}^q$ is a projection matrix and $\eta_k \sim \mathcal{N}(0_q, \sigma_o^2 I_q)$ are iid Gaussian errors in observation. Given all the observations up to time $t = gk$, the filtering problem asks us to compute the distribution of the state vector at that time i.e. $p(x_k|y_{1:k})$. Here, as a simple application, we will calculate the one-step filtering density $p(x_1|y_1)$.

3.3 Examples

All the examples here parallel the examples that appear in the last chapter where we call a system a *gradient system* if the corresponding μ can be written as the gradient of some potential function,

$$\mu = -\nabla V \tag{3.5}$$

and otherwise we call it *non-gradient system*. Gradient systems provided important test cases in the last chapter while solving stationary FPEs since their steady states are known in analytical form. But for time-dependent FPEs analytical solutions are not known in general even if μ satisfies (3.5).

3.3.1 Gradient systems

We use the following gradient systems to verify the effectiveness of our algorithm in high-dimensions. A corresponding non-trivial zero of the Fokker-Planck operator was calculated in the last chapter for each case.

2D ring system

For $V = (x^2 + y^2 - 1)^2$ and $\mu = -\nabla V$ we get the following FPE,

$$\frac{\partial p}{\partial t} = 4(x^2 + y^2 - 1)\left(x\frac{\partial p}{\partial x} + y\frac{\partial p}{\partial y}\right) + 8(2x^2 + 2y^2 - 1)p + D\Delta p \tag{3.6}$$

The corresponding ODE system has the unit circle as a global attractor. We solve this system for $D = 1$. As the initial condition we use an equal Gaussian mixture with the components having centers at $\left(-\frac{1}{2}, -\frac{1}{2}\right)$ and $\left(\frac{1}{2}, \frac{1}{2}\right)$ with covariance matrix $\frac{1}{4}I_2$.

$$p_0(x,y) = \frac{1}{\pi}\exp\left(-2\left(x + \frac{1}{2}\right)^2 - 2\left(y + \frac{1}{2}\right)^2\right) + \frac{1}{\pi}\exp\left(-2\left(x - \frac{1}{2}\right)^2 - 2\left(y - \frac{1}{2}\right)^2\right) \tag{3.7}$$

This system has a unique solution, for a proof see appendix 3.7.1.

2nD ring system

We can daisy-chain the previous system to build decoupled systems in higher dimensions. In this case the potential is given by

$$V(\mathbf{x}) = \sum_{i=0}^{\frac{d}{2}-1} (x_{2i}^2 + x_{2i+1}^2 - 1)^2, \qquad d = 2n \tag{3.8}$$

We use the following initial condition which can be obtained by daisy-chaining the initial condition in (3.7).

$$\begin{aligned} p_0(\mathbf{x}) = \prod_{i=0}^{\frac{d}{2}-1} \Bigg[\frac{1}{\pi} \exp\left(-2\left(x_{2i} + \frac{1}{2} \right)^2 - 2\left(x_{2i+1} + \frac{1}{2} \right)^2 \right) + \\ \frac{1}{\pi} \exp\left(-2\left(x_{2i} - \frac{1}{2} \right)^2 - 2\left(x_{2i+1} - \frac{1}{2} \right)^2 \right) \Bigg] \end{aligned} \tag{3.9}$$

Since our algorithm does not differentiate between coupled and decoupled systems, this example serves as a great high-dimensional test case. Although the analytical solution for this problem is not known, the decoupled nature of this problem implies that we can compare our solution to the 2nD ring system with the solution to the 2D ring system which can be easily computed with other methods (e.g. Monte Carlo) that are not efficient in higher dimensions. Here we solve this system for $n = 1, 2, 3, 4, 5$ and $D = 1$. This system has a unique solution which follows directly from the fact that the 2D ring system has a unique solution.

3.3.2 Non-gradient Systems

Even though the analytic solution for the gradient case is not known for time-dependent FPEs, the gradient structure of the drift makes solving time-dependent FPEs much *easier* as we will see in section 3.4.4. Non-gradient systems on the other hand pose a much harder challenge due to the blow-up of auxiliary SDEs, see section 3.4.4 for more details. Here we deal with the following non-gradient systems. A corresponding non-trivial zero of the Fokker-Planck operator was calculated in the last chapter for each case.

Noisy Lorenz-63 system

The Lorenz-63 system was first proposed by Edward Lorenz as an oversimplified model for atmospheric convection [113]. The corresponding ODE possesses the famous butterfly attractor. This system and its variants like Lorenz-96 are staple test problems in the field of data assimilation [28], [178] which is why we also use this system for the calculation of one-step filtering density. We use the standard parameters to define the drift.

$$\begin{aligned} \mu &= [\alpha(y-x),\, x(\rho-z)-y,\, xy-\beta z]^\top \\ \alpha &= 10\,,\, \beta = \frac{8}{3}\,,\, \rho = 28 \end{aligned} \tag{3.10}$$

We solve this system for $D = 50$ with the following Gaussian mixture as the initial condition,

$$p_0(\mathbf{x}) = \frac{1}{2\sqrt{(2\pi)^3}} \exp\left(-\frac{\|\mathbf{x}+2\times\mathbf{1}_3\|_2^2}{2}\right) + \frac{1}{2\sqrt{(2\pi)^3}} \exp\left(-\frac{\|\mathbf{x}-2\times\mathbf{1}_3\|_2^2}{2}\right) \tag{3.11}$$

where $\mathbf{1}_3$ denotes the vector with all entries 1 with respect to the standard basis in $\mathbb{R}^3$. This system has a unique solution, for a proof see appendix 3.7.1.

Noisy Thomas system

The deterministic or ODE version of this system was proposed by René Thomas [164]. This is a 3-dimensional system with cyclical symmetry in x, y, z and the corresponding ODE system has a cyclically symmetric strange attractor.

$$\begin{aligned} \mu &= [\sin y - bx,\, \sin z - by,\, \sin x - by]^\top \\ b &= 0.2 \end{aligned} \tag{3.12}$$

We solve this system for $D = 1$ with the following Gaussian mixture as the initial condition,

$$p_0(\mathbf{x}) = \frac{1}{2\sqrt{(2\pi)^3}} \exp\left(-\frac{\|\mathbf{x}+2\times\mathbf{1}_3\|_2^2}{2}\right) + \frac{1}{2\sqrt{(2\pi)^3}} \exp\left(-\frac{\|\mathbf{x}-2\times\mathbf{1}_3\|_2^2}{2}\right) \tag{3.13}$$

This system has a unique solution, for a proof see appendix 3.7.1.

To verify our method we will compare the solutions obtained by our method with Monte Carlo simulations of 3.1, for details of the Monte Carlo algorithm see appendix 3.7.2.

3.4 The p_∞+FK algorithm

First, we will go through the main challenges for solving (3.2) in high dimensions. These challenges and their solutions will naturally lead us to an algorithm for solving high-dimensional FPEs. Lastly, we will discuss various nuances associated with this algorithm.

3.4.1 Failure of the physics-informed way

Since we used deep learning to find zeros of the Fokker-Planck operator in the prequel, the most natural question becomes does an analogous algorithm work for time-dependent FPEs? For an overview of learning solutions to PDEs in a *physics-informed* fashion see section 2.5 in chapter 2 or [144], [16], [159]. As noted in section 2.5 of the last chapter, the first step to learning the solution to a PDE is to convert it into an optimization problem. For notational convenience let us first define the function space $\mathcal{F}$ as follows,

$$\mathcal{F} \stackrel{\text{def}}{=} \{f : [0,T] \times \mathbb{R}^d \to [0,+\infty) : \int_{\mathbb{R}^d} f(t,\mathbf{x})\, d\mathbf{x} = 1 \;\forall\; t \in [0,T]\} \tag{3.14}$$

Proceeding parallely to section 2.5 in chapter 2 or [159] we might be tempted to solve the following optimization problem,

$$\arg\inf_{f\in\mathcal{F}} \mathcal{J}(f) \stackrel{\text{def}}{=} \arg\inf_{f\in\mathcal{F}} \left[\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(\frac{\partial f}{\partial t}(t_i,\mathbf{x}_j) - \mathcal{L}f(t_i,\mathbf{x}_j)\right)^2 + \frac{1}{N}\sum_{j=1}^{N}(f(0,\mathbf{x}_j) - p_0(\mathbf{x}_j))^2\right] \tag{3.15}$$

where $\{\mathbf{x}_j\}_{j=1}^N$ is a uniform sample from our compact domain of interest Ω_I (see section 2.6.1 in chapter 2) and $\{t_i\}_{i=1}^M$ is a sample from $[0,T]$. But it turns out problem

(3.15) is not a *well-behaved* problem. The following proposition explains why this is the case.

Proposition 3.4.1 *Suppose* (3.2) *has a unique, strong solution* p *and*

$$\mathcal{L}f = 0 \tag{3.16}$$

also has a unique, strong, probability solution p_∞*. Then there exists a sequence of functions* $f_k \in \mathcal{F}$ *independent of the samples* $\{\mathbf{x}_j\}_{j=1}^N$, $\{t_i\}_{i=1}^M$ *such that*

$$\lim_{k\to+\infty} \mathcal{J}(f_k) = 0 \tag{3.17}$$

but

$$\lim_{k\to+\infty} f_k \neq p \tag{3.18}$$

Proof. WLOG assume that $\{t_i\}_{i=1}^M$ is ordered and $0 = t_1 < t_2 < \cdots < t_M$. Since we have included $t_1 = 0$ in our sample, while considering $\frac{\partial}{\partial t}$ at $t = t_1 = 0$ we'll only consider the right derivative. For $k \in \mathbb{N}$ define the sequence,

$$f_k(t,\mathbf{x}) = \begin{cases} p(t,\mathbf{x}), & t \leq \frac{1}{k} \\ \phi_k(t)\, p(t,\mathbf{x}) + (1-\phi_k(t))\, p_\infty(\mathbf{x}), & t > \frac{1}{k} \end{cases} \tag{3.19}$$

where,

$$\phi_k(t) = e^{1-kt} \tag{3.20}$$

Since p, p_∞ are probability densities, it is easy to see that $f_k \in \mathcal{F}$, $\forall\, k \in \mathbb{N}$. The second term or the initial condition term in $\mathcal{J}(f_k)$ vanishes since $f_k(0,\mathbf{x}) = p(0,\mathbf{x}) = p_0(\mathbf{x})$. Moreover since p is a zero of $\frac{\partial}{\partial t} - \mathcal{L}$,

$$\sum_{j=1}^N \left(\frac{\partial f_k}{\partial t}(t_1,\mathbf{x}_j) - \mathcal{L}f_k(t_1,\mathbf{x}_j) \right)^2 = 0 \tag{3.21}$$

where as noted before we have only considered the right time-derivative at $t = t_1 = 0$. Now pick $K > 0 : \frac{1}{K} < t_2$. For $k > K$, using the fact that p, p_∞ are both zeros of $\frac{\partial}{\partial t} - \mathcal{L}$ we see that,

$$\mathcal{J}(f_k) = \frac{1}{MN} \sum_{i=2}^{M} \sum_{j=1}^{N} \left(\frac{d\phi_k}{dt}(t_i)(p_\infty(\mathbf{x}_j) - p(t_i, \mathbf{x}_j)) \right)^2 \tag{3.22}$$

Due to continuity of p, p_∞ and compactness of Ω_I there exists $B \geq 0$:

$$\sup \left\{ p(t, \mathbf{x})^2, p_\infty(\mathbf{x})^2 : t \in [0, T],\ \mathbf{x} \in \Omega_I \right\} = B \tag{3.23}$$

which implies $\forall\, k > K$,

$$\mathcal{J}(f_k) \leq \frac{2B}{MN} \sum_{i=2}^{M} \sum_{j=1}^{N} \left(\frac{d\phi_k}{dt}(t_i) \right)^2 = \frac{2Be^2k^2}{MN} \sum_{i=2}^{M} \sum_{j=1}^{N} e^{-2kt_i} < 2Be^2k^2e^{-2kt_2} \tag{3.24}$$

Since $t_2 > 0$ we have

$$\lim_{k \to +\infty} \mathcal{J}(f_k) = 0 \tag{3.25}$$

But clearly the pointwise limit of f_k is given by

$$f(t, \mathbf{x}) = \begin{cases} p_0(\mathbf{x}), & t = 0 \\ p_\infty(\mathbf{x}), & t > 0 \end{cases} \tag{3.26}$$

□

The proof assumes very little about the operator $\mathcal{L}$ and hence can be extended to other partial differential equations of parabolic type. Only assumptions required to prove Proposition 3.4.1 are the existence of strong, unique probability solutions which are true for all of our example problems, see appendix 2.9.1 in chapter 2 and appendix 3.7.1 in this chapter. Since we only have finite computational resources it is sensible to investigate loss functions with finite sample-sizes and note that the proof of Proposition 3.4.1 works for any choice of samples $\{t_i\}_{i=1}^M$, $\{\mathbf{x}_j\}_{j=1}^N$ with arbitrary but finite sample-sizes. Note that the pathological sequence we constructed is independent of the samples $\{t_i\}_{i=1}^M$, $\{\mathbf{x}_j\}_{j=1}^N$ which is an important component of the failure of

the physics-informed way. In the case of stationary Fokker-Planck equations we can also construct pathological functions that make the physics-informed loss (defined in section 2.6.3 of chapter 2) vanish without converging to a solution by simply interpolating a solution at the sample points with non-solutions at unsampled points. But the physics-informed method still produces correct solutions in the stationary case since these pathological functions are sample-dependent and as the sample-size increases, they resemble a solution more and more closely. From the perspective of training a neural network to optimize problem 3.15 the main difficulty is revealed by the limit of the minimizing sequence given in (3.26). The network can behave like the initial condition for a short amount of time and then mimic p_∞ for all subsequent times losing all variation in time. Moreover, one can construct many such sequences with pathological behaviors, not to mention restricting neural networks to $\mathcal{F}$ and have them be expressive is another challenge since normalization is an extremely difficult condition to implement in high dimensions, see the discussion in sections 2.2 and 2.6.4 of chapter 2. Apart from these difficulties, for other explorations of failure modes of physics-informed neural networks see [103], [9].

3.4.2 The Feynman-Kac formula

Due to the ineffectiveness of the physics informed method, we turn to another very useful paradigm for solving high dimensional PDEs. The Feynman-Kac formula [137] has been used very successfully in recent years to solve high-dimensional PDEs [77]. They have also been used alongside deep-learning to create algorithms for semilinear parabolic PDEs [70]. Moreover, while verifying algorithms for very high dimensional PDEs one often relies on the Feynman-Kac formula to get a reference solution, see for example the examples given in sections 4, 5 of [159]. The pointwise nature of the Feynman-Kac formula makes it very attractive for high-dimensional problems, since it waives the requirement for a mesh. Below we briefly discuss a few versions of the formula and the challenges associated with using it for (3.2).

The following version of the formula appears in [131] and it is one of the most well-known versions.

Theorem 3.4.2 (Feynman-Kac) *Suppose $f \in C_0^2(\mathbb{R}^d)$ and $q \in C(\mathbb{R}^d)$. If q is bounded below then*

$$v(t,\mathbf{x}) = \mathbb{E}\left[\exp\left(-\int_0^t q(\bar{X}_s)\,ds\right) f(\bar{X}_t)\,\middle|\,\bar{X}_0 = \mathbf{x}\right] \tag{3.27}$$

is the unique solution of

$$\begin{aligned} \frac{\partial u}{\partial t} &= Au - qu \\ u(0,\mathbf{x}) &= f(\mathbf{x}) \end{aligned} \tag{3.28}$$

that's bounded on $K \times \mathbb{R}^d$ for every compact $K \subset [0,+\infty)$, where $\bar{X}_t$ is a time-homogeneous Itô process satisfying

$$d\bar{X}_t = \bar{\mu}(\bar{X}_t)\,dt + \bar{\sigma}(\bar{X}_t)\,dW_t \tag{3.29}$$

with

$$|\bar{\mu}(\mathbf{x}) - \bar{\mu}(\mathbf{y})| + |\bar{\sigma}(\mathbf{x}) - \bar{\sigma}(\mathbf{y})| \leq C|\mathbf{x}-\mathbf{y}|,\ \forall\,\mathbf{x},\mathbf{y} \in \mathbb{R}^d \tag{3.30}$$

where C is a positive constant and A is the infinitesimal generator of $\bar{X}_t$ given by

$$Af = \bar{\mu}\cdot\nabla f + \frac{1}{2}\mathrm{tr}(\bar{\sigma}\bar{\sigma}^\top \odot \nabla^2 f) \tag{3.31}$$

Note the three requirements for Theorem 3.4.2 are global Lipschitzness of $\bar{\mu}$ and $\bar{\sigma}$, vanishing of the initial condition f at infinity and the lower-boundedness of q. Lipschitzness of drift and diffusion coefficient ensures that SDE (3.29) has a unique, non-exploding, strong solution [78]. Vanishing of the initial condition f and lower-boundedness of q guarantee that the expectation in (3.27) is bounded.

If we try to apply formula (3.27) directly to the Fokker-Planck equation (3.3), we have to identify the following quantities,

$$\begin{aligned} \bar{\mu} &= -\mu \\ \bar{\sigma} &= \sigma \\ q &= -\nabla\cdot\mu \end{aligned} \tag{3.32}$$

In our examples none of the three requirements for Theorem 3.4.2 are always satisfied. Concretely, the 2D ring system (3.6) satisfies none of the requirements. Lipschitzness of the drift is too restrictive a condition to model real world phenomena and therefore attempts have been made to relax this condition. In [177] the Lipschitz condition is replaced with Hölder condition for 1D SDEs, furthermore in [58], [112] sufficient conditions for existence of pathwise unique solutions of 1D SDEs have been discussed. [174] discusses the more general case when the SDEs are multi-dimensional with non-Lipschitz drifts and gives us an identical Feynman-Kac formula for (3.28) under the condition that (3.29) has a unique, weak solution. For the sake of completeness, below we provide a special case of the more general result proved in Theorem 7.5 of [174].

Theorem 3.4.3 (Feyman-Kac with non-Lipschitz drift) *Assume the following.*

1. (3.29) *has a unique weak solution.*
2. $v(\cdot,\cdot) : [0,T] \times \mathbb{R}^d \to \mathbb{R}$ *lies in* $C^{1,2}([0,T) \times \mathbb{R}^d) \cap C_b([0,T] \times \mathbb{R}^d)$ *and satisfies the Cauchy problem* (3.28).
3. f *is uniformly bounded.*

Then v is given by the formula (3.27).

