TECHNISCHE UNIVERSITÄT MÜNCHEN

TUM School of Engineering and Design

Physics-aware, probabilistic machine learning in the Small Data regime

Sebastian Johannes Kaltenbach

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz:	Prof. Dr.-Ing. Nikolaus Adams
Prüfer*innen der Dissertation:	1. Prof. Phaedon-Stelios Koutsourelakis, Ph.D. 2. Prof. Paris Perdikaris, Ph.D. 3. Prof. Petros Koumoutsakos, Ph.D.

Die Dissertation wurde am 23.09.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 27.02.2023 angenommen.

Zusammenfassung

Das Beschreiben und Berechnen von hochdimensionalen, nichtlinearen Systemen ist eine der zentralen Herausforderungen in den Ingenieurwissenschaften und der numerischen Physik. Wir stellen neuartige Physik-inspirierte Modelle des maschinellen Lernens vor, die sowohl auf physikalischem Wissen als auch eine kleine Anzahl von Daten basieren und nach einer anfänglichen Trainingsphase in der Lage sind, die anfangs erwähnten hochdimensionalen Systeme effizient zu lösen. Das zentrale Element dieses Ansatzes ist das Einbeziehen von induktiver Verzerrung im Gegensatz zu rein statistischen Algorithmen, die im Allgemeinen eine sehr große Menge an teuren Trainingsdaten benötigen und deren Interpretierbarkeit in der Regel nicht gegeben ist.

Wir stellen das Konzept der virtuellen Beobachtungen vor, um physikalische Randbedingungen in unsere Modelle einzubeziehen, und zeigen, dass das Hinzufügen dieser virtuellen Beobachtungen extrapolative Vorhersagen ermöglicht. Virtuelle Beobachtungen können problemlos in jedes probabilistische Zustandsraummodell integriert werden und wir können sehr genaue Modelle reduzierter Ordnung für verschiedene dynamische Systeme berechnen.

Zusätzlich zeigen wir, dass es für Modellreduktionsprobleme in der numerischen Physik von Vorteil sein kann, die Dynamik der Variablen reduzierter Ordnung per Konstruktion auf eine stabile Dynamik einzuschränken. Dies basiert auf der Beobachtung, dass die meisten physikalische System einen Gleichgewichtszustand erreichen, obwohl sie vor allem zu Beginn hochgradig nicht-stationär sein können. Der vorgeschlagene Algorithmus nutzt eine flexible A-priori Verteilung im komplexen Zahlenraum, welche die Identifikation von verborgenen langsamen Prozessen ermöglicht und die Stabilität der gelernten Dynamik garantiert. Aufgrund dieser induktiven Verzerrung können wir das vollständig probabbilistische Modell mit einer sehr geringen Menge an Daten trainieren. Wir demonstrieren die Genauigkeit des Verfahrens am Beispiel mehrstufiger physikalischer Systeme, indem wir probabilistische, langfristige Vorhersagen für nicht in den Trainingsdaten enthaltene Phänomene erzeugen.

Schließlich erweitern wir die sogenannten Deep Operator Network (DeepONet) Architektur, indem wir diese invertierbar machen, um sowohl das Vorwärtsproblem als auch das inverse Problem mit nur einem Neuronalen Netzwerk lösen zu können. Der resultierende Algorithmus des maschinellen Lernens ist insbesondere geeignet für physikalische Systeme, da wir bei diesen im Allgemeinen sowohl am Vorwärtsoperator als auch am Inversen Operator interessiert sind. Das resultierende DeepONet kann auch für Bayessche Inverse Probleme verwendet werden, wobei wir eine approximative A-posteriori Verteilung ohne jegliche kostenintensive Markow-Chain-Monte-Carlo (MCMC) Verfahren erzeugen können.

Abstract

Solving high-dimensional, nonlinear systems is a key challenge in engineering and computational physics. We propose novel physics-aware machine learning models that rely both on physical knowledge as well as a small amount of data and are, after an initial training phase, able to solve these aforementioned high-dimensional systems efficiently. The key characteristic of this approach is incorporating inductive bias in contrast to purely statistical frameworks that, in general, lack interpretability and rely on large amounts of expensive simulation data.

We propose the concept of virtual observables to incorporate physical constraints and show that the addition of virtual observables enables extrapolative predictions. Virtual observables can be seamlessly integrated into any probabilistic state space model, and we are able to obtain very accurate reduced order models for different dynamical systems.