The requirement that q be lower-bounded can be easily arranged while simultaneously simplifying (3.27) by looking at an auxiliary equation. To derive this auxiliary equation we can write,

$$p(t,\mathbf{x}) = h(t,\mathbf{x})p_\infty(\mathbf{x}) \tag{3.33}$$

where p_∞ is a nonzero zero of the operator $\mathcal{L}$. Substituting (3.33) into (3.3) and recalling $D = \frac{\sigma^2}{2}$ we get,

$$\frac{\partial(p_\infty h)}{\partial t} = -\nabla\cdot(\mu p_\infty h) + \frac{\sigma^2}{2}\Delta(p_\infty h) \tag{3.34}$$

$$\implies p_\infty\frac{\partial h}{\partial t} = -h\nabla\cdot(\mu p_\infty) - p_\infty\mu\cdot\nabla h + \frac{\sigma^2}{2}(h\Delta p_\infty + 2\nabla p_\infty\cdot\nabla h + p_\infty\Delta h) \tag{3.35}$$

$$\implies p_\infty\frac{\partial h}{\partial t} = h\left[-\nabla\cdot(\mu p_\infty) + \frac{\sigma^2}{2}\Delta p_\infty\right] - p_\infty\mu\cdot\nabla h + \frac{\sigma^2}{2}(2\nabla p_\infty\cdot\nabla h + p_\infty\Delta h) \tag{3.36}$$

Since p_∞ solves the stationary FPE, the first term in the RHS of (3.36) vanishes and after rearranging terms we end up with,

$$p_\infty \frac{\partial h}{\partial t} = (\sigma^2 \nabla p_\infty - p_\infty \mu) \cdot \nabla h + p_\infty \frac{\sigma^2}{2} \Delta h \tag{3.37}$$

Therefore, h satisfies the following PDE,

$$\begin{aligned} &\frac{\partial h}{\partial t} = (\sigma^2 \nabla \log p_\infty - \mu) \cdot \nabla h + \frac{\sigma^2}{2} \Delta h, \qquad \mathbf{x} \in \mathbb{R}^d, t \in (0, T] \\ &h(0, \mathbf{x}) = \frac{p_0(\mathbf{x})}{p_\infty(\mathbf{x})}, \qquad \mathbf{x} \in \mathbb{R}^d \end{aligned} \tag{3.38}$$

Note that $\mathcal{L}$, being a linear operator, can have infinitely many distinct zeros and each such p_∞ will give rise to a different PDE for h. If we can compute p_∞ then we can compute p by computing h. In chapter 2 we provide a deep learning method to compute positive zeros of $\mathcal{L}$ and thus for our purposes it suffices to compute h. To compute h with the Feynman-Kac formula we identify the following quantities,

$$\begin{aligned} \bar{\mu} &= (\sigma^2 \nabla \log p_\infty - \mu) \\ \bar{\sigma} &= \sigma \\ q &= 0 \end{aligned} \tag{3.39}$$

In this construction q has automatically become lower-bounded. Moreover, the formula (3.28) simplifies to

$$h(t, \mathbf{x}) = \mathbb{E}\left[\frac{p_0(\bar{X}_t)}{p_\infty(\bar{X}_t)} \middle| \bar{X}_0 = \mathbf{x} \right] \tag{3.40}$$

where $\bar{X}_t$ satisfies,

$$d\bar{X}_t = (\sigma^2 \nabla \log p_\infty - \mu)\, dt + \sigma\, dW_t \tag{3.41}$$

Finally, $p(t, \mathbf{x})$ can be computed with the formula,

$$p(t, \mathbf{x}) = \mathbb{E}\left[\frac{p_0(X_t)}{p_\infty(X_t)} \middle| X_0 = \mathbf{x} \right] p_\infty(\mathbf{x}) \tag{3.42}$$

Note that even if the computed p_∞ is unnormalized since we both multiply and divide by p_∞ in (3.42), the normalization constant cancels out. Also, since $\bar{\mu}$ depends on p_∞ through $\nabla \log p_\infty$, the normalization constant for p_∞ is again eliminated and X_t appearing in (3.42) is independent of the normalization constant. Therefore, we can recover the correct p with (3.42), for any nonzero zero of $\mathcal{L}$ which can be computed following the recipe laid out in chapter 2. For the sake of convenience, we will refer to equations (3.38) and (3.41) as the *h-equation* and the *h-SDE* respectively.

We have only dealt with two of the three requirements of Theorem 3.4.2 so far by appealing to the more general version of the Feynamn-Kac formula [174] and transforming the original FPE into the h-equation. The remaining requirement of Theorem 3.4.2 is $f \in C_0^2$ or the initial condition vanishes at infinity. This can be weakened to the condition that the initial condition is uniformly bounded if we appeal to Theorem 3.4.3. This translates to $\frac{p_0}{p_\infty}$ vanishing at infinity or being uniformly bounded since $\frac{p_0}{p_\infty}$ is the initial condition for the h-equation (3.38). This assumption helps make sure that the expectation in (3.27) is bounded. Even though we do not impose this restriction in our examples, in section 3.4.4 we see that this condition is immaterial to our situation since we restrict the trajectories of the h-SDE (3.41) to a finite domain and the expectation in (3.42) is therefore bounded.

3.4.3 The algorithm

Following the discussion in section 3.4.2 we propose the following hybrid algorithm 6 for solving (3.2) in high dimensions which uses the powers of both deep learning and the Feynman-Kac formula. In order to generate trajectories of the h-SDE (3.29) we will use some time-discretized scheme like Euler-Maruyama or Milstein [98]. Just as in chapter 2, we will restrict ourselves to computing the solution on a finite domain $\mathcal{D}$ where almost all the probability mass lies throughout $t \in [0, T]$. Besides the space-time boundaries $\mathcal{D}, T$, algorithm 6 has two other hyperparameters M, N describing the time-discretization for Euler-Maruyama and the sample-size for estimating the Feynman-Kac expectation respectively.

Algorithm 6 The p_∞+FK algorithm

1. Choose $\mathcal{D}, T, M, N$ according to the PDE and available computational resources.
2. Use a known analytic zero of $\mathcal{L}$ or compute p_∞ according to algorithm 4 in chapter 2.
3. Pick $t \in [0, T]$ and $\mathbf{x} \in \mathcal{D}$. Generate N trajectories of the SDE given by,

$$dX_\tau = (\sigma^2 \nabla \log p_\infty - \mu)\, d\tau + \sigma\, dW_\tau$$

   starting from $\mathbf{x}$ and running till time t with Euler-Maruyama method. Suppose the trajectories are computed at times $0 = \tau_0 < \tau_1 < \cdots < \tau_M = t$. Let $X_j^{(i)}$ denote the j-th point (in time) in the i-th trajectory.
4. Approximate $p(\tau_j, \mathbf{x})$ with the following quantity,

$$\frac{p_\infty(\mathbf{x})}{N} \sum_{i=1}^{N} \frac{p_0(X_j^{(i)})}{p_\infty(X_j^{(i)})}$$

3.4.4 Strengths and limitations

The main strength of algorithm 6 lies in the fact that we can focus on the solutions pointwise without having to deal with global structures, thus somewhat mitigating the curse of dimensionality. Also, generating trajectories for the Feynman-Kac expectation with Euler-Maruyama lends itself to extreme parallelization, a must-have ingredient for solving high dimensional problems. The nature of the algorithm makes it applicable to a fairly large set of problems. Moreover, to compute solutions for the same problem but different initial conditions one can just reuse the same trajectory data and once the trajectory data are generated, solutions can be found for all of the time-points used to generate the trajectory data.

The limitations of algorithm 6 closely related to the properties dynamical system we are dealing with. In order to explore these, first, we need to explore the behavior of the h-SDE (3.41) for different dynamical systems.

Behavior of h-SDE trajectories and domain contraction

The stationary Fokker-Planck equation $\mathcal{L}p_\infty = 0$ has analytical solution for gradient systems. In this case, the solution can be expressed as

$$p_\infty \propto \exp\left(-\frac{2V}{\sigma^2}\right) \tag{3.43}$$

$$\implies \sigma^2 \nabla \log p_\infty = 2\mu \tag{3.44}$$

$$\implies \bar{\mu} = \mu \tag{3.45}$$

where V is defined as in (3.5), for a derivation see for example, section 2.3.1 in chapter 2. Consequently, the h-SDE for gradient systems can be written as

$$d\bar{X}_t = \mu\, dt + \sigma\, dW_t \tag{3.46}$$

So for the gradient case, the h-SDE is identical to the original SDE (3.1) describing the underlying dynamical system. Since the dynamical system possesses an attractor, the trajectories of the h-SDE are attracted to this this attractor and we can safely avoid blow-up even if the drift μ is non-Lipschitz.

But the same cannot be said for the non-gradient systems where the drift term $\bar{\mu}$ in the h-SDE might be dominated by $-\mu$ rather than being μ as in the original SDE, thus losing its attracting properties. In such cases the solutions of the h-SDE might experience finite or infinite time blow-up.

Our knowledge of $\bar{\mu}$ is determined by our knowledge of p_∞. If we do not have an analytic form for p_∞ and have computed p_∞ (up to the normalization constant) only up to the domain Ω then we can only be sure of our knowledge of $\bar{\mu}$ up to the domain Ω. Consequently, if we want to compute the solution on the domain $\mathcal{D}$ then we must select $\mathcal{D} \subset \Omega$ in a way such that the trajectories of h-SDE that start inside $\mathcal{D}$, do not leave Ω till time T with high probability. To quantify this notion, we can choose a tolerance $\varepsilon > 0$ and select $\mathcal{D}, T$ such that,

$$\xi(T, \mathcal{D}, \Omega) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{x}\sim U(\mathcal{D})}[P(\bar{X}_t \in \Omega\ \forall\ t \in [0, T] | \bar{X}_0 = \mathbf{x})] > 1 - \varepsilon \tag{3.47}$$

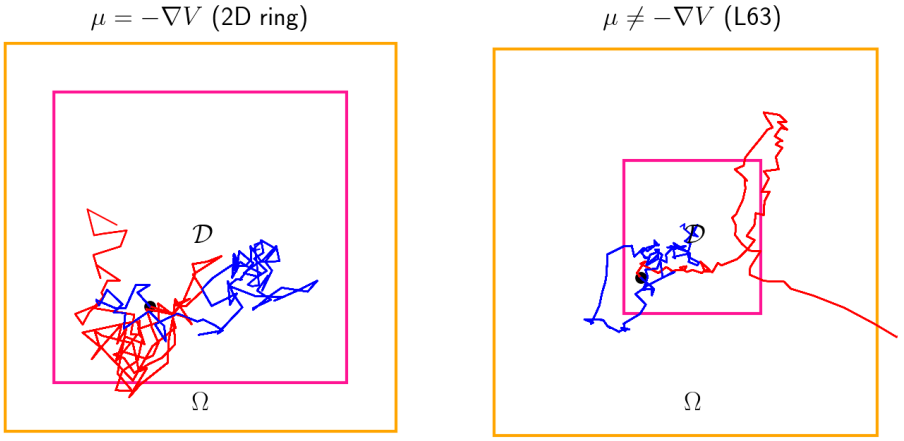

Fig. 3.1 h-SDE trajectories for various systems. In both cases a pair of trajectories start from the same point (depicted as a black dot) in $\mathcal{D}$. While the trajectories for the gradient system might leave $\mathcal{D}$ (smaller rectangle), they do not leave Ω (larger rectangle). However, the same is not true for the non-gradient system.

$\xi(T, \mathcal{D}, \Omega)$ denotes the average probability that a trajectory stays inside Ω till time T given that it started inside $\mathcal{D}$, $U(\mathcal{D})$ denotes the uniform distribution on $\mathcal{D}$ in (3.47). ξ helps us specify the space-time boundaries for the effective employment of algorithm 6. While for a gradient system, choosing Ω such that it contains the corresponding attractor, we can make sure that h-SDE trajectories originating from $\mathcal{D}$ do not leave Ω with high probability, the same cannot be said for a non-gradient system. Figure 3.1 shows the difference between h-SDE trajectories for typical gradient and non-gradient systems. Since for gradient systems the h-SDE trajectories do not leave Ω with high probability, $\xi(T, \mathcal{D}, \Omega) \approx 1$ for any T. In fact, empirically it might evaluate exactly to 1. But for non-gradient system $\xi(T, \mathcal{D}, \Omega)$ is a decreasing function of T as seen in figure 3.2 and in such cases we select the hyperparameter T for algorithm 6 according to (3.47) using a pre-chosen tolerance ε.

We are only able to solve (3.2) on a subset $\mathcal{D}$ of where we solved its stationary counterpart, namely Ω. We refer to this phenomenon as *domain contraction*. Even though we have included gradient systems in the figures 3.1, 3.2 for expository purposes, since we have perfect knowledge of p_∞ (up to the normalization constant) over entire $\mathbb{R}^d$, domain contraction is a practical issue only for non-gradient systems and T can be chosen arbitrarily large for gradient systems.

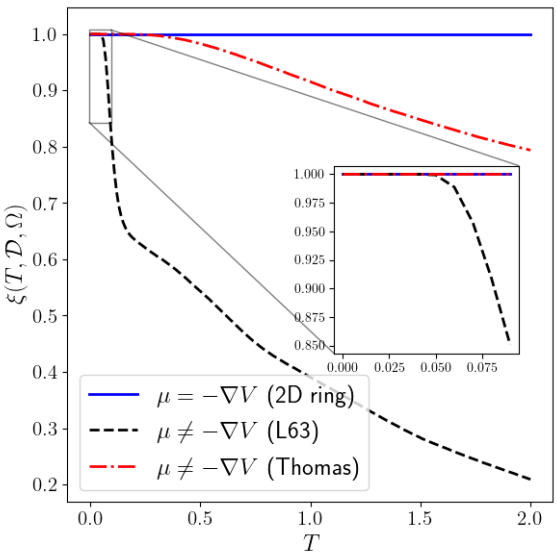

Fig. 3.2 $\xi(T, \mathcal{D}, \Omega)$ as a function of T for various systems.

Interpretation of finite time viability for non-gradient systems via Feynman-Kac on finite domains

Since we are dealing with finite domains, we can also look at the h-SDE through the lens of the Feynman-Kac formula on finite domains. In order to apply the finite domain version of the Feynman-Kac formula one requires perfect knowledge of the solution at the boundary at all times. In our case we need to know $h(t, \mathbf{x})$ on $([0, T) \times \partial\Omega) \cup (\{0\} \times \bar{\Omega})$. For the ease of discussion let us define the following quantity,

$$h(t, \mathbf{x}) = \begin{cases} \Psi(T, \mathbf{x}), \ \forall \ \mathbf{x} \in \bar{\Omega}, \ t = 0 \\ \Psi(T - t, \mathbf{x}) \ \forall \ (t, \mathbf{x}) \in [0, T) \times \partial\Omega \end{cases} \tag{3.48}$$

Assuming we know Ψ, the Feynman-Kac formula for h becomes,

$$h(t, \mathbf{x}) = \mathbb{E}\left[\Psi(\tau \wedge T, \bar{X}_{\tau \wedge T}) \middle| \bar{X}_0 = \mathbf{x}\right] \tag{3.49}$$

where τ denotes the first exit-time of $\bar{X}$ for Ω or,

$$\tau(\mathbf{x}) \stackrel{\text{def}}{=} \inf\{s > 0 : \bar{X}_s \notin \Omega\} \tag{3.50}$$

For derivations of such formulae the interested reader can refer to Theorem 4.4.5 in [64] orTheorem 4.2 in chapter 7 of [179] and its preceding section. Since in reality we only know $h(t, \mathbf{x})$ at $t = 0$ and therefore have no knowledge of $\Psi(T - t, \mathbf{x})$ at any point other than $t = 0$, we can hope to effectively use (3.49) only if the trajectories of h-SDE do not exit Ω till time T with high probability. In such a scenario $\tau \wedge T$ becomes equal to T and we are not forced to evaluate Ψ at any point where we have no knowledge of Ψ. Using the finite domain version of Feynman-Kac therefore leads us to similar conclusions as in the previous section, namely, algorithm 6 can be used to solve gradient and non-gradient systems up to arbitrarily large and finite times respectively.

Effect allowing h-SDE trajectories to leave Ω in case of perfect knowledge of p_∞ in Ω

Since p_∞ is not computed by algorithm 6 but rather is an input to it, a scenario of interest is when we have perfect (rather than approximate) knowledge of p_∞ on Ω. In such a scenario we would like the algorithm to produce reasonably good approximate solutions even when we are willing tolerate $\xi(T, \mathcal{D}, \Omega) \neq 1$ or some h-SDE trajectories leaving Ω within our chosen T. To analyze this scenario let us define our knowledge of p_∞ as,

$$\hat{p}_\infty(\mathbf{x}) = \begin{cases} p_\infty(\mathbf{x}), & \mathbf{x} \in \Omega \\ p_\infty^\sharp(\mathbf{x}), & \mathbf{x} \in \Omega^c \end{cases} \tag{3.51}$$

where $p_\infty^\sharp \neq p_\infty$ represents imperfect knowledge of p_∞ outside Ω. Similarly we can define a modified h-SDE as,

$$d\hat{X}_t = (\sigma^2 \nabla \log \hat{p}_\infty - \mu)\, dt + \sigma\, dW_t \tag{3.52}$$

and the corresponding solution generated by algorithm 6 as,

$$\hat{h}(t, \mathbf{x}) = \mathbb{E}\left[\frac{p_0(\hat{X}_t)}{\hat{p}_\infty(\hat{X}_t)} \middle| \hat{X}_0 = \mathbf{x} \right] \tag{3.53}$$

$$\hat{p}(t, \mathbf{x}) = \hat{h}(t, \mathbf{x}) \hat{p}_\infty(\mathbf{x}) \tag{3.54}$$

Now we are ready to analyze the pointwise error incurred when we allow h-SDE trajectories to escape Ω within time T with probability less than ε or $\xi(T, \mathcal{D}, \Omega) > 1 - \varepsilon$.

Proposition 3.4.4 *Let $\bar{E}, \hat{E}$ be the events[1] that $\bar{X}, \hat{X}$ stay inside Ω till time t after starting at $\mathbf{x} \in \mathcal{D} \subset \Omega$ respectively. Assume the following,*

1. $\xi(T, \mathcal{D}, \Omega) > 1 - \varepsilon$
2. $\exists$ *a constant* $K > 0$ *such that,*

$$p_\infty(\mathbf{x})\, \mathbb{E}[h_0(\bar{X}_t)|\bar{X}_0 = \mathbf{x}, \bar{E}^c],\ \hat{p}_\infty(\mathbf{x})\, \mathbb{E}[\hat{h}_0(\hat{X}_t)|\hat{X}_0 = \mathbf{x}, \hat{E}^c] < K\ \forall\ (t, \mathbf{x}) \in [0, T] \times \mathcal{D} \tag{3.55}$$

where,

$$h_0 \stackrel{\text{def}}{=} \frac{p_0}{p_\infty} \tag{3.56}$$

$$\hat{h}_0 \stackrel{\text{def}}{=} \frac{p_0}{\hat{p}_\infty} \tag{3.57}$$

Then,

$$\mathbb{E}_{\mathbf{x} \sim U(\mathcal{D})}[|\hat{p}(t, \mathbf{x}) - p(t, \mathbf{x})|] < 2K\varepsilon\ \forall\ t \in [0, T] \tag{3.58}$$

Proof. Let $\mathbf{x} \in \mathcal{D}$ and $\Delta_h = h_0 - \hat{h}_0$.

$$|\hat{p}(t, \mathbf{x}) - p(t, \mathbf{x})| = |\Delta_1 + \Delta_2| \tag{3.59}$$

where

$$\Delta_1 = p_\infty(\mathbf{x})\, \mathbb{E}\left[\frac{p_0(\bar{X}_t)}{p_\infty(\bar{X}_t)}\middle|\, \bar{X}_0 = \mathbf{x}\right] - \hat{p}_\infty(\mathbf{x})\, \mathbb{E}\left[\frac{p_0(\bar{X}_t)}{\hat{p}_\infty(\bar{X}_t)}\middle|\, \bar{X}_0 = \mathbf{x}\right] \tag{3.60}$$

and,

$$\Delta_2 = \hat{p}_\infty(\mathbf{x})\, \mathbb{E}\left[\frac{p_0(\bar{X}_t)}{\hat{p}_\infty(\bar{X}_t)}\middle|\, \bar{X}_0 = \mathbf{x}\right] - \hat{p}_\infty(\mathbf{x})\, \mathbb{E}\left[\frac{p_0(\hat{X}_t)}{\hat{p}_\infty(\hat{X}_t)}\middle|\, \hat{X}_0 = \mathbf{x}\right] \tag{3.61}$$

[1] Note that $\bar{E}, \hat{E}$ are events dependent on $t, \mathbf{x}$ but to avoid notational cluttering we do not make this dependence explicit.

Now note that,

$$\begin{aligned}\Delta_1 =& p_\infty(\mathbf{x})\,\mathbb{E}[h_0(\bar{X}_t)|\bar{X}_0=\mathbf{x},\bar{E}]\,P(\bar{E}) + p_\infty(\mathbf{x})\,\mathbb{E}[h_0(\bar{X}_t)|\bar{X}_0=\mathbf{x},\bar{E}^c]\,(1-P(\bar{E}))\\ &-\hat{p}_\infty(\mathbf{x})\,\mathbb{E}[\hat{h}_0(\bar{X}_t)|\bar{X}_0=\mathbf{x},\bar{E}]\,P(\bar{E}) - \hat{p}_\infty(\mathbf{x})\,\mathbb{E}[\hat{h}_0(\bar{X}_t)|\bar{X}_0=\mathbf{x},\bar{E}^c]\,(1-P(\bar{E}))\end{aligned} \tag{3.62}$$

Recalling that we have perfect knowledge of p_∞ inside Ω we get,

$$\begin{aligned}\Delta_1 =& p_\infty(\mathbf{x})\,[\mathbb{E}[\Delta_h(\bar{X}_t)|\bar{X}_0=\mathbf{x},\bar{E}]\,P(\bar{E}) + \mathbb{E}[\Delta_h(\bar{X}_t)|\bar{X}_0=\mathbf{x},\bar{E}^c]\,(1-P(\bar{E}))] &(3.63)\\ =& p_\infty(\mathbf{x})\,\mathbb{E}[\Delta_h(\bar{X}_t)|\bar{X}_0=\mathbf{x},\bar{E}^c]\,(1-P(\bar{E})) &(3.64)\end{aligned}$$

where we arrive at the last equality by noticing that $\Delta_h = 0$ in the event of $\bar{E}$. And,

$$\begin{aligned}\Delta_2 =& \hat{p}_\infty(\mathbf{x})\,[\mathbb{E}[\hat{h}_0(\bar{X}_t)|\bar{X}_0=\mathbf{x},\bar{E}]\,P(\bar{E}) + \mathbb{E}[\hat{h}_0(\bar{X}_t)|\bar{X}_0=\mathbf{x},\bar{E}^c]\,(1-P(\bar{E}))]\\ &-\hat{p}_\infty(\mathbf{x})\,[\mathbb{E}[\hat{h}_0(\hat{X}_t)|\hat{X}_0=\mathbf{x},\hat{E}]\,P(\hat{E}) + \mathbb{E}[\hat{h}_0(\hat{X}_t)|\hat{X}_0=\mathbf{x},\hat{E}^c]\,(1-P(\hat{E}))]\end{aligned} \tag{3.65}$$

Since $\bar{X}, \hat{X}$ follow identical dynamics inside Ω, $P(\bar{E}) = P(\hat{E})$ and we have,

$$\Delta_2 = \hat{p}_\infty(\mathbf{x})\,[\mathbb{E}[\hat{h}_0(\bar{X}_t)|\bar{X}_0=\mathbf{x},\bar{E}^c] - \mathbb{E}[\hat{h}_0(\hat{X}_t)|\hat{X}_0=\mathbf{x},\hat{E}^c]]\,(1-P(\hat{E})) \tag{3.66}$$

□

Again noting $\hat{p}_\infty(\mathbf{x}) = p_\infty(\mathbf{x})$ and $P(\bar{E}) = P(\hat{E})$ we get,

$$\Delta_1 + \Delta_2 = [p_\infty(\mathbf{x})\,\mathbb{E}[h_0(\bar{X}_t)|\bar{X}_0=\mathbf{x},\bar{E}^c] - \hat{p}_\infty(\mathbf{x})\,\mathbb{E}[\hat{h}_0(\hat{X}_t)|\hat{X}_0=\mathbf{x},\hat{E}^c]]\,(1-P(\hat{E})) \tag{3.67}$$

$$\implies |\Delta_1 + \Delta_2| < 2K\,(1-P(\hat{E})) \tag{3.68}$$

Since ξ is a monotone decreasing function of its first argument, the first assumption is equivalent to,

$$\xi(t,\mathcal{D},\Omega) > 1-\varepsilon\ \forall\ t\in[0,T] \tag{3.69}$$

Therefore, according to the definition of ξ,

$$\mathbb{E}_{\mathbf{x}\sim U(\mathcal{D})}[P(\bar{E})] > 1-\varepsilon\ \forall\ t\in[0,T] \tag{3.70}$$

which implies,

$$\mathbb{E}_{\mathbf{x}\sim U(\mathcal{D})}[1 - P(\hat{E})] < \varepsilon \; \forall \; t \in [0, T] \tag{3.71}$$

$$\implies \mathbb{E}_{\mathbf{x}\sim U(\mathcal{D})}[|\Delta_1 + \Delta_2|] < 2K\varepsilon \tag{3.72}$$

which completes our proof.

The second assumption in Proposition 3.4.4 makes sure if we compute the probability densities using only the trajectories that exit Ω before time t, we get bounded quantities independent of ε. Proposition 3.4.4 tells us, *under suitable circumstances, the average error that is caused by allowing some h-SDE trajectories to leave Ω is proportional to the fraction of trajectories that leave Ω.* If we modify the first assumption to require that the inequality holds pointwise for every $\mathbf{x} \in \mathcal{D}$ instead, we can bound the supremum norm of the error over $\mathcal{D}$ instead of the average error.

3.5 Results

In this section we describe the results in a manner that parallels the examples in section 3.3 with additional problem-specific details. We refer to the solutions obtained via algorithm 6 as the *learned* solutions in the figures below. We use the Monte-Carlo method as detailed in algorithm 7 to produce reference solutions in 2 and 3 dimensions. But in the absence of analytical solutions, we refrain from calculating the difference between the reference solution and the solution produced by algorithm 6 since Monte-Carlo solutions can be significantly erroneous, sometimes by more than an order of magnitude compared to its counterpart, even for simple problems, as evidenced by figure 2.3 in chapter 2. When dealing with high-dimensional PDEs, one widely accepted paradigm is to construct solutions that are *statistically* accurate or have the correct coarse-grained structures, especially for particle-based methods in geophysics [21], [32] and financial modelling [43]. We present our results in the same spirit. For each system we use a bimodal initial condition as described in section 3.3. Figure 3.3 shows the initial condition for the noisy Lorenz-63 system defined in (3.11). For ease of visualization we have integrated out the last coordinate. The symmetry of the initial condition implies the other 2D marginals are identical. The other initial

conditions described in section 3.3 are either identical or qualitatively similar. In order to produce 2D marginal densities we use Gauss-Legendre quadrature the details of which can be found in appendix 2.9.3.

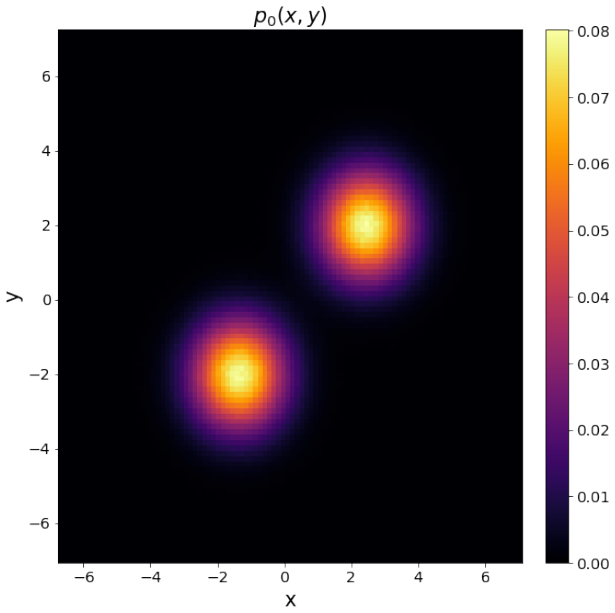

Fig. 3.3 Initial condition for the noisy Lorenz-63 system.

3.5.1 10D ring system

Figure 3.4 shows the computed solution for the $2n$D ring system presented in section 3.3.1 for $n = 5$, at time $t = 0.1$. More specifically, the left panel of figure 3.4 depicts the probability density when all but the variables x_4, x_5 are set to 0. For better visualization and comparison with a reference solution, the *learned* solution has been normalized in a way such that,

$$\int_{\mathbb{R}} \int_{\mathbb{R}} p(0.1, 0, 0, 0, 0, x_4, x_5, 0, 0, 0, 0)\, dx_4\, dx_5 = 1$$

Here we have used $M = 10$, $N = 10^5$ and $\mathcal{D} = [-2, 2]^{10}$ for algorithm 6.

There are no analytical solutions for this problem and classical methods are unsuitable due to the curse of dimensionality. Therefore, in order to compute a reference solution we use the careful design of the problem itself. The system described by (3.8) and the initial condition (3.9) is essentially n identical, decoupled 2D systems and we can approximately solve this 2D system with Monte-Carlo method which is depicted

in the right panel of figure 3.4. We use 10^7 trajectories to generate the Monte-Carlo solution. Note that, to compute the *learned* solution in figure 3.4 we use a neural network approximation of p_∞ obtained via the method described in chapter 2 rather than using the analytical version of p_∞. This demonstrates the viability of algorithm 6 in conjunction with algorithm 4 in chapter 2 for high dimensional problems.

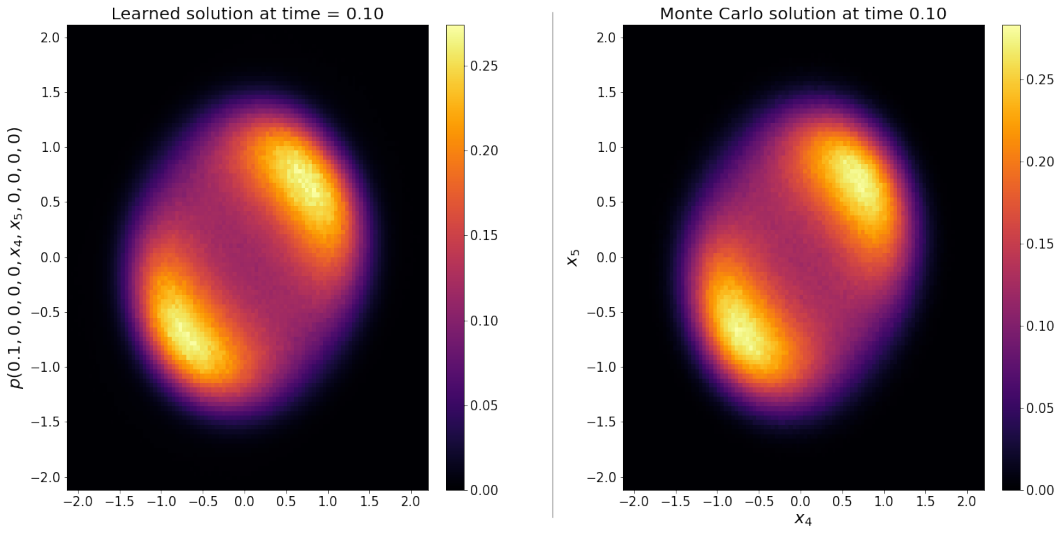

Fig. 3.4 Solutions for the 10D time-dependent system at time $t = 0.1$. The learned solution has been normalized such that $\int_{\mathbb{R}} \int_{\mathbb{R}} p(0.1, 0, 0, 0, 0, x_4, x_5, 0, 0, 0, 0)\, dx_4\, dx_5 = 1$. The right panel depicts the Monte-Carlo solution for the 2D Fokker-Planck equation corresponding to the variables x_4, x_5. The learned and Monte-Carlo solutions were computed using 10^5 (pointwise) and 10^7 trajectories respectively.

3.5.2 Noisy Lorenz-63 system

Figure 3.5 shows the solutions for the noisy Lorenz system defined by (3.10) at time $t = 0.03$ with $\mathcal{D} = [-10, 10] \times [-15, 15] \times [7, 28], \Omega = [-30, 30] \times [-40, 40] \times [0, 70]$. As seen in figure 3.2, $\xi(t, \mathcal{D}, \Omega) \approx 1$. For easier visualization we present the 2D marginals $p(t, x, y), p(t, y, z), p(t, z, x)$. To compute p_∞ in a functional form for this problem we use algorithm 4 in chapter 2. We use $M = 3$ and $N = 200$ for the learned solution and 10^7 trajectories to generate the corresponding Monte-Carlo solution. This shows that we can produce solutions with algorithm 6 that are comparable to Monte-Carlo method using several orders of magnitude fewer trajectories for each point. Both methods use 0.01 as the step length for Euler-Maruyama.

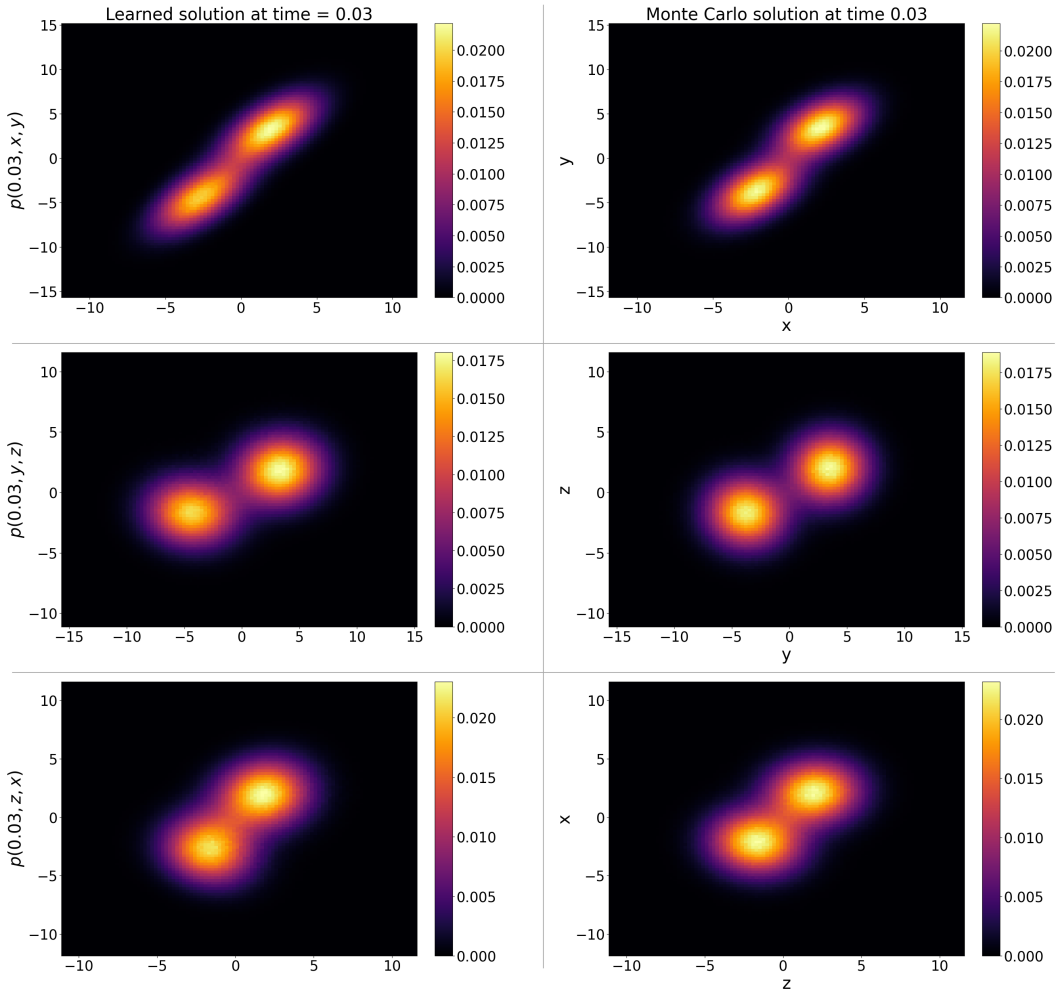

Fig. 3.5 Solutions for the noisy Lorenz 63 system at time t=0.03. The learned and Monte-Carlo solutions were computed using 200 (pointwise) and 10^7 trajectories respectively.

3.5.3 Noisy Thomas system

Figure 3.6 shows the solutions for the noisy Thomas system defined by (3.12) at time $t = 1.0$ with $\mathcal{D} = [-8, 8]^3$ and $\Omega = [-10, 10]^3$. We only present $p(t, x, y)$, noting that the symmetry of the problem renders demonstrations of the other 2D marginals redundant. We use algorithm 4 in chapter 2 for computing p_∞. We use $M = 10, N = 50$ for the learned solution and the Monte-Carlo counterpart is computed using 10^7 trajectories which reinstates our intuition that computing the Feynman-Kac

expectation requires far fewer trajectories compared to Monte-Carlo for a similar level of accuracy. Both methods use 0.1 as the step length for Euler-Maruyama. In figure 3.2 we see that $\xi(t, \mathcal{D}, \Omega) \approx 0.916$. So even after letting nearly 8.4% of the h-SDE trajectories escape Ω, we achieve a reasonable approximation.

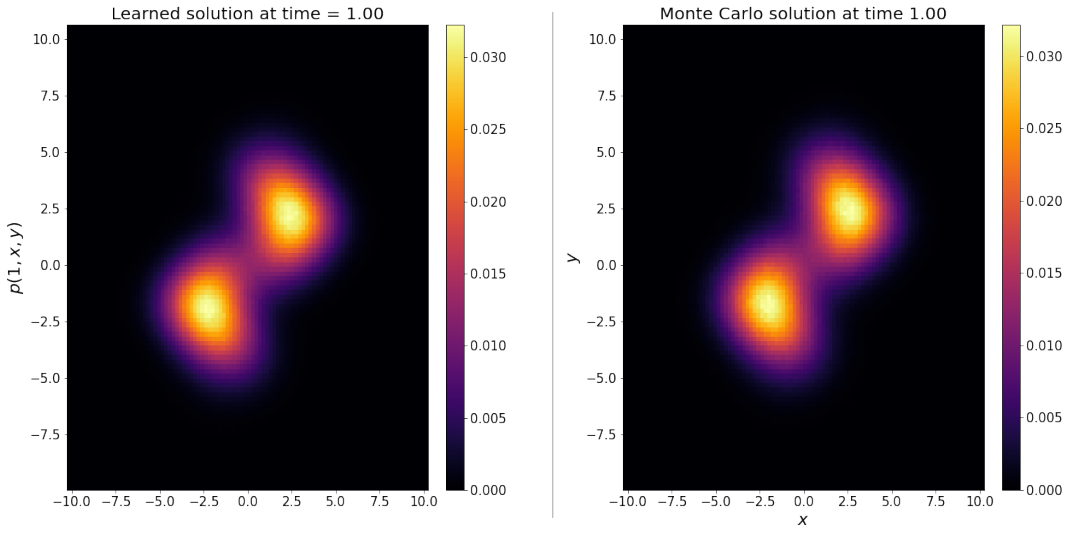

Fig. 3.6 Solutions for the noisy Thomas system at time t=1. The learned and Monte-Carlo solutions were computed using 50 (pointwise) and 10^7 trajectories respectively.

3.5.4 One step filter

Suppose in the filtering problem described in section 3.2.2 we only observe x, z coordinates of the noisy Lorenz 63 system and our observation noise standard deviation is $\sigma_o = 5$. The one-step filtering density can be written as

$$p(x_1|y_1) \propto p(x_1|x_0)p(y_1|x_1) \tag{3.73}$$

For a derivation see chapter 6 of [153] or [50]. $p(x_1|x_0)$ can be thought of as the solution to the corresponding Fokker-Planck equation at time $t = g = 0.03$ (the observation gap) with the initial condition being equal to the density of x_0,. We can calculate the likelihood $p(y_1|x_1)$ using σ_o which lets us estimate the 2D marginals of the one-step filtering density for this problem. We calculate the solution to the Fokker-Planck equation with algorithm 6 and Monte-Carlo. The final one-step filtering densities are shown in figure 3.7. Computing the filtering density with Monte-Carlo in this way is

akin to using the bootstrap particle filter, a popular nonlinear filtering algorithm, see chapter 11 of [153] or [50] for more discussion on particle filters. Therefore, we refer to the Monte-Carlo estimate for the filtering density as the particle filter estimate in figure 3.7. We use the same M, N and the same number of trajectories for Monte-Carlo as we did in section 3.5.2. It is interesting to note that the solution in figure 3.5 is bimodal whereas in the filtering density in figure 3.7 one of the mode collapses after we make an observation.

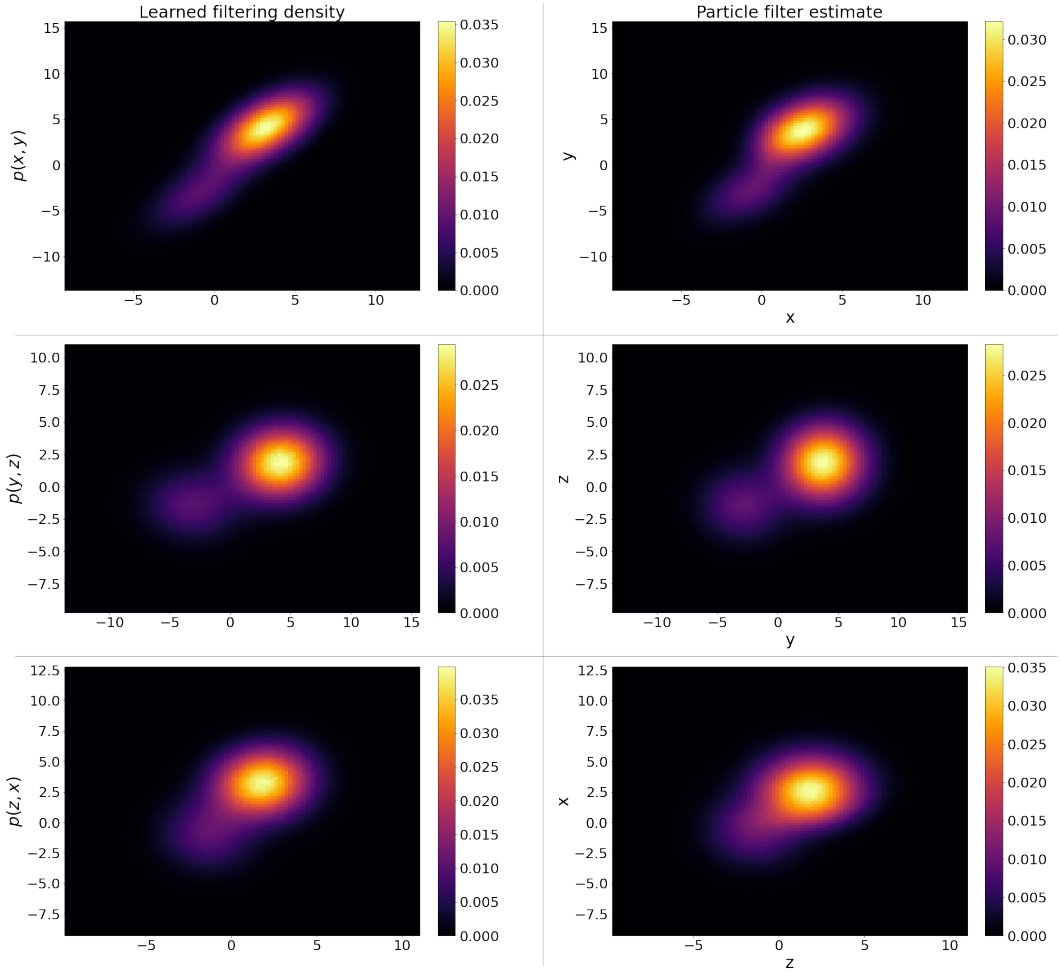

Fig. 3.7 One step filtering density for the noisy Lorenz 63 system. The learned and particle filter solutions were computed using 200 (pointwise) and 10^7 trajectories respectively.

3.6 Summary and Future Work

Physics-informed neural networks are unsuitable for a large class of partial differential equations of parabolic type. But algorithm 6 in conjunction with algorithm 4 in chapter 2 gives us a viable method for computing solutions to Fokker-Planck equations in high dimensions. The ability to focus on the solution pointwise lets us avoid the curse of dimensionality in a practical sense. To compute the Feynman-Kac expectaion we can compute the Euler-Maruyama trajectories in a highly parallel way and the same trajectories can be used to calculate solutions for different initial conditions. We are able to compute solutions for gradient systems up to arbitrarily long times. But for non-gradient system we encounter domain contraction, a phenomenon where we are able to calculate the solution on a subset of where know the stationary solution. Due to blow-up of the h-SDE for non-gradient systems we can compute solutions only up to a finite time. In case we have perfect knowledge of p_∞, under ideal conditions the error that is introduced by letting some trajectories of h-SDE escape the region where we know the stationary solution is proportional to the fraction of trajectories that escape. Even after allowing a significant portion of trajectories to escape, we can achieve decent approximations. Moreover, algorithm 6 can achieve solutions that are comparable to Monte-Carlo solutions using orders of magnitude fewer trajectories for each point. The connection between the Fokker-Planck equation and the stochastic filtering problem suggests that modifications/extensions of algorithm 6 can be useful for devising new filtering algorithms.

3.7 Appendix

3.7.1 Existence and uniqueness of solutions to example problems

Since appropriately scaling t, μ we can change (3.3) to have unit diffusion, according to section 1 of [17] and Theorem 9.4.6 and example 9.4.7 of [18] it suffices to prove that $\exists\, C > 0$ such that

$$\mu \cdot \mathbf{x} \leq C + C\|\mathbf{x}\|_2^2, \qquad \forall\ \mathbf{x} \in \mathbb{R}^d \tag{3.74}$$

in order to confirm the existence of a unique solution to (3.2).

It is easy to check that the above condition is satisfied for the drift terms for each of the examples discussed in section 3.3. For the 2D ring system in (3.6), we note that $\mu \cdot \mathbf{x} = -4r^2(r^2 - 1) \leq 1$, using the fact that the function $f(x) = x(1-x)$ achieves global maximum at $x = \frac{1}{2}$, so we can choose $C = 1$. For the noisy L63 system defined by (3.10) we have $\mu \cdot \mathbf{x} = -\alpha x^2 - y^2 - \beta z^2 + (\alpha + \rho)xy \leq (\alpha + \rho)r^2$ so it suffices to set $C = (\alpha + \rho)$. Lastly, for the noisy Thomas system defined by (3.12) we have $\mu \cdot \mathbf{x} = -br^2 + x \sin y + y \sin z + z \sin x \leq 3r \leq 3 + 3r^2$, therefore $C = 3$ is a suitable choice.

3.7.2 Monte Carlo algorithm

The relationship between (3.1) and (3.2) [149], [18] gives us the following way of estimating solutions of Fokker-Planck equations. We can evolve multiple particles according to (3.1) up to time t using Euler-Maruyama method [98], subdivide the domain $\mathcal{D}$ into d-dimensional boxes and count the how many particles lie inside each box to compute the probability density at the centers of these boxes. Here $\mathcal{N}$ denotes the multivariate normal distribution.

Algorithm 7 Monte Carlo algorithm

Sample $\{X_0^{(i)}\}_{i=1}^N \sim p_0$.
Set the time-step $h = \frac{t}{M}$.
for $j = 1, 2 \cdots, M$ **do**
  Sample $w_k^i \sim \mathcal{N}(\mathbf{0}_d, hI_d) \ \ \forall\ i$
  $X_k^{(i)} \leftarrow X_{k-1}^{(i)} + \mu\left(X_{k-1}^{(i)}\right) h + \sigma w_k^i \ \ \forall\ i$
end
Subdivide the domain of interest $\mathcal{D}$ into d-dimensional boxes.
Count the number of $X_S^{(i)}$ that are in a box to estimate the probability density at the center of the box.

Chapter 4

In this work we present deep learning implementations of two popular theoretical constrained optimization algorithms in infinite dimensional Hilbert spaces, namely, the penalty and the augmented Lagrangian methods. We test these algorithms on some toy problems originating in either calculus of variations or physics. We demonstrate that both methods are able to produce decent approximations for the test problems and are comparable in terms of different errors produced. Leveraging the common occurrence of the Lagrange multiplier update rule being computationally less expensive than solving subproblems in the penalty method, we achieve significant speedups in cases when the output of the constraint function is itself a function.

Chapter 4

Learning solutions to some toy constrained optimization problems

4.1 Introduction

In chapter 2 we solved an infinite dimensional optimization problem in a mesh-free manner in order to find the zeros of high dimensional Fokker-Planck operators. We composed this problem as an unconstrained problem since it gave us an efficient algorithm for computing the zeros. This was also sufficient for further analysis and derivation of a new algorithm for calculating solutions to time dependent Fokker-Planck equations in chapter 3. A natural question of interest is, how can one solve such infinite dimensional optimization problems in a mesh-free manner in presence of constraints? We explore this question in this final chapter.

Pierre de Fermat authored many important works on his method of maxima and minima, out of which the last two were titled *The analysis of refractions* and *The synthesis of refractions*. These contained derivations of the law of refraction now commonly known as *Snell's law*. In these papers Fermat states his intuition about the nature of physical laws as, *"nature operates by means and ways that are easiest and fastest"* [65]. Even though the ancient Greeks had considered some classic problems in calculus of variations such as isoperimetric problems [1], Fermat's proclamations are one of the first instances where we encounter the notion that the laws of physics can often be stated in terms of optimization problems. This notion takes its final form as the principle of stationary action in modern physics appearing in nearly every

subfield from classical mechanics, thermodynamics, relativity, quantum mechanics, string theory and everything in between [40], [150]. Since Fermat's time calculus of variations in general has found applications in most fields dealing with mathematical models, be it chemistry [141] or economics [68] and its stochastic counterpart is useful in economics [130] and mathematical finance [118].

Calculus of variations deals with finding functions as optimizer of functionals under constraints. Due to recent technical advancements in automatic differentiation and machine learning, it has become a popular paradigm to cast many engineering or basic science problems such as finding language models for Shakespearean text [85], learning generative models for natural images [74], solving partial differential equations [16] etc as optimization problems and then solve them using well-established optimization algorithms like stochastic gradient descent [151], [22]. This pattern very naturally yields itself to the *function-finding* problems of calculus of variations. In finite dimensions, constrained optimization problems are routinely handled with *penalty method*, *augmented Lagrangian method* and their many variants [129], [20], [13]. In infinite dimensions or for function finding problems analogues of these algorithms have been discussed extensively in terms of theory [81], [91], [51], [56]. But numerical implementation of these algorithms remain few and far between. This work aims to bridge the gap between the theory of infinite dimensional constrained optimization algorithms and their practical implementations using deep learning. Recently variational problems with essential boundary conditions have been explored by Huang et al [75]. In this work we explore more general problems. Our goal is to evaluate our algorithms, rather than solving the specific problems we list here. We, therefore, apply them on some simple toy problems with known solutions. Our problems are either taken from the classics in calculus of variations or inspired by physics.

4.2 Problem Statement and examples

In this work, we are interested in problems of the following form.

$$\begin{aligned} &\operatorname*{arginf}_{u \in X} f(u) \\ &\text{subject to } g(u) = 0 \end{aligned} \tag{4.1}$$

where $f : X \to \mathbb{R}$ and $g : X \to W$ and X, W are real Hilbert spaces. X, in particular, is an infinite dimensional Hilbert space whereas W can be either finite or infinite dimensional. This ensures that problem (4.1) is indeed an infinite-dimensional optimization problem. This setup allows us to encompass a fairly large class of problems with one or multiple constraints or even unconstrained problems if we set g to be the zero function. To better familiarize ourselves with this setup let us first look at a few examples.

4.2.1 The minimal surface problem

During the later half of the eighteenth century Lagrange in his correspondence with Euler delineated the foundations of calculus of variations and derived the famous Euler-Lagrange formula [65]. One of the problems he considered during this time asks to find the surface of least area stretched across a given contour. Although Lagrange did not find any solutions other than the plane, Euler and Jean Baptiste Meusnier later showed that helicoid and catenoid are also valid solutions to the minimal surface problem [126]. Since then the theory of minimal surfaces has seen multiple revivals with Schwarz's solution to the Björling problem [44], the discovery of Costa's surface [41] and has even found its way into mathematical physics through topics like positive energy theorem [156]. The rich theory behind minimal surfaces allows them to be expressed in many different ways [38]. Here we will work with a definition that closely resembles Lagrange's original formulation. Rather than describing the minimal surface problem in its full generality, we describe the specific problem we will solve below. We define X to be an appropriate Sobolev space, f to be an area functional and g to be the boundary condition.

$$\begin{aligned}
&\Omega = (0,1) \times (-2\pi, 2\pi),\ X = W^{1,2}(\Omega; \mathbb{R}),\ W = L^2((-2\pi, 2\pi); \mathbb{R}) \\
&f(u) = \int_{-2\pi}^{2\pi} \int_0^1 \sqrt{\left[1 + \left(\frac{\partial u}{\partial r}\right)^2\right] r^2 + \left(\frac{\partial u}{\partial \theta}\right)^2}\, dr\, d\theta \\
&g(u) : \theta \mapsto u(1, \theta) - \theta
\end{aligned} \tag{4.2}$$

Here $W^{k,p}$ denotes the Sobolev space of function with k p-integrable weak derivatives. Our question thus becomes, what is the surface of minimal area given it has a unit

helix as its boundary? The solution u^* gives us the minimal surface $(r, \theta, u^*(r, \theta))$. Note that even though we have used the standard area integral in polar coordinates, we are working beyond the standard domain of θ which is $[0, 2\pi)$. Therefore, when we visualize the solution to this problem using standard polar to Cartesian conversion we get a multivalued function or a helicoid with two full twists rather than just one, as seen in section 4.4.1.

4.2.2 Geodesics on a surface

Johann Bernoulli was interested in several problems in calculus of variations and investigated both curves of shortest length and time between two points [162], [65]. The former type of curves are known as geodesics while the latter are known as brachistochrones. After having found the solution to the brachistochrone problem Bernoulli had challenged his contemporaries to come up with their own solutions (a practice that was not uncommon in the era) to which Newton (anonymously), Jacob Bernoulli, Leibniz and de L'Hôpital had responded with their own solutions. The aftermath of this challenge would eventually lead to the infamous calculus controversy between Leibniz and Newton [134], [65]. Even though the brachistochrone problem is one of the oldest problems to be posed in calculus of variations with a rich history of mathematical rivalry associated with it, the geodesic problem would go on to outpace it in terms of importance with the development of differential geometry. Eventually geodesics would become an essential part in our understanding of motion under gravity with the advent of general relativity [171]. Here we look at the simple problem of finding the shortest path on unit a sphere given two points $(1, \theta_0, \phi_0), (1, \theta_1, \phi_1)$ (in spherical polar coordinates) on it by setting,

$$\begin{aligned} &\Omega = [\theta_0, \theta_1],\ X = W^{1,2}(\Omega; [0, 2\pi)),\ W = \mathbb{R} \\ &f(u) = \int_{\theta_0}^{\theta_1} \sqrt{1 + \left(\sin\theta \frac{du}{d\theta}\right)^2}\, d\theta \\ &g(u) = \sqrt{(u(\theta_0) - \phi_0)^2 + (u(\theta_1) - \phi_1)^2} \end{aligned} \tag{4.3}$$

If u^* is the solution then $(1, \theta, u^*(\theta))$ gives us a parametrization for the geodesic curve.

4.2.3 Grad-Shafranov equation

Grad-Shafranov equation is an elliptic partial differential equation describing the poloidal flux under ideal magnetohydrodynamics for a 2D plasma [Smitha and Hattorib]. Modelling the plasma equilibrium is an important aspect of designing magnetic confinement devices like tokamaks in the field of nuclear fusion. Although originally used for axis-symmetric tokamaks, the Grad-Shafranov equation has been analyzed for non-axis symmetric magnetohydrodynamic equilibrium as well [26]. In 1968 Solov'ev derived a family of analytic solutions for the Grad-Shafranov equation under the assumption that there is distributed toroidal current filling all space [175] and since then these Solev'ev solutions have become an import benchmarking tool for plasma equilibrium codes [86]. Below we describe the Grad-Shafranov equation, this specific version can also be found in [175].

$$\begin{aligned}
&\frac{\partial^2 u}{\partial z^2} + r\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial u}{\partial r}\right) = ar^2 + bR^2, \qquad (r,z) \in \Omega = [0.9R, 1.1R] \times [-0.1R, 0.1R] \\
&u(r,z) = \frac{1}{2}(b+c_0)R^2z^2 + c_0R\zeta z^2 + \frac{1}{2}(a-c_0)R^2\zeta^2, \qquad (r,z) \in \partial\Omega \\
&\text{where } \zeta = \frac{r^2 - R^2}{2R}, R = 1.0,\ a = 1.2,\ b = -1.0,\ c_0 = 1.1
\end{aligned} \tag{4.4}$$

In order to cast this problem into the format of (4.1), we set

$$\begin{aligned}
&X = W^{1,2}(\Omega; \mathbb{R}),\ W = L^2(\partial\Omega; \mathbb{R}) \\
&f(u) = \int_{-0.1R}^{0.1R}\int_{0.9R}^{1.1R}\left(\frac{\partial^2 u}{\partial z^2} + r\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial u}{\partial r}\right) - ar^2 - bR^2\right)^2 dr\,dz \\
&g(u) : (r,z) \mapsto \frac{1}{2}(b+c_0)R^2z^2 + c_0R\zeta z^2 + \frac{1}{2}(a-c_0)R^2\zeta^2 - u(r,z)
\end{aligned} \tag{4.5}$$

4.2.4 Beltrami fields

Beltrami fields are special vector fields that are eigenfunctions of the curl operator. They play an important role in fluid dynamics as steady solutions to the Euler equation [4]. In this problem we ask, given Beltrami boundary data, what is the magnetic field of least energy in a 3D volume? Gauss's law [83] dictates that we have to take the nondivergence of magnetic fields into account which can be done in multiple ways while formulating our question, either as a part of the Hilbert space X (since divergence is a linear operator) or as an addition to the boundary condition g. Here

we choose to impose Gauss's law as a part of the Hilbert space X.

$$\begin{aligned}
&\Omega = \left[-\frac{1}{2}, \frac{1}{2}\right]^3, \; X = \overline{\{u \in W^{1,2}(\Omega; \mathbb{R}^3) : \nabla \cdot u = 0\}}, \; W = L^2(\partial\Omega; \mathbb{R}^3) \\
&f(u) = \frac{1}{2} \int_{-\frac{1}{2}}^{\frac{1}{2}} \int_{-\frac{1}{2}}^{\frac{1}{2}} \int_{-\frac{1}{2}}^{\frac{1}{2}} |u(x,y,z)|^2 \, dx \, dy \, dz \\
&g(u) : (x,y,z) \mapsto u(x,y,z) - \begin{bmatrix} \sin(z) + \cos(y) \\ \sin(x) + \cos(z) \\ \sin(y) + \cos(x) \end{bmatrix}
\end{aligned} \tag{4.6}$$

Unlike the other problems stated here, this problem is *manufactured* and has no direct practical applications but nevertheless serves as an interesting toy problem.

4.3 Methodology

Before discussing our algorithm for solving the problems stated in section 4.2, we briefly look at constrained optimization algorithms for finite dimensional problems and their infinite dimensional analogues since they illustrate the guiding principles that will help us devise our own algorithm.

4.3.1 Constrained optimization algorithms in finite dimensions

Unconstrained optimization problems are typically easier to solve than constrained optimization problems and they are often solved using variants of gradient descent or Newton's method [20], [129]. Therefore, in order to solve constrained problems we often transform them into unconstrained problems first. When X, W are finite dimensional, in order to solve problem (4.1) we can convert it from a constrained optimization problem to a sequence of unconstrained subproblems as follows,

$$u_k = \operatorname*{arginf}_{u \in X} \mathcal{L}(u, \mu_k) \stackrel{\text{def}}{=} f(u) + \frac{\mu_k}{2} |g(u)|_W^2, \; k = 1, 2, \cdots \tag{4.7}$$

where $|\cdot|_W$ and denotes the canonical norm on W and $\{\mu_k\}_{k=1}^{\infty}$ is a positive increasing sequence such that $\mu_k \uparrow \infty$ and u_k is the exact global solution to the k-th subproblem. It can be shown that every limit point of $\{u_k\}_{k=1}^{\infty}$ is a solution to the original constrained

problem, for a proof see Theorem 17.1 in [129] or for a local version of the statement see Theorem 1 in [140]. The strategy of using subproblems (4.7) to solve (4.1) is known as the quadratic *penalty method*. In case we have access to only approximate solutions to the subproblems then limit points of these approximate solutions might be infeasible or they might only satisfy the first order KKT condition [104], [66], [19] rather than being global minimizers. Moreover, the Hessian of the unconstrained objective function $\mathcal{L}$ becomes ill-conditioned as $\mu_k \uparrow \infty$. If we attempt to find an approximate solution to (4.7) by trying to satisfy the first order condition using Newton's method, we quickly run into significant numerical errors when μ_k is large. For an excellent discussion of the nuances associated with the penalty method see chapter 17 in [129]. Typically the convergence rate of the quadratic penalty method is $O(k^{-\frac{1}{2}})$ but for strongly convex problems it increases to $O(k^{-1})$ [110], [140].

In order to avoid ill-conditioning we can modify our subproblems as follows,

$$u_k = \operatorname*{arginf}_{u \in X} \mathcal{L}_A(u, \mu_k, \lambda_k) \stackrel{\text{def}}{=} f(u) + \frac{\mu_k}{2}|g(u)|_W^2 + \langle \lambda_k, g(u) \rangle_W, \; k = 1, 2, \cdots \tag{4.8}$$

where $\langle \cdot \rangle_W$ is the canonical inner product on W, $\{\mu_k\}_{k=1}^{\infty}$ is a positive, nondecreasing sequence but not necessarily unbounded and λ_k follows the update rule,

$$\lambda_{k+1} = \lambda_k + \mu_k g(u_k) \tag{4.9}$$

This update rule is the consequence of an attempt to satisfy the first order condition for optimality. In this setting, μ_k and λ_k play the roles of the penalty factor for deviating from the constraint and the Lagrange multiplier respectively. This method is known as the *augmented Lagrangian method*. It can be shown that, under suitable conditions, if λ_k converges to λ^* then $\exists\, \mu^* > 0$ such that for $\mu \geq \mu^*$, any local solution to the original constrained problem is a local minimizer of $\mathcal{L}_A(\cdot, \mu, \lambda^*)$, see Theorem 17.5 in [129] or for a global version of this statement see Theorem 5.2 in [14]. This gives the augmented Lagrangian method a strong theoretical foundation but in practice we might only have approximate knowledge of λ^*. In such a case i.e. when λ_k is close to λ^*, it can be shown that a local minimizer of $\mathcal{L}_A(\cdot, \mu, \lambda_k)$ solves the original constrained problem for large enough μ, see Theorem 17.6 in [129] or Proposition 4.2.3 in [13]. These results show that the augmented Lagrangian method can approximately solve (4.1) when

either the penalty μ is large or we have good knowledge of the optimal Lagrange multiplier λ^*. The appeal of the augmented Lagrangian method therefore lies in the possibility that we can replace the requirement that $\mu_k \uparrow \infty$ with the convergence of the Lagrange multiplier λ_k thus avoiding the ill-conditioning of the Hessian and all the numerical difficulties that arise because of it.

4.3.2 Constrained optimization algorithms in infinite dimensions

When X is infinite dimensional and $f(\cdot), \langle g(\cdot), g(\cdot)\rangle_W$ are lower-semicontinuous functionals, limit points of the exact global solutions to the subproblems are solutions to the original constrained problem, for a proof see Theorem 1 in [51] and for a treatment of the penalty method on general topological spaces see [56]. The augmented Lagrangian method has also been extended in many different scenarios where X is an infinite dimensional Hilbert space by Ito and Kunisch [80], [79], [81]. More recently the case when X is an infinite dimensional Banach space has been considered by Kanzow et al [91]. If we assume that problem (4.1) has a solution, f, g are twice continuously Fréchet differentiable near the solution, derivative of g at the solution is surjective, a Lagrange multiplier exists for this solution, f is weakly lower-semicontinuous and g maps weakly convergent sequences to weakly convergent sequences then we can prove that the augmented Lagrangian subproblems have local solutions, the Lagrange multiplier λ_k converges to λ^* and these local solutions converge to a local solution of (4.1). For an in-depth look at the technical details, we refer the reader to chapter 3 of [81].

4.3.3 Deep learning variants for infinite dimensional algorithms

The first challenge in implementing these algorithms is representing elements of X and W when they are infinite dimensional. A direct approach to do this would be to represent an element of X as a neural network $u_\eta^{\mathcal{A}}$ and in case W is infinite dimensional, we can represent the Lagrange multiplier as another network $\lambda_\xi^{\mathcal{B}}$ where η, ξ represent the trainable parameters of the networks and $\mathcal{A}, \mathcal{B}$ represent the structure or architecture of the networks. Universal approximation theorems [139], [46] imply that with appropriately chosen $\mathcal{A}, \mathcal{B}$ we might be able to sufficiently approximate

the solutions to the subproblems. Suppose the dimensions of η, ξ or the number of trainable parameters are a, b respectively. Then the subproblems in the penalty algorithm can be rewritten as,

$$\eta_k = \underset{\eta \in \mathbb{R}^a}{\operatorname{arginf}} \mathcal{L}(u_\eta^{\mathcal{A}}, \mu_k) = f(u_\eta^{\mathcal{A}}) + \frac{\mu_k}{2} |g(u_\eta^{\mathcal{A}})|_W^2, \ k = 1, 2, \cdots \tag{4.10}$$

Similarly, the subproblems in the augmented Lagrangian algorithm can be rewritten as,

$$\eta_k = \underset{\eta \in \mathbb{R}^a}{\operatorname{arginf}} \mathcal{L}_A(u_\eta^{\mathcal{A}}, \mu_k, \lambda_{\xi_k}^{\mathcal{B}}) = f(u_\eta^{\mathcal{A}}) + \frac{\mu_k}{2} |g(u_\eta^{\mathcal{A}})|_W^2 + \langle \lambda_{\xi_k}^{\mathcal{B}}, g(u_\eta^{\mathcal{A}}) \rangle_W, \ k = 1, 2, \cdots \tag{4.11}$$

When W is infinite dimensional, the Lagrange multiplier update rule can be rewritten as,

$$\lambda_{\xi_{k+1}}^{\mathcal{B}} = \lambda_{\xi_k}^{\mathcal{B}} + \mu_k g(u_{\eta_k}^{\mathcal{A}}) \tag{4.12}$$

Note that, with this rewriting our infinite-dimensional subproblems have become finite dimensional since a is finite. The update rule (4.12) cannot be implemented directly since the Lagrange multipliers are functions rather than finite dimensional vectors in this scenario. Therefore, we try to find the optimal ξ_{k+1} that makes the left hand side of (4.12) functionally or in an L^2 sense, equal to the right hand side by solving the following optimization problem,

$$\xi_{k+1} = \underset{\xi \in \mathbb{R}^b}{\operatorname{arginf}} \left| \lambda_\xi^{\mathcal{B}} - \lambda_{\xi_k}^{\mathcal{B}} - \mu_k g(u_{\eta_k}^{\mathcal{A}}) \right|_W^2 \tag{4.13}$$

If we solve K subproblems then we approximate our final solution as u_{η_K}. In order to solve each subproblem in (4.10) and (4.11) we use our solution to the last subproblem as an initial guess and perform gradient descent. In order to solve (4.13) we use ξ_k as an initial guess since we expect the sequence $\lambda_{\xi_k}^{\mathcal{B}}$ to converge. The selection of μ_k is an important part of the algorithm but no general purpose techniques for this selection are available in the literature. Larger μ_k results in better theoretical convergence rates while deteriorating the numerical estimates at same time, see section 3.3 in [81] for

comments on this topic. The update rule in (4.9) can be modified in different ways to achieve better estimates when one has some extra information about the problem (4.1), see the ALM algorithm in [81] for example. But in practice such information is nearly impossible to come by and therefore we will stick to the simple update rule in (4.9). The deep learning variants of the penalty method, augmented Lagrangian method when W is finite dimensional and augmented Lagrangian method when W is infinite dimensional can found in algorithms 8 (P^∞), 9 (AL_F^∞), 10 (AL_∞^∞) respectively.

Algorithm 8 P^∞: Infinite dimensional penalty algorithm

Choose architecture $\mathcal{A}$, penalty factor sequence $\{\mu_k\}_{k=1}^\infty$, adaptive learning rate $\{\delta_{k,j}\}_{k,j=1}^\infty$ and stopping criteria $\{P_k\}_{k=1}^\infty, \{Q_{k,j}\}_{k,j=1}^\infty$
$k \leftarrow 0$
while stopping criterion P_k is not met **do**
  $k \leftarrow k+1$
  if $k = 1$ **then**
    Initialize η randomly
  else
    Initialize $\eta \leftarrow \eta_{k-1}$
  end if
  $j \leftarrow 1$
  while stopping criterion $Q_{k,j}$ is not met **do**
    $L \leftarrow f(u_\eta^{\mathcal{A}}) + \frac{\mu_k}{2}|g(u_\eta^{\mathcal{A}})|_W^2$
    $\eta \leftarrow \eta - \delta_{k,j}\nabla_\eta L$
    $j \leftarrow j+1$
  end while
  $\eta_k \leftarrow \eta$
end while
$u_{\eta_k}^{\mathcal{A}}$ is our approximate solution to (4.1)

4.4 Results

In this section we describe the results along with the specific details of the algorithms used for each problem. We solve every problem with both penalty and augmented Lagrangian methods.

- **Architecture**: We use two different types of architecture in our experiments, see appendix 4.6.1 for a description of these types and refer to table 4.1 for details of $\mathcal{A}, \mathcal{B}$ used in the experiments. We use the same architecture to represent the approximate solutions of penalty and augmented Lagrangian algorithms.

Algorithm 9 $\text{AL}_{\text{F}}^{\infty}$: Infinite dimensional augmented Lagrangian algorithm when W is finite dimensional

Choose architectures $\mathcal{A}$, penalty factor sequence $\{\mu_k\}_{k=1}^{\infty}$, adaptive learning rate $\{\delta_{k,j}\}_{k,j=1}^{\infty}$ and stopping criteria $\{P_k\}_{k=1}^{\infty}, \{Q_{k,j}\}_{k,j=1}^{\infty}$
$k \leftarrow 0$
while stopping criterion P_k is not met **do**
  $k \leftarrow k+1$
  if $k=1$ **then**
    Initialize η randomly
  else
    Initialize $\eta \leftarrow \eta_{k-1}$
  end if
  $j \leftarrow 1$
  while stopping criterion $Q_{k,j}$ is not met **do**
    $L \leftarrow f(u_{\eta}^{\mathcal{A}}) + \frac{\mu_k}{2}|g(u_{\eta}^{\mathcal{A}})|_W^2 + \langle \lambda_{\xi_k}^{\mathcal{B}}, g(u_{\eta}^{\mathcal{A}})\rangle_W$
    $\eta \leftarrow \eta - \delta_{k,j}\nabla_{\eta} L$
    $j \leftarrow j+1$
  end while
  $\eta_k \leftarrow \eta$
  $\xi_{k+1} \leftarrow \xi_k + g(u_{\eta_k}^{\mathcal{A}})$
end while
$u_{\eta_k}^{\mathcal{A}}$ is our approximate solution to (4.1)

- **Stopping criteria**: Although sophisticated stopping criteria such as the norm of the gradient of the objective function in the subproblem falling below a pre-selected threshold, can be used for algorithms 8, 9, 10, here we stop the loops after a pre-selected number of iterations is reached.

- **Number of gradient descent steps**: We denote this number of iterations with P for the outer loops in algorithms 8, 9, 10, Q for the inner loops in algorithms 8, 9 and Q^A, Q^B for the first and second inner loops in the algorithm 10 respectively. We define E to be the total number of gradient descent steps used. Therefore, $E = PQ$ for algorithms 8 and 9 and $E = P(Q^A + Q^B)$ for algorithm 10. For each problem we use the same total gradient descent steps E for both the penalty and the augmented Lagrangian algorithms. To facilitate this, when W is finite dimensional we use the same Q for algorithms 8 and 9 and when W is infinite dimensional we set $Q^A = Q^B = Q/2$. We use the popular Adam optimizer [97] to perform the gradient descent steps.

Algorithm 10 AL_∞^∞: Infinite dimensional augmented Lagrangian algorithm when W is infinite dimensional

Choose architectures $\mathcal{A}$, $\mathcal{B}$, penalty factor sequence $\{\mu_k\}_{k=1}^\infty$, adaptive learning rates $\{\delta_{k,j}^A\}_{k,j=1}^\infty$, $\{\delta_{k,j}^B\}_{k,j=1}^\infty$ and stopping criteria $\{P_k\}_{k=1}^\infty$, $\{Q_{k,j}^A\}_{k,j=1}^\infty$, $\{Q_{k,j}^B\}_{k,j=1}^\infty$
$k \leftarrow 0$
while stopping criterion P_k is not met **do**
  $k \leftarrow k+1$
  if $k = 1$ **then**
    Initialize η randomly
    Initialize ξ randomly
  else
    Initialize $\eta \leftarrow \eta_{k-1}$
  end if
  $j \leftarrow 1$
  while stopping criterion $Q_{k,j}^A$ is not met **do**
    $L \leftarrow f(u_\eta^{\mathcal{A}}) + \frac{\mu_k}{2}|g(u_\eta^{\mathcal{A}})|_W^2 + \langle \lambda_{\xi_k}^{\mathcal{B}}, g(u_\eta^{\mathcal{A}})\rangle_W$
    $\eta \leftarrow \eta - \delta_{k,j}^A \nabla_\eta L$
    $j \leftarrow j+1$
  end while
  $\eta_k \leftarrow \eta$
  $j \leftarrow 1$
  while stopping criterion $Q_{k,j}^B$ is not met **do**
    $L_\lambda \leftarrow \left|\lambda_\xi^{\mathcal{B}} - \lambda_{\xi_k}^{\mathcal{B}} - \mu_k g(u_{\eta_k}^{\mathcal{A}})\right|_W^2$
    $\xi \leftarrow \xi - \delta_{k,j}^B \nabla_\xi L_\lambda$
    $j \leftarrow j+1$
  end while
  $\xi_{k+1} \leftarrow \xi$
end while
$u_{\eta_k}^{\mathcal{A}}$ is our approximate solution to (4.1)

- **Learning rate**: We use an initially oscillating and finally decaying learning rate δ that depends on 7 distinct hyperparameters. The oscillatory nature of δ as seen in figure 4.1, is employed to essentially rejuvenate the previously decaying learning rate every time we start an inner loop in the algorithms. For details of this learning rate δ, see appendix 4.6.3. While using algorithms 8 and 9 we set,

$$\delta_{k,j} = \delta((k-1)Q + j) \tag{4.14}$$

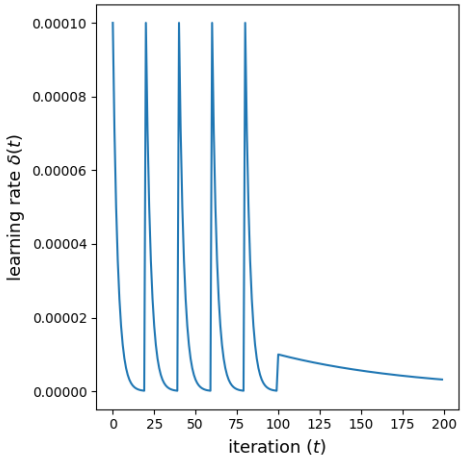

Fig. 4.1 Example behavior of the oscillating learning rate δ

and while using algorithm 10 we use,

$$\delta^A_{k,j} = \delta((k-1)Q^A + j) \tag{4.15}$$

$$\delta^B_{k,j} = \delta((k-1)Q^B + j) \tag{4.16}$$

- **The penalty factors**: We use a stopped geometric sequence as our μ_k,

$$\mu_k = \min\{\mu_1 r^{k-1}, \mu_{\max}\} \tag{4.17}$$

  The exact values of $\mu_1, \mu_{\max}, r$ for various problems can be found in table 4.2.

- **Computation of functionals**: To compute the functional f and when W is infinite dimensional, the functional $|\cdot|_W$, we use either Gauss-Legendre quadrature (in 1 or 2 dimensions) or a Monte Carlo estimate (in 3 dimensions).

- **Errors**: We evaluate our algorithms using three different kinds of errors produced. If $\hat{u}$ is the solution produced by an algorithm and u^{true} is the true solution of the problem, we define the absolute error to be an weighted L^2-norm of $\hat{u} - u^{\text{true}}$,

$$\text{absolute error} = \sqrt{\frac{\int_\Omega (\hat{u} - u^{\text{true}})^2 \, dV}{\int_\Omega \, dV}} \tag{4.18}$$

where dV denotes a volume element in Ω. We define the relative objective error to be the relative error in the value of the objective function,

$$\text{relative objective error} = \left| \frac{f(\hat{u}) - f(u^{\text{true}})}{f(u^{\text{true}})} \right| \tag{4.19}$$

Lastly, we define the constraint error to be how closely $\hat{u}$ satisfies the constraint,

$$\text{constraint error} = \frac{|g(\hat{u})|_W}{Z} \tag{4.20}$$

We set $Z = 1$ for the geodesic problem (4.3) and $Z = \sqrt{|\partial\Omega|}$ for all the other problems. The use of normalization constants in (4.18), (4.20) results in errors that are akin to RMSE. With these definitions we are now ready to present the results.

4.4.1 The minimal surface problem

Figure 4.2 shows the true and approximate solutions to the minimal surface problem in Cartesian coordinates. The true solution for this problem is a helicoid. We use

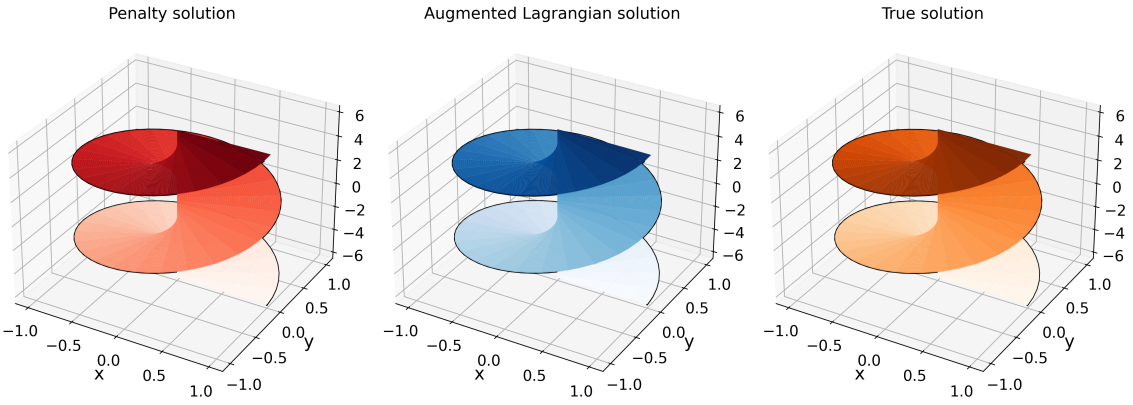

Fig. 4.2 Solutions to the minimal surface problem. Darker color implies higher u value.

$E = 20000$ total gradient steps and $P = 1000$ subproblems for this problem. This setup implies to calculate solutions to the subproblems, we use $Q = 20$ gradient descent steps for the penalty algorithm (P^∞) and $Q^A = Q^B = 10$ gradient steps for the augmented Lagrangian algorithm (AL^∞_∞). Figure 4.3 shows various errors as functions of gradient descent steps for this problem. In terms of algorithms 8, 9, 10, iteration in figure 4.3 can be understood as $(k-1)E/P + j$. The errors and run times have been plotted every 100 steps. Both methods are able to produce good approximations

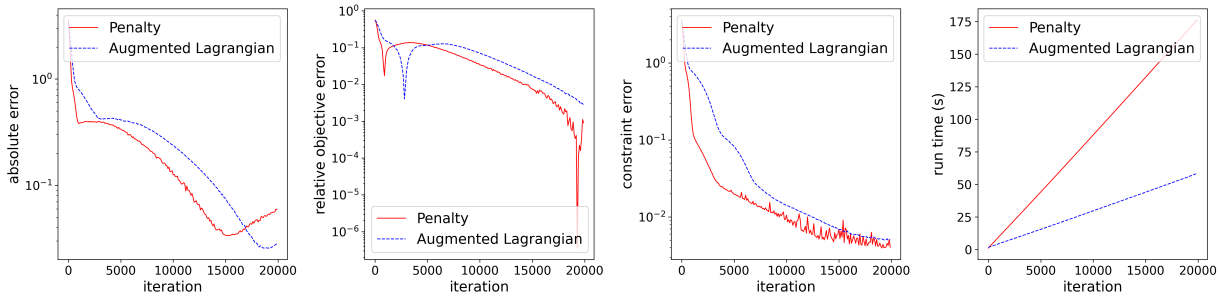

Fig. 4.3 Errors and run times for the minimal surface problem as functions of gradient descent steps. The errors have been plotted in a semilog fashion. All quantities have been plotted every 100 steps.

but the penalty method fluctuates more during training compared to its counterpart for this problem. Looking at table 4.1 we see that sizes of the networks representing the solution and the Lagrange multiplier a, b are close to each other. But solving the problem (4.13) is computationally much cheaper than solving the problem (4.11). This is a typical scenario and is reflected in the run times of the algorithms in figure 4.3 where the augmented Lagrangian is 3.016 times faster than its counterpart.

4.4.2 Geodesics on a surface

Figure 4.4 shows the approximate solutions to the geodesic problem. The true solution to this problem is the arc between the given points that lies on the great circle connecting them. We use $E = 50000$ and $P = 2500$ for this problem which implies the number of gradient steps used to solve subproblems for the both the penalty (P^{∞}) and the augmented Lagrangian ($\mathrm{AL}_{\mathrm{F}}^{\infty}$) algorithms is $Q = 20$. Figure 4.5 shows the errors and run times for this problem as a function of gradient steps. Although the absolute error decreases without significant fluctuations, $\mathrm{AL}_{\mathrm{F}}^{\infty}$ shows considerably more fluctuations in other errors compared to its counterpart during training. Since (4.11) is computationally more expensive to solve than (4.10) and we use the same number of gradient descent steps to solve both of them, the augmented Lagrangian algorithm is slower in this case as seen in figure 4.5.

In figure 4.4 the points on the sphere are chosen to be nonantipodal, leading to a unique solution to the geodesic problem. In case these points are antipodal there are infinitely many great circles that connect them, leading to infinitely many solutions. The solution set in this degenerate case is homeomorphic to a connected 1D manifold.

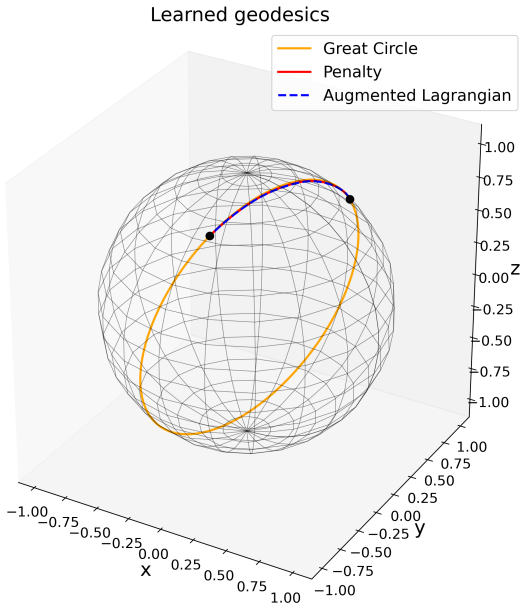

Fig. 4.4 Solutions to the geodesic problem. The distance between the black dots is being minimized.

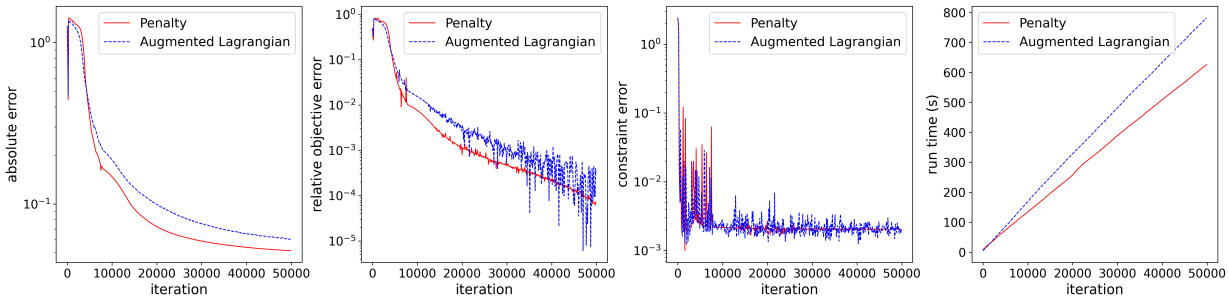

Fig. 4.5 Errors and run times for the geodesic problem as functions of gradient descent steps. The errors have been plotted in a semilog fashion. All quantities have been plotted every 100 steps.

Figure 4.6 shows the geodesics learned with the penalty method for such a degenerate problem. As can be seen, the solution produced by the penalty algorithm (P^∞) quite surprisingly does not hop from one great circle to another but rather stays on a single great circle with more and more gradient descent steps. Even though the true minima of the original problem lie on a connected manifold, the discretized version of the problem where we look for $\eta \in \mathbb{R}^a$, might have a different distribution of minima, where possibly each minimum can be separated from any other minimum by open sets. The geometry of the loss landscape in the subproblems is an interesting topic and requires further investigation.

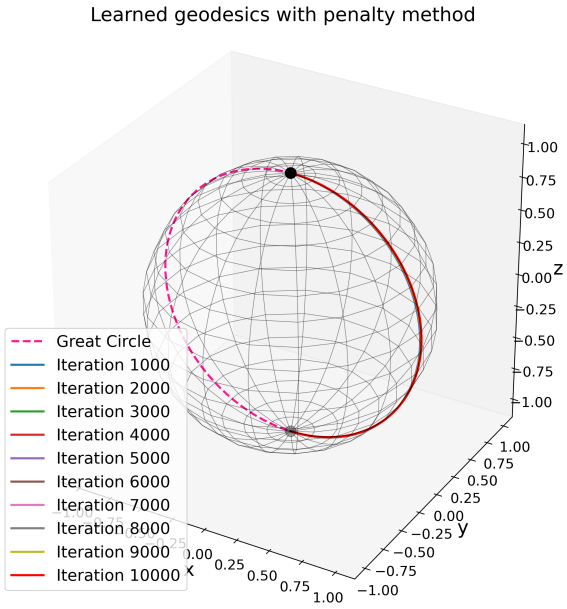

Fig. 4.6 Solutions to the geodesic problem when the points (black dots) are antipodal

4.4.3 Grad-Shafranov equation

Figure 4.7 shows the approximate and true solutions for the Grad-Shafranov equation. We use $E = 50000$ and $P = 2500$ for this problem which implies that we use $Q = 20$

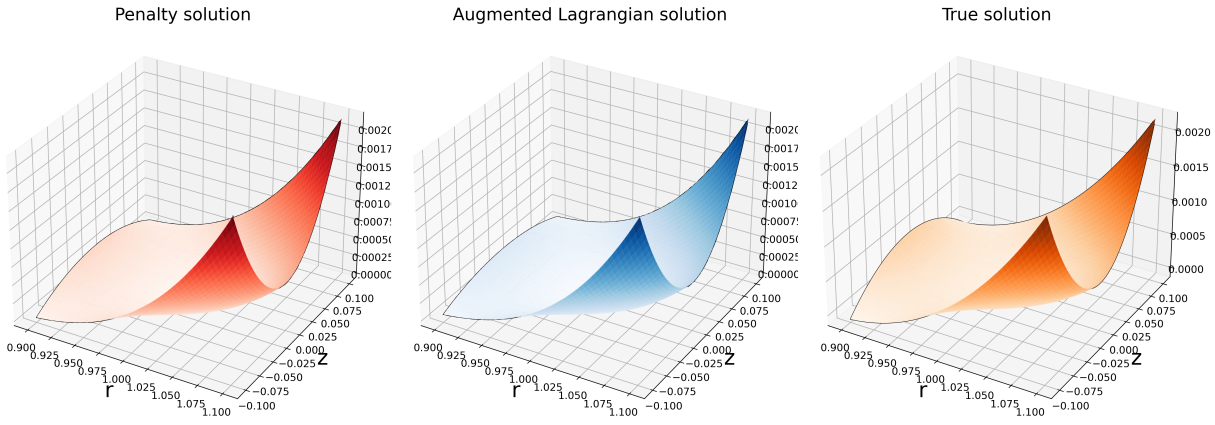

Fig. 4.7 Solutions to the Grad-Shafranov equation. Darker color implies higher u value.

gradient descent steps to solve the subproblems in penalty method (P^{∞}) and $Q^A = Q^B = 10$ gradient descent steps to solve the subproblems in augmented Lagrangian method (AL^{∞}_{∞}). Figure 4.8 shows the various errors and run times for this problem. Both methods produce qualitatively similar error curves. The Lagrange multiplier in this case can be thought of as a function of a single variable since it is a member of $W^{1,2}(\partial\Omega;\mathbb{R})$ and $\partial\Omega$ is homeomorphic to a closed curve in $\mathbb{R}^2$. But for a convenient implementation we represent it as a function of two variables (or on Ω) as seen in

table 4.1. This does not cause any practical issues since we never encounter the multiplier anywhere expect $\partial\Omega$ during the run time of $\mathrm{AL}_\infty^\infty$ i.e. we only optimize the multiplier on $\partial\Omega$. According to table 4.1 the size of the multiplier network b is significantly smaller than the size of the solution network a in this case. These choices reflect the fact that inherently the solution and the multiplier are functions of two and one variables respectively. Even though on a machine, (4.11) is more expensive to solve than (4.10), since (4.13) is much cheaper to solve than (4.10), the augmented Lagrangian is ultimately much faster than its counterpart in this case as is seen in figure 4.8.

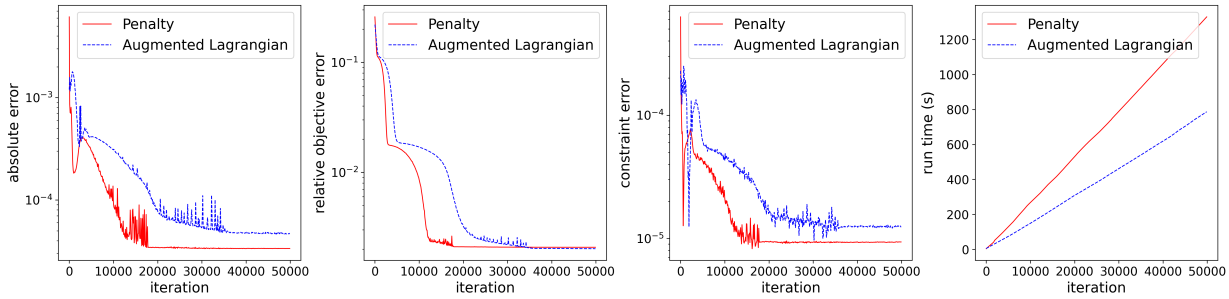

Fig. 4.8 Errors and run times for the Grad-Shafranov problem as functions of gradient descent steps. The errors have been plotted in a semilog fashion. All quantities have been plotted every 100 steps.

4.4.4 Beltrami fields

Figure 4.9 shows the approximate and true solutions for the Beltrami field problem. We use $E = 50000$ and $P = 2500$ for this problem which implies that we use $Q = 20$ gradient descent steps to solve the subproblems in penalty method (P^∞) and $Q^A = Q^B = 10$ gradient descent steps to solve the subproblems in augmented Lagrangian method ($\mathrm{AL}_\infty^\infty$). Figure 4.10 shows the various errors and run times for this problem. In order to enforce Gauss's law we represent the solution magnetic field $\hat{u}$ as the curl of a vector potential and we represent this vector potential as a neural network H_η^A,

$$\hat{u} = \nabla \times H_\eta^A \tag{4.21}$$

This clearly allows many such vector potentials to generate solutions to our problem but we do not concern ourselves with gauge-fixing since we are only interested in

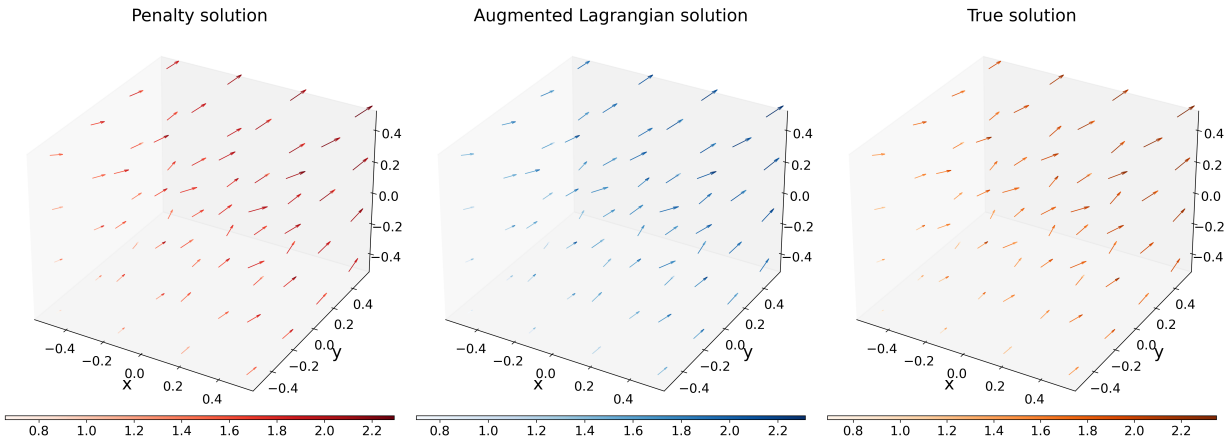

Fig. 4.9 Solutions to the Beltrami field problem. The arrows-lengths have been normalized for visual clarity. The colorbars depict the magnitude of the vectors.

the magnetic field $\nabla \times H_\eta^A$ rather than the potential H_η^A itself. According to table 4.1 the vector potential network is much larger than the Lagrange multiplier network in this case. These choices reflect the fact that the solution is a function defined on a volume while the Lagrange multiplier is a function defined on a surface. Just like the previous problem we implement the multiplier as a function of 3 variables while optimizing its values only the boundary $\partial\Omega$ without causing any practical issues. The functional f in this problem requires integration on a 3D volume. To do so we use a Monte-Carlo estimate with only 10^3 points to maintain a relatively low computational budget. We do achieve qualitatively decent approximations as seen in figure 4.9 but our computational parsimony along with the use of the curl operator leading to more floating point operations and errors result in higher absolute and constraint errors compared to the other problems discussed here. The relative computational ease of solving (4.13) when compared to (4.10) results in a faster performance for the augmented Lagrangian algorithm.

4.5 Summary and Future Work

In this work we present some practical implementations of popular constrained optimization algorithms in infinite dimensional Hilbert spaces. Both penalty and augmented Lagrangian methods produce decent, comparable solutions for our toy problems in terms of various metrics. Dimension of the Hilbert space W is an important factor when it comes to the difference in the run times of penalty and augmented

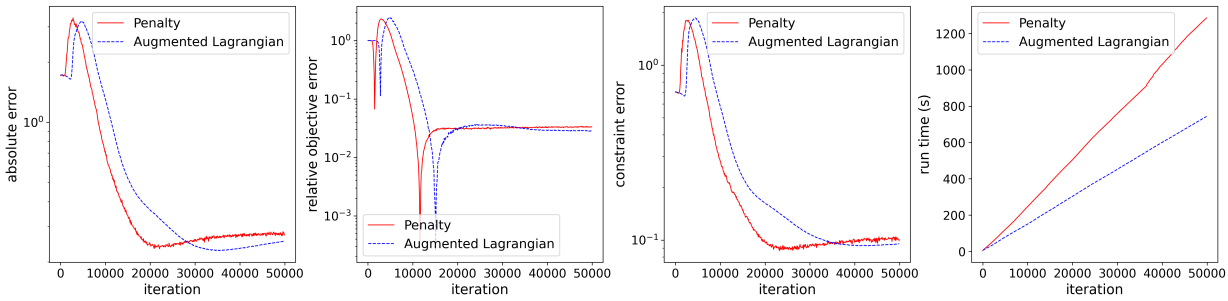

Fig. 4.10 Errors and run times for the Beltrami field problem as functions of gradient descent steps. The errors have been plotted in a semilog fashion. All quantities have been plotted every 100 steps.

Lagrangian algorithms. When W is infinite dimensional we might be able to achieve considerably lower run times for the augmented Lagrangian algorithm compared to the penalty method since updating the multiplier is generally less expensive than solving the subproblem in the penalty method. Some constraints like Gauss's law can be implemented reasonably well through architecture. Different update rules for the Lagrange multiplier lead to different variants of algorithm 10 exploring which is a worthwhile topic for future research. The geometry of the objective function in the subproblems and the distribution of their optima also deserve further exploration.

Acknowledgements

This work was supported by the Department of Atomic Energy, Government of India, under project no. RTI4001 and the Infosys-TIFR Leading Edge travel grant. The author would like to thank Zhisong Qu, Matthew Hole and Robert Dewar for useful discussions on these topics.

4.6 Appendix

4.6.1 Architecture

We use two different architectures here which we refer to as LSTM (long short-term memory) and FF (feed-forward). LSTM type networks have been employed to solve partial differential equations (see [159] or chapter 2 of this thesis) and are useful for avoiding vanishing gradients in deep networks [158], [169]. We use the same LSTM

architecture that appears in chapter 2. For the sake of completeness we describe this architecture in detail below. Here $\mathbf{0}_k$ implies a zero vector of dimension k, $\odot$ implies the Hadamard product, d_I, d_O denote the input and output dimensions and θ represents the ordered set of training parameters. The architecture has two numeric hyperparameters m, L describing the size of individual layers and the number of LSTM blocks respectively. Activation $\mathtt{A}$ is the elementwise tanh function and finally, $n_\theta^{\text{LSTM}(m,L,d_I,d_O)}$ represents the complete network.

$$i \in \{1, 2, \cdots, L\} \tag{4.22}$$

$$\mathtt{c}_0(\mathbf{x}) \stackrel{\text{def}}{=} \mathbf{0}_m \tag{4.23}$$

$$\mathtt{h}_0(\mathbf{x}) \stackrel{\text{def}}{=} \mathbf{0}_{d_I} \tag{4.24}$$

$$\mathtt{f}_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{A}(\mathtt{W}_f^{(i)}\mathbf{x} + \mathtt{U}_f^{(i)}\mathtt{h}_{i-1}(\mathbf{x}) + \mathtt{b}_f^{(i)}) \tag{4.25}$$

$$\mathtt{g}_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{A}(\mathtt{W}_g^{(i)}\mathbf{x} + \mathtt{U}_g^{(i)}\mathtt{h}_{i-1}(\mathbf{x}) + \mathtt{b}_g^{(i)}) \tag{4.26}$$

$$\mathtt{r}_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{A}(\mathtt{W}_r^{(i)}\mathbf{x} + \mathtt{U}_r^{(i)}\mathtt{h}_{i-1}(\mathbf{x}) + \mathtt{b}_r^{(i)}) \tag{4.27}$$

$$\mathtt{s}_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{A}(\mathtt{W}_s^{(i)}\mathbf{x} + \mathtt{U}_s^{(i)}\mathtt{h}_{i-1}(\mathbf{x}) + \mathtt{b}_s^{(i)}) \tag{4.28}$$

$$\mathtt{c}_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{f}_i(\mathbf{x}) \odot \mathtt{c}_{i-1}(\mathbf{x}) + \mathtt{g}_i(\mathbf{x}) \odot s_i(\mathbf{x}) \tag{4.29}$$

$$\mathtt{h}_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{r}_i(\mathbf{x}) \odot \mathtt{A}(\mathtt{c}_i(\mathbf{x})) \tag{4.30}$$

$$\mathtt{d}_L(\mathbf{y}) \stackrel{\text{def}}{=} \mathtt{W}^\top\mathbf{y} + \mathtt{b}, \qquad \mathbf{y} \in \mathbb{R}^m \tag{4.31}$$

$$n_\theta^{\text{LSTM}(m,L,d_I,d_O)} \stackrel{\text{def}}{=} \mathtt{d}_L \circ \mathtt{h}_L \tag{4.32}$$

Here $\{\mathtt{f}_i, \mathtt{g}_i, \mathtt{r}_i, \mathtt{s}_i, \mathtt{c}_i, \mathtt{h}_i : i = 1, \cdots, L\} \cup \{\mathtt{d}_L\}$ are the hidden layers and

$$\theta = \{\mathtt{W}_f^{(i)}, \mathtt{U}_f^{(i)}, \mathtt{b}_f^{(i)}, \mathtt{W}_g^{(i)}, \mathtt{U}_g^{(i)}, \mathtt{b}_g^{(i)}, \mathtt{W}_r^{(i)}, \mathtt{U}_r^{(i)}, \mathtt{b}_r^{(i)}, \mathtt{W}_s^{(i)}, \mathtt{U}_s^{(i)}, \mathtt{b}_s^{(i)} : i = 1, \cdots, L\} \cup \{\mathtt{W}, \mathtt{b}\} \tag{4.33}$$

is the set of the trainable parameters. The dimensions of these parameters are given below,

$$\mathtt{W}_f^{(i)}, \mathtt{W}_g^{(i)}, \mathtt{W}_r^{(i)}, \mathtt{W}_s^{(i)} \in \mathbb{R}^{m \times d_I} \tag{4.34}$$

$$\mathtt{U}_f^{(i)}, \mathtt{U}_g^{(i)}, \mathtt{U}_r^{(i)}, \mathtt{U}_s^{(i)} \in \begin{cases} \mathbb{R}^{m \times d_I}, & \text{if } i = 1 \\ \mathbb{R}^{m \times m}, & \text{otherwise} \end{cases} \tag{4.35}$$

$$\mathtt{b}_f^{(i)}, \mathtt{b}_g^{(i)}, \mathtt{b}_r^{(i)}, \mathtt{b}_s^{(i)} \in \mathbb{R}^m \tag{4.36}$$

$$\mathtt{W} \in \mathbb{R}^{d_O \times m}, \mathtt{b} \in \mathbb{R}^{d_O} \tag{4.37}$$

which implies the size of the network or cardinality of θ is $4m[d_I(L+1) + m(L-1) + 1] + d_O(m+1)$.

We use $n_\phi^{\mathrm{FF}(m,L,d_I,d_O)}$ to represent a simple feed-forward network without any skip connections with ϕ being the set of trainable parameters. In this case the hyperparameters m, L denote the size of an individual layer and the number of hidden layers respectively.

$$i \in \{1, \cdots, L-1\} \tag{4.38}$$

$$\mathtt{f}_0(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{A}(\mathtt{W}_f^{(0)}\mathbf{x} + \mathtt{b}_f^{(0)}) \tag{4.39}$$

$$\mathtt{f}_i(\mathbf{x}) \stackrel{\text{def}}{=} \mathtt{A}(\mathtt{W}_f^{(i)}\mathtt{f}_{i-1}(\mathbf{x}) + \mathtt{b}_f^{(i)}) \tag{4.40}$$

$$\mathtt{d}_L(\mathbf{y}) \stackrel{\text{def}}{=} \mathtt{W}^\top \mathbf{y} + \mathtt{b}, \qquad \mathbf{y} \in \mathbb{R}^m \tag{4.41}$$

$$n_\phi^{\mathrm{FF}(m,L,d_I,d_O)} \stackrel{\text{def}}{=} \mathtt{d}_L \circ \mathtt{f}_L \tag{4.42}$$

So ϕ is given by,

$$\phi = \{\mathtt{W}_f^i, \mathtt{b}_f^i : i = 0, 1, \cdots L-1\} \cup \{\mathtt{W}, \mathtt{b}\} \tag{4.43}$$

and the dimensions of these trainable parameters are given below,

$$\mathtt{W}_f^{(i)} \in \begin{cases} \mathbb{R}^{m\times d_I}, & \text{if } i=0 \\ \mathbb{R}^{m\times m}, & \text{otherwise} \end{cases} \tag{4.44}$$

$$\mathtt{b}_f^{(i)} \in \mathbb{R}^m \tag{4.45}$$

$$\mathtt{W} \in \mathbb{R}^{d_O\times m}, \mathtt{b} \in \mathbb{R}^{d_O} \tag{4.46}$$

The size of $n_\phi^{\mathrm{FF}(m,L,d_I,d_O)}$ is therefore $m[d_I+(L+1)+m(L-1)]+(d_O-1)(m+1)$. We list the network architectures and sizes used in our experiments in table 4.1.

Problem	Algorithm	$\mathcal{A}$	a	$\mathcal{B}$	b
Minimal surface	P^∞	FF(50, 3, 2, 1)	5300	-	-
Minimal surface	$\mathrm{AL}_\infty^\infty$	FF(50, 3, 2, 1)	5300	FF(50, 3, 1, 1)	5250
Geodesic	P^∞	LSTM(50, 3, 1, 1)	21051	-	-
Geodesic	$\mathrm{AL}_\mathrm{F}^\infty$	LSTM(50, 3, 1, 1)	21051	scalar	1
Grad-Shafranov	P^∞	LSTM(50, 3, 2, 1)	21851	-	-
Grad-Shafranov	$\mathrm{AL}_\infty^\infty$	LSTM(50, 3, 2, 1)	21851	FF(50, 3, 2, 1)	5300
Beltrami field	P^∞	LSTM(50, 3, 3, 3)	22753	-	-
Beltrami field	$\mathrm{AL}_\infty^\infty$	LSTM(50, 3, 3, 3)	22753	FF(50, 3, 3, 3)	5452

Table 4.1 Networks used in various experiments

4.6.2 Penalty factor

Recall that we use a stopped geometric sequence as our μ_k (4.17). We list the hyperparameters that determine μ_k in table 4.2.

Problem	Algorithm	μ_1	$\mu_{\max}$	r
Minimal surface	P^∞	100	5000	1.01
Minimal surface	$\mathrm{AL}_\infty^\infty$	100	5000	1.01
Geodesic	P^∞	100	500	1.01
Geodesic	$\mathrm{AL}_\mathrm{F}^\infty$	100	500	1.01
Grad-Shafranov	P^∞	100	1000	1.01
Grad-Shafranov	$\mathrm{AL}_\infty^\infty$	100	1000	1.01
Beltrami field	P^∞	100	5000	1.01
Beltrami field	$\mathrm{AL}_\infty^\infty$	100	5000	1.01

Table 4.2 Hyperparameters of the penalty factor for various experiments

4.6.3 Learning rate

The learning rate δ depends on 7 hyperparameters which are the initial learning rate (L_0), initial decay rate (D_0), initial decay steps (S_0), tipping point (T), final learning rate (L_1), final decay rate (D_1), final decay steps (S_1). We define δ as,

$$\delta(t) = \begin{cases} L_0 D_0^{\frac{t \bmod S_0}{S_0}}, & t < T \\ L_1 D_1^{\frac{t-T}{S_1}}, & t \geq T \end{cases} \tag{4.47}$$

We list these parameters for our experiments in table 4.3. We use,

$$S_0 = \frac{2E}{P} \tag{4.48}$$

$$T = \left\lfloor \frac{2E(\mu_{\max} - \mu_1)}{Pr} \right\rfloor \tag{4.49}$$

$$S_1 = E - T \tag{4.50}$$

For definitions of E, P see section 4.6. In case $T > E$, we never reach the tipping point and hence do not list L_1, D_1.

Problem	Algorithm	L_0	D_0	L_1	D_1
Minimal surface	P^{∞}	10^{-4}	10^{-1}	-	-
Minimal surface	$\mathrm{AL}^{\infty}_{\infty}$	10^{-4}	2×10^{-1}	-	-
Geodesic	P^{∞}	10^{-3}	10^{-1}	10^{-4}	10^{-2}
Geodesic	$\mathrm{AL}^{\infty}_{\mathrm{F}}$	10^{-3}	10^{-1}	10^{-4}	10^{-2}
Grad-Shafranov	P^{∞}	10^{-4}	10^{-1}	10^{-6}	10^{-2}
Grad-Shafranov	$\mathrm{AL}^{\infty}_{\infty}$	10^{-4}	10^{-1}	10^{-6}	10^{-2}
Beltrami field	P^{∞}	10^{-4}	10^{-1}	-	-
Beltrami field	$\mathrm{AL}^{\infty}_{\infty}$	10^{-4}	10^{-1}	-	-

Table 4.3 Hyperparameters of the learning rate for various experiments

Summary and Future Work

In the first chapter we explore the topic of filter stability in data assimilation from a numerical standpoint. We devise a new definition for practical computations of filter stability and show that this definition is a stronger version of an existing definition in a meaningful way. We show that both particle filter and EnKF achieve stability exponentially fast in the test cases. For particle filters we see that as the number of particles increase we get closer to the true definition numerical filter stability. Before reaching stability the filter RMSE and the Wasserstein distance are linearly related at least in the test cases. This somewhat surprisingly indicates that RMSE can be used to determine stability of a numerical filter in a practical sense. The mechanism behind this linear relationship is an important topic for further exploration.

In the second chapter we see that deep learning can be efficiently employed to compute zeros of high-dimensional Fokker-Planck operators in a functional form. For the same sample size deep learning seems to produce more accurate solutions compared to Monte Carlo even in low dimensions which is expected since deep learning has direct access to derivatives of the solution or higher order information while Monte Carlo does not. For small values of the loss function, it is linearly related to the distance from truth which justifies algorithm 4. Learning fine-grained structures in the solutions and the geometry or optima of the loss function are important topics for future research.

In the third chapter we show that physics informed networks are not suitable for a class of parabolic PDEs. But a combination of the zeros calculated with algorithm 4 and the Feynman-Kac formula can be used to solve time-dependent Fokker-Planck equations. The main benefit of this method is the ability to solve equations in a pointwise fashion which is immensely useful in fighting the curse of dimensionality.

This method can be used solve gradient systems up to arbitrary times. For non-gradient systems, due to the blow-up of the auxiliary h-SDE we can solve the corresponding Fokker-Planck equation only up to a finite time. We also prove that if we have perfect knowledge of the stationary solution up to a normalization constant, the error incurred due to letting some trajectories of the h-SDE escape our domain of knowledge, is proportional to the fraction of trajectories that escape. Investigation of the behavior of algorithm 6 for pathological solutions such as heavy-tailed distributions is an interesting topic for future exploration.

In the last chapter we examine penalty and augmented Lagrangian algorithms for infinite-dimensional problems. We devise deep learning versions of these algorithms and successfully estimate solutions to a few toy problems. We observe that both methods produce qualitatively and quantitatively similar errors in the test cases. When the output of the constraint is a function, the augmented Lagrangian algorithm can run significantly faster than the penalty method for the same number of iterations since the Lagrange multiplier update rule is typically computationally cheaper than solving the penalty subproblem. Additionally, in most cases we can use a smaller multiplier network (compared to the solution network) to achieve faster run times for the augmented Lagrangian algorithm. Understanding the structure of the set of minima of the loss function for degenerate problems, application of our algorithms in more complex scenarios and investigation of potential failure modes of these algorithms are some worthwhile future research directions.

We end our discussion with some potential applications of the different original and adapted methods presented in this thesis. The Sinkhorn algorithm gives us a practical way to evaluate the stability of data assimilation algorithms. Online versions of this algorithm [125] can be used to deal with large empirical distributions efficiently. The omnipresence of Fokker-Planck equations makes the methods described here or their modifications applicable for a variety of problems such as wind turbine modelling [170], analysis of open quantum systems [52], design of new data assimilation algorithms [10] etc. Note that, these examples are high-dimensional in nature and hence are ideal for capitalizing on the mesh-free methods described here. The constrained optimization problem considered in the final chapter is fairly generic. Therefore, even though our treatment of the topic was not application-oriented, it is easy to imagine

possible applications in a wide array of problems, such as shape optimization, optimal control of discrete systems, financial optimization, variational image processing [30] and ill-posed inverse problems [6] to name a few.

References

[1] (Alexandrinus.), P. and Ver Eecke, P. (1933). *Pappus d'Alexandrie: La collection mathématique*. Blanchard.

[2] Allen, L. J. (2010). *An introduction to stochastic processes with applications to biology*. CRC press.

[3] Apte, A., Hairer, M., Stuart, A., and Voss, J. (2007). Sampling the posterior: an approach to non-Gaussian data assimilation. *Physica D*, 230:50–64.

[4] Aris, R. (2012). *Vectors, tensors and the basic equations of fluid mechanics*. Courier Corporation.

[5] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

[6] Arridge, Simon and Maass, Peter and Öktem, Ozan and Schönlieb, Carola-Bibiane (2019). Solving inverse problems using data-driven models. *Acta Numerica*, 28:1–174.

[7] Asch, M., Bocquet, M., and Nodet, M. (2016). *Data Assimilation: Methods, Algorithms, and Applications*. SIAM.

[8] Ballani, J. and Grasedyck, L. (2013). A projection method to solve linear systems in tensor format. *Numerical linear algebra with applications*, 20(1):27–43.

[9] Basir, S. (2022). Investigating and mitigating failure modes in physics-informed neural networks (pinns). *arXiv preprint arXiv:2209.09988*.

[10] Belyaev, K., Meyers, S., and O'Brien, J. (2000). Application of the Fokker-Planck equation to data assimilation into hydrodynamical models. *Journal of Mathematical Sciences*, 99:1393–1402.

[11] Bengtsson, L., Ghil, M., and Källén, E. (1981). *Dynamic meteorology: data assimilation methods*, volume 36. Springer.

[12] Berezin, Y. A., Khudick, V., and Pekker, M. (1987). Conservative finite-difference schemes for the Fokker-Planck equation not violating the law of an increasing entropy. *Journal of Computational Physics*, 69(1):163–174.

[13] Bertsekas, D. P. (1995). Athena scientific. *Nonlinear programming*, 4.

[14] Birgin, E. G. and Martínez, J. M. (2014). *Practical augmented Lagrangian methods for constrained optimization*. SIAM.

[15] Bishop, A. N. and Del Moral, P. (2017). On the Stability of Kalman–Bucy Diffusion Processes. *SIAM Journal on Control and Optimization*, 55(6):4015–4047.

[16] Blechschmidt, J. and Ernst, O. G. (2021). Three ways to solve partial differential equations with neural networks—a review. *GAMM-Mitteilungen*, 44(2):e202100006.

[17] Bogachev, V. I., Krasovitskii, T. I., and Shaposhnikov, S. V. (2021). On uniqueness of probability solutions of the Fokker-Planck-Kolmogorov equation. *Sbornik: Mathematics*, 212(6):745.

[18] Bogachev, V. I., Krylov, N. V., Röckner, M., and Shaposhnikov, S. V. (2022). *Fokker–Planck–Kolmogorov Equations*, volume 207. American Mathematical Society.

[19] Boltyanski, V., Martini, H., and Soltan, V. (1998). *Geometric methods and optimization problems*, volume 4. Springer Science & Business Media.

[20] Bonnans, J.-F., Gilbert, J. C., Lemaréchal, C., and Sagastizábal, C. A. (2006). *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media.

[21] Bosler, P. A. (2013). *Particle methods for geophysical flow on the sphere*. PhD thesis, University of Michigan.

[22] Bottou, L. and Bousquet, O. (2007). The tradeoffs of large scale learning. *Advances in neural information processing systems*, 20.

[23] Brezis, H. and Brézis, H. (2011). *Functional analysis, Sobolev spaces and partial differential equations*, volume 2. Springer.

[24] Bris, C. L. and Lions, P.-L. (2008). Existence and uniqueness of solutions to fokker–planck type equations with irregular coefficients. *Communications in Partial Differential Equations*, 33(7):1272–1317.

[25] Buduma, N., Buduma, N., and Papa, J. (2022). *Fundamentals of deep learning*. " O'Reilly Media, Inc.".

[26] Burby, J. W., Kallinikos, N., and MacKay, R. S. (2020). Generalized Grad–Shafranov equation for non-axisymmetric MHD equilibria. *Physics of Plasmas*, 27(10).

[27] Carrassi, A., Bocquet, M., Bertino, L., and Evensen, G. (2018). Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535.

[28] Carrassi, A., Bocquet, M., Demaeyer, J., Grudzien, C., Raanes, P., and Vannitsem, S. (2022). Data assimilation for chaotic dynamics. *Data Assimilation for Atmospheric, Oceanic and Hydrologic Applications (Vol. IV)*, pages 1–42.

[29] Carrassi, A., Trevisan, A., Descamps, L., Talagrand, O., and Uboldi, F. (2008). Controlling instabilities along a 3DVar analysis cycle by assimilating in the unstable subspace: a comparison with the EnKF. *Nonlinear Process. Geophys.*, 15:503–521.

[30] Cassel, K. W. (2013). *Variational methods with applications in science and engineering*. Cambridge University Press.

[31] Cérou, F. (2000). Long time behavior for some dynamical noise free nonlinear filtering problems. *SIAM J. Control. Optim.*, 38:1086–1101.

[32] Chen, N. and Majda, A. J. (2018). Efficient statistically accurate algorithms for the Fokker–Planck equation in large dimensions. *Journal of Computational Physics*, 354:242–268.

[33] Chigansky, P. (2006). Stability of nonlinear filters: A survey. Lecture notes, Petropolis, Brazil.

[34] Chigansky, P., Liptser, R., and Van Handel, R. (2009). Intrinsic methods in filter stability. *Handbook of Nonlinear Filtering*.

[35] Chopin, N. and Papaspiliopoulos, O. (2020). *Convergence and Stability of Particle Filters*, pages 167–188. Springer International Publishing, Cham.

[36] Chow, P.-L. and Khasminskii, R. (2014). Almost sure explosion of solutions to stochastic differential equations. *Stochastic Processes and their Applications*, 124(1):639–645.

[37] Cioica-Licht, P. A., Hutzenthaler, M., and Werner, P. T. (2022). Deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear partial differential equations. *arXiv preprint arXiv:2205.14398*.

[38] Colding, T. H. and Minicozzi, W. P. (2011). *A course in minimal surfaces*, volume 121. American Mathematical Soc.

[39] Compo, G. P., Whitaker, J. S., Sardeshmukh, P. D., Matsui, N., Allan, R. J., Yin, X., Gleason, B. E., Vose, R. S., Rutledge, G., Bessemoulin, P., et al. (2011). The twentieth century reanalysis project. *Quarterly Journal of the Royal Meteorological Society*, 137(654):1–28.

[40] Coopersmith, J. (2017). *The lazy universe: An introduction to the principle of least action*. Oxford University Press.

[41] Costa, C. J. (1984). Example of a complete minimal immersion in ir 3 of genus one and three-embedded ends. *Boletim da Sociedade Brasileira de Matemática-Bulletin/Brazilian Mathematical Society*, 15:47–54.

[42] Crisan, D. and Rozovskii, B. (2011). *The Oxford handbook of nonlinear filtering*. Oxford University Press.

[43] Cui, S., Kurganov, A., and Medovikov, A. (2015). Particle methods for PDEs arising in financial modeling. *Applied Numerical Mathematics*, 93:123–139.

[44] Darboux, G. (1896). *Leçons sur la théorie générale des surfaces et les applications géométriques du calcul infinitésimal: ptie. Déformation infiniment petite et réprésentation sphérique. Notes et additions: I. Sur les méthodes d'approximations successives dans la théorie des équations différentielles, par E. Picard. II. Sur les géodésiques à intégrales quadratiques, par G. Koenigs. III. Sur la théorie des équations aux dérivées partielles du second ordre, par E. Cosserat. IV-XI. Par l'auteur. 1896*. Gauthier-Villars.

[45] Datta, L. (2020). A survey on activation functions and their relation with Xavier and He normal initialization. *arXiv preprint arXiv:2004.06632*.

[46] De Ryck, T., Lanthaler, S., and Mishra, S. (2021). On the approximation of functions by tanh neural networks. *Neural Networks*, 143:732–750.

[47] Del Moral, P. and Del Moral, P. (2004). *Feynman-Kac formulae*. Springer.

[48] Del Moral, P. and Tugaut, J. (2018). On the stability and the uniform propagation of chaos properties of ensemble Kalman–Bucy filters. *The Annals of Applied Probability*, 28(2):790–850.

[49] Delong, Ł. (2013). *Backward stochastic differential equations with jumps and their actuarial and financial applications*. Springer.

[50] Doucet, A., Johansen, A. M., et al. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3.

[51] Dussault, J. P., Shen, H., and Bandrauk, A. (2007). Penalty algorithms in Hilbert spaces. *Acta Mathematica Sinica, English Series*, 23:229–236.

[52] Elliott, M. and Ginossar, E. (2016). Applications of the Fokker-Planck equation in circuit quantum electrodynamics. *Physical Review A*, 94(4):043840.

[53] Evensen, G. (2003). The Ensemble Kalman Filter: theoretical formulation and practical implementation. *Ocean Dynamics*, 53:343–367.

[54] Farchi, A. and Bocquet, M. (2018). Comparison of local particle filters and new implementations. *Nonlinear Processes in Geophysics*, 25(4):765–807.

[55] Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouvé, A., and Peyré, G. (2019). Interpolating between optimal transport and MMD using Sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690. PMLR.

[56] Fiacco, A. V. and Jones, A. P. (1969). Generalized penalty methods in topological spaces. *SIAM Journal on Applied Mathematics*, 17(5):996–1000.

[57] Fletcher, S. (2017). *Data Assimilation for the Geosciences: From Theory to Application*. Elsevier.

[58] Fu, Z. and Li, Z. (2010). Stochastic equations of non-negative processes with jumps. *Stochastic Processes and their Applications*, 120(3):306–330.

[59] Gao, C., Isaacson, J., and Krause, C. (2020). i-flow: High-dimensional integration and sampling with normalizing flows. *Machine Learning: Science and Technology*, 1(4):045023.

[60] Gardiner, C. (2009). *Stochastic methods*, volume 4. Springer Berlin.

[61] Genevay, A. (2019). *Entropy-regularized optimal transport for machine learning*. PhD thesis, Paris Sciences et Lettres.

[62] Genevay, A., Peyré, G., and Cuturi, M. (2018). Learning generative models with Sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617. PMLR.

[63] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

[64] Gobet, E. (2016). *Monte-Carlo methods and stochastic processes: from linear to non-linear*. CRC Press.

[65] Goldstine, H. H. (2012). *A History of the Calculus of Variations from the 17th through the 19th Century*, volume 5. Springer Science & Business Media.

[66] Gordon, G. and Tibshirani, R. (2012). Karush-Kuhn-Tucker conditions. *Optimization*, 10(725/36):725.

[67] Gottwald, G. A. and Majda, A. J. (2013). A mechanism for catastrophic filter divergence in data assimilation for sparse observation networks. *Nonlinear Processes in Geophysics*, 20(5):705–712.

[68] Guzowska, M., Malinowska, A. B., and Sidi Ammi, M. R. (2015). Calculus of variations on time scales: applications to economic models. *Advances in Difference Equations*, 2015(1):1–15.

[69] Hackbusch, W., Khoromskij, B. N., and Tyrtyshnikov, E. E. (2005). Hierarchical Kronecker tensor-product approximations. In *J. Num. Math.*

[70] Han, J., Jentzen, A., and E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510.

[71] Hinrichs, A., Novak, E., Ullrich, M., and Woźniakowski, H. (2014). The curse of dimensionality for numerical integration of smooth functions. *Mathematics of Computation*, 83(290):2853–2863.

[72] Hirsch, M. W., Smale, S., and Devaney, R. L. (2012). *Differential equations, dynamical systems, and an introduction to chaos*. Academic press.

[73] Holstermann, J. (2023). On the expressive power of neural networks.

[74] Huang, H., Yu, P. S., and Wang, C. (2018). An introduction to image synthesis with generative adversarial nets. *arXiv preprint arXiv:1803.04469*.

[75] Huang, J., Wang, H., and Zhou, T. (2021). An augmented Lagrangian deep learning method for variational problems with essential boundary conditions. *arXiv preprint arXiv:2106.14348*.

[76] Huang, W., Ji, M., Liu, Z., and Yi, Y. (2015). Steady states of Fokker–Planck equations: I. existence. *Journal of Dynamics and Differential Equations*, 27:721–742.

[77] Hutzenthaler, M., Jentzen, A., Kruse, T., et al. (2021). Multilevel Picard iterations for solving smooth semilinear parabolic heat equations. *Partial Differential Equations and Applications*, 2(6):1–31.

[78] Ikeda, N. and Watanabe, S. (2014). *Stochastic differential equations and diffusion processes*. Elsevier.

[79] Ito, K. and Kunisch, K. (1990a). An augmented Lagrangian technique for variational inequalities. *Applied Mathematics and Optimization*, 21(1):223–241.

[80] Ito, K. and Kunisch, K. (1990b). The augmented Lagrangian method for equality and inequality constraints in Hilbert spaces. *Mathematical programming*, 46(1-3):341–360.

[81] Ito, K. and Kunisch, K. (2008). *Lagrange multiplier approach to variational problems and applications*. SIAM.

[82] Ivanov, M. and Shvets, V. (1980). Method of stochastic differential equations for calculating the kinetics of a collision plasma. *Zhurnal Vychislitelnoi Matematiki i Matematicheskoi Fiziki*, 20:682–690.

[83] Jackson, J. D. (1999). Classical electrodynamics 3rd ed john wiley & sons. *Inc., NewYork, NY*, page 5.

[84] Jefferies, B. (2013). *Evolution processes and the Feynman-Kac formula*, volume 353. Springer Science & Business Media.

[85] Jhamtani, H., Gangal, V., Hovy, E., and Nyberg, E. (2017). Shakespearizing modern language using copy-enriched sequence-to-sequence models. *arXiv preprint arXiv:1707.01161*.

[86] Johnson, J. L., Dalhed, H., Greene, J., Grimm, R., Hsieh, Y., Jardin, S., Manickam, J., Okabayashi, M., Storer, R., Todd, A., et al. (1979). Numerical determination of axisymmetric toroidal magnetohydrodynamic equilibria. *Journal of Computational Physics*, 32(2):212–234.

[87] Jordan, R., Kinderlehrer, D., and Otto, F. (1998). The variational formulation of the Fokker–Planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17.

[88] Kakutani, S. (1944a). 131. On Brownian Motions in n-Space. *Proceedings of the Imperial Academy*, 20(9):648–652.

[89] Kakutani, S. (1944b). 143. Two-dimensional Brownian Motion and Harmonic Functions. *Proceedings of the Imperial Academy*, 20(10):706–714.

[90] Kalnay, E. (2003). *Atmospheric modeling, data assimilation, and predictability*. Cambridge University Press.

[91] Kanzow, C., Steck, D., and Wachsmuth, D. (2018). An augmented Lagrangian method for optimization problems in Banach spaces. *SIAM Journal on Control and Optimization*, 56(1):272–291.

[92] Karatzas, I., Karatzas, I., Shreve, S., and Shreve, S. E. (1991). *Brownian motion and stochastic calculus*, volume 113. Springer Science & Business Media.

[93] KARoUI, N. E. and Quenez, M. (1997). Non-linear pricing theory and backward stochastic differential equations. *Financial mathematics*, pages 191–246.

[94] Kelly, D., Law, K., and Stuart, A. (2014). Well-posedness and accuracy of the ensemble Kalman filter In discrete and continuous time. *Nonlinearity*, 27:2579–2603.

[95] Kidger, P. and Lyons, T. (2020). Universal approximation with deep narrow networks. In *Conference on learning theory*, pages 2306–2327. PMLR.

[96] Kilpeläinen, T. (1994). Weighted sobolev spaces and capacity. *Ann. Acad. Sci. Fenn. Ser. AI Math*, 19(1):95–113.

[97] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[98] Kloeden, P. E., Platen, E., Kloeden, P. E., and Platen, E. (1992). *Stochastic differential equations*. Springer.

[99] Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

[100] Kontorovich, V. and Lovtchikova, Z. (2009). Non linear filtering algorithms for chaotic signals: A comparative study. In *2009 2nd International Workshop on Nonlinear Dynamics and Synchronization*, pages 221–227. IEEE.

[101] Kovachki, N., Lanthaler, S., and Mishra, S. (2021). On universal approximation and error bounds for Fourier neural operators. *The Journal of Machine Learning Research*, 22(1):13237–13312.

[102] Kressner, D. and Tobler, C. (2010). Krylov subspace methods for linear systems with tensor product structure. *SIAM journal on matrix analysis and applications*, 31(4):1688–1714.

[103] Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560.

[104] Kuhn, H. W. and Tucker, A. W. (2013). Nonlinear programming. In *Traces and emergence of nonlinear programming*, pages 247–258. Springer.

[105] Laloyaux, P., Balmaseda, M., Dee, D., Mogensen, K., and Janssen, P. (2016). A coupled data assimilation system for climate reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 142(694):65–78.

[106] Law, K., Stuart, A., and Zygalakis, K. (2015). *Data Assimilation*. Springer.

[107] Law, K. J., Sanz-Alonso, D., Shukla, A., and Stuart, A. (2016). Filter accuracy for the Lorenz 96 model: Fixed versus adaptive observation operators. *Physica D: Nonlinear Phenomena*, 325:1–13.

[108] Lelievre, T. and Stoltz, G. (2016). Partial differential equations and stochastic methods in molecular dynamics. *Acta Numerica*, 25:681–880.

[109] Leobacher, G. and Pillichshammer, F. (2014). *Introduction to quasi-Monte Carlo integration and applications*. Springer.

[110] Li, H., Fang, C., and Lin, Z. (2017). Convergence rates analysis of the quadratic penalty method and its applications to decentralized distributed optimization. *arXiv preprint arXiv:1711.10802*.

[111] Li, X.-M. and Scheutzow, M. (2011). Lack of strong completeness for stochastic flows.

[112] Li, Z. and Mytnik, L. (2011). Strong solutions for stochastic differential equations with jumps. In *Annales de l'IHP Probabilités et statistiques*, volume 47, pages 1055–1067.

[113] Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141.

[114] Lorenz, E. N. (1995). Predictability: a problem partly solved. In *Seminar on Predictability I*, pages 1–18. ECMWF, Reading UK.

[115] Lu, Y. and Lu, J. (2020). A universal approximation theorem of deep neural networks for expressing probability distributions. *Advances in neural information processing systems*, 33:3094–3105.

[116] Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. (2017). The expressive power of neural networks: A view from the width. *Advances in neural information processing systems*, 30.

[117] Mahan, S., King, E. J., and Cloninger, A. (2021). Nonclosedness of sets of neural networks in Sobolev spaces. *Neural Networks*, 137:85–96.

[118] Malliavin, P. (2006). *Stochastic calculus of variations in mathematical finance*. Springer.

[119] Mandal, P. (2024). Learning solutions to some toy constrained optimization problems in infinite dimensional Hilbert spaces. *arXiv preprint arXiv:2401.01306*.

[120] Mandal, P. and Apte, A. (2023). Learning zeros of Fokker-Planck operators. *arXiv preprint arXiv:2306.07068*.

[121] Mandal, P. and Apte, A. (2024). Solving Fokker-Planck equations using the zeros of Fokker-Planck operators and the Feynman-Kac formula. *arXiv preprint arXiv:2401.01292*.

[122] Mandal, P., Roy, S. K., and Apte, A. (2021). Stability of nonlinear filters - numerical explorations of particle and ensemble Kalman filters. In *2021 Seventh Indian Control Conference (ICC)*, pages 307–312. IEEE.

[123] Mandal, P., Roy, S. K., and Apte, A. (2023). Probing robustness of nonlinear filter stability numerically using Sinkhorn divergence. *Physica D: Nonlinear Phenomena*, 451:133765.

[124] Masud, A. and Bergman, L. A. (2005). Application of multi-scale finite element methods to the solution of the Fokker–Planck equation. *Computer Methods in Applied Mechanics and Engineering*, 194(12-16):1513–1526.

[125] Mensch, Arthur and Peyré, Gabriel (2020). Online sinkhorn: Optimal transport distances from sample streams. *Advances in Neural Information Processing Systems*, 33:1657–1667.

[126] Meusnier, J. B. (1785). Mémoire sur la courbure des surfaces. *Mem des savan etrangers*, 10(1776):477–510.

[127] Muller, M. E. (1956). Some continuous Monte Carlo methods for the Dirichlet problem. *The Annals of Mathematical Statistics*, pages 569–589.

[128] Náprstek, J. and Král, R. (2014). Finite element method analysis of Fokker–Planck equation in stationary and evolutionary versions. *Advances in Engineering Software*, 72:28–38.

[129] Nocedal, J. and Wright, S. J. (2006). *Numerical optimization*. Springer.

[130] Øksendal, B. (1997). An introduction to Malliavin calculus with applications to economics.

[131] Øksendal, B. and Øksendal, B. (2003). *Stochastic differential equations*. Springer.

[132] Oljača, L., Kuna, T., and Bröcker, J. (2021). Exponential stability and asymptotic properties of the optimal filter for signals with deterministic hyperbolic dynamics. *arXiv preprint arXiv:2103.01190*.

[133] Ott, E. (1981). Strange attractors and chaotic motions of dynamical systems. *Reviews of Modern Physics*, 53(4):655.

[134] Palomo, M. (2021). New insight into the origins of the calculus war. *Annals of Science*, 78(1):22–40.

[135] Papamakarios, G. (2019). Neural density estimation and likelihood-free inference. *arXiv preprint arXiv:1910.13233*.

[136] Peddie, J. (2023). *The History of the GPU-New Developments*. Springer Nature.

[137] Pham, H. (2015). Feynman-Kac representation of fully nonlinear PDEs and applications. *Acta Mathematica Vietnamica*, 40:255–269.

[138] Pichler, L., Masud, A., and Bergman, L. A. (2013). Numerical solution of the Fokker–Planck equation by finite difference and finite element methods—a comparative study. In *Computational Methods in Stochastic Dynamics*, pages 69–85. Springer.

[139] Pinkus, A. (1999). Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195.

[140] Polyak, B. T. (1971). The convergence rate of the penalty function method. *USSR Computational Mathematics and Mathematical Physics*, 11(1):1–12.

[141] Quapp, W. (2008). Chemical reaction paths and calculus of variations. *Theoretical Chemistry Accounts*, 121:227–237.

[142] Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. (2016). Survey of expressivity in deep neural networks. *arXiv preprint arXiv:1611.08083*.

[143] Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. (2017). On the expressive power of deep neural networks. In *international conference on machine learning*, pages 2847–2854. PMLR.

[144] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707.

[145] Ranocha, H. (2019). Mimetic properties of difference operators: product and chain rules as for functions of bounded variation and entropy stability of second derivatives. *BIT Numerical Mathematics*, 59(2):547–563.

[146] Reddy, A. S. and Apte, A. (2021). Stability of non-linear filter for deterministic dynamics. *Foundations of Data Science*, 3(3):647–675.

[147] Reddy, A. S., Apte, A., and Vadlamani, S. (2020). Asymptotic properties of linear filter for noise free dynamical system. *Systems & Control Letters*, 139:104676.

[148] Reich, S. and Cotter, C. (2015). *Probabilistic forecasting and Bayesian data assimilation*. Cambridge University Press.

[149] Risken, H. and Risken, H. (1996). *Fokker-planck equation*. Springer.

[150] Rojo, A. and Bloch, A. (2018). *The principle of least action: History and physics*. Cambridge University Press.

[151] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[152] Särkkä, S. (2013). *Bayesian filtering and smoothing*, volume 3. Cambridge University Press.

[153] Särkkä, S. and Svensson, L. (2023). *Bayesian filtering and smoothing*, volume 17. Cambridge university press.

[154] Sawhney, R., Miller, B., Gkioulekas, I., and Crane, K. (2023). Walk on stars: A grid-free monte carlo method for pdes with neumann boundary conditions. *arXiv preprint arXiv:2302.11815*.

[155] Schäfer, A. M. and Zimmermann, H. G. (2006). Recurrent neural networks are universal approximators. In *Artificial Neural Networks–ICANN 2006: 16th International Conference, Athens, Greece, September 10-14, 2006. Proceedings, Part I 16*, pages 632–640. Springer.

[156] Schoen, R. and Yau, S.-T. (1979). On the proof of the positive mass conjecture in general relativity. *Communications in Mathematical Physics*, 65:45–76.

[157] Sepehrian, B. and Radpoor, M. K. (2015). Numerical solution of non-linear Fokker–Planck equation using finite differences method and the cubic spline functions. *Applied mathematics and computation*, 262:187–190.

[158] Sherstinsky, A. (2020). Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306.

[159] Sirignano, J. and Spiliopoulos, K. (2018). Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364.

[Smitha and Hattorib] Smitha, S. G. L. and Hattorib, Y. Axisymmetric magnetic vortices with swirl.

[161] Strauss, R. and Effenberger, F. (2017). A hitch-hiker's guide to stochastic differential equations. *Space Science Reviews*, 212(1):151–192.

[162] Struik, D. J. (1961). *Lectures on classical differential geometry*. Courier Corporation.

[163] Sun, Y. and Kumar, M. (2014). Numerical solution of high dimensional stationary Fokker–Planck equations via tensor decomposition and Chebyshev spectral differentiation. *Computers & Mathematics with Applications*, 67(10):1960–1977.

[164] Thomas, R. (1999). Deterministic chaos seen in terms of feedback circuits: Analysis, synthesis," labyrinth chaos". *International Journal of Bifurcation and Chaos*, 9(10):1889–1905.

[165] Uria, B., Murray, I., and Larochelle, H. (2013). Rnade: The real-valued neural autoregressive density-estimator. *Advances in Neural Information Processing Systems*, 26.

[166] van Handel, R. (2008). Hidden Markov models. *Unpublished lecture notes*.

[167] van Kekem, D. (2018). *Dynamics of the Lorenz-96 model: Bifurcations, symmetries and waves*. PhD thesis, University of Groningen.

[168] Van Leeuwen, P. J., Cheng, Y., and Reich, S. (2015). *Nonlinear data assimilation*. Springer.

[169] Vennerød, C. B., Kjærran, A., and Bugge, E. S. (2021). Long short-term memory RNN. *arXiv preprint arXiv:2105.06756*.

[170] Wang, Keyou and Crow, Mariesa L (2012). Fokker-Planck equation application to analysis of a simplified wind turbine model. In *2012 North American Power Symposium (NAPS)*, pages 1–5. IEEE.

[171] Weinberg, S. (1972). Gravitation and cosmology: principles and applications of the general theory of relativity.

[172] Whiteley, N. (2013). Stability properties of some particle filters. *The Annals of Applied Probability*, 23(6):2500–2537.

[173] Whitney, J. C. (1970). Finite difference methods for the Fokker-Planck equation. *Journal of Computational Physics*, 6(3):483–509.

[174] Xi, F. and Zhu, C. (2019). Jump type stochastic differential equations with non-Lipschitz coefficients: non-confluence, Feller and strong Feller properties, and exponential ergodicity. *Journal of Differential Equations*, 266(8):4668–4711.

[175] Xu, T. and Fitzpatrick, R. (2019). Vacuum solution for Solov'ev's equilibrium configuration in tokamaks. *Nuclear Fusion*, 59(6):064002.

[176] Xu, Y., Zhang, H., Li, Y., Zhou, K., Liu, Q., and Kurths, J. (2020). Solving Fokker-Planck equation using deep learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(1):013133.

[177] Yamada, T. and Watanabe, S. (1971). On the uniqueness of solutions of stochastic differential equations. *Journal of Mathematics of Kyoto University*, 11(1):155–167.

[178] Yeong, H. C., Beeson, R. T., Namachchivaya, N., and Perkowski, N. (2020). Particle filters with nudging in multiscale chaotic systems: With application to the Lorenz'96 atmospheric model. *Journal of Nonlinear Science*, 30(4):1519–1552.

[179] Yong, J. and Zhou, X. Y. (1999). *Stochastic controls: Hamiltonian systems and HJB equations*, volume 43. Springer Science & Business Media.

[180] Zhai, J., Dobson, M., and Li, Y. (2022). A deep learning method for solving Fokker-Planck equations. In *Mathematical and Scientific Machine Learning*, pages 568–597. PMLR.

List of works

1. Mandal, P., Roy, S. K., and Apte, A. (2021). Stability of nonlinear filters - numerical explorations of particle and ensemble Kalman filters. In *2021 Seventh Indian Control Conference (ICC)*, pages 307–312. IEEE

2. Mandal, P., Roy, S. K., and Apte, A. (2023). Probing robustness of nonlinear filter stability numerically using Sinkhorn divergence. *Physica D: Nonlinear Phenomena*, 451:133765

3. Mandal, P. and Apte, A. (2023). Learning zeros of Fokker-Planck operators. *arXiv preprint arXiv:2306.07068*

4. Mandal, P. and Apte, A. (2024). Solving Fokker-Planck equations using the zeros of Fokker-Planck operators and the Feynman-Kac formula. *arXiv preprint arXiv:2401.01292*

5. Mandal, P. (2024). Learning solutions to some toy constrained optimization problems in infinite dimensional Hilbert spaces. *arXiv preprint arXiv:2401.01306*