Moreover, we show that for model order reduction problems in computational physics, it can be beneficial to restrict the dynamics of the reduced order variables to dynamics that are inherently stable. This is based on the observation that most physical systems reach equilibrium in the long term despite being highly non-stationary. The proposed framework uses a flexible prior on the complex plane that facilitates the discovery of latent slow processes and ensures the long-term stability of the learned dynamics. Due to the added inductive bias, we are able to train the fully probabilistic model in the small data regime, and we demonstrate its accuracy in multiscale physical systems of particle dynamics where probabilistic, long-term predictions of phenomena not contained in the training data are produced.

Finally, we extend the Deep Operator Network (DeepONet) architecture by making it invertible to solve both forward and inverse problems with one neural network. The resulting machine learning algorithm is very suitable for physical problems as, in most cases, we are simultaneously interested in forward and inverse operators. The obtained invertible DeepONet can also be used for Bayesian inverse problems, for which we can construct an approximative posterior without the need for any costly Markov Chain Monte Carlo (MCMC) sampling.

Acknowledgements

During my time as a PhD candidate, many people supported me and helped me to grow both personally and professionally:

First and foremost, I am deeply indebted to my PhD advisor Prof. Phaedon-Stelios Koutsourelakis for his support of my research. Your perspective and insight have always been very helpful for me. I have always enjoyed our discussions and have benefited a lot from them. Besides always taking the time to share some advice regarding current research projects, I also want to thank you for your advice regarding navigating academia. I consider myself fortunate to have been a part of your research group.
I would like to express my deep gratitude to Prof. Paris Perdikaris for welcoming me to Philadelphia for a research stay and for the exciting time I was able to spend with you and your team at the University of Pennsylvania. During my time in Philadelphia and also during the continued research afterward, I learned a lot from you and acquired new skills.
I would like to extend my sincere thanks to Prof. Petros Koumoutsakos for agreeing to be a member of the examination committee of this thesis. Your research is inspiring, and I am looking forward to working with you.
Moreover, I want to thank Prof. Nikolaus Adams, who is voluntarily chairing the examination committee.

Furthermore, I would like to thank all my friends and current or former colleagues at the Professorship of Data-driven Materials Modeling: Maximilian Rixner, Atul Agrawal, Constantin Grigo, Markus Schöberl, Jonas Nitzler, and Luca Berardocco. I have benefited from the interactions with all of you and enjoyed collective lunch breaks and other activities a lot.
Special thanks also to our secretary Sigrid Harnauer, who helped me with all the administrative tasks.
Moreover, I want to thank Shyam Sankaran, Mohamed Aziz Bhouri, Leanordo Ferreira Guilhoto, and George Kissas for welcoming me to Philadelphia and helping me to navigate Penn during my stay there.

Finally, I would like to thank all my friends and my parents for their support throughout the last years. This thesis would not have been possible without you.

Contents

1. Introduction

This thesis deals with the development of probabilistic machine learning algorithms for physical systems that account for physical knowledge as inductive bias and only rely on a small amount of data. As physical data is, in general, high-dimensional, the presented algorithms perform a model order reduction in order to obtain a surrogate model that can be used for efficient predictions. Due to the probabilistic nature of physical systems as well as model uncertainties, we follow a Bayesian approach to account for the uncertainty during model training and prediction.

This chapter outlines the challenges we want to overcome with our novel physics-aware machine learning methods and presents the contribution of this thesis toward solving these challenges.

1.1. Motivation and background

Most machine learning algorithms rely on very large datasets that are easy to obtain or readily available (Big Data) and limited prior knowledge. While this is, in general, a good strategy for applications where the aforementioned vast amount of data is available and has been shown to lead to very promising results (Saharia et al., 2022; Brock et al., 2021), there are different kinds of applications for which a sufficiently large training data set is hard to obtain and about which we have more prior knowledge available instead. According to the author, three main challenges have to be overcome to apply machine learning algorithms successfully to these kinds of problems:

- We have to account for physical knowledge as inductive bias. Predictions are not allowed to contradict known physical constraints, and it would be wasteful to ignore already existing knowledge about the problem.
- We have to adopt a probabilistic approach to capture both the uncertainty due to model error and randomness in the training data.
- Small Data: Data is, in general, expensive and sparse, especially if we have to run high-dimensional simulations to generate our datasets. Therefore, we have to develop algorithms that only rely on a very small amount of data.

In the following subsections, we take a closer look at each of these challenges.

1.1.1. Physics-aware modeling

Whereas machine learning models that rely only on training data can be a reasonable approach for a Big Data problem, the addition of inductive bias has proven to be very useful (Battaglia et al., 2018). For example, even for Big Data problems such as Image Recognition,

specifically designed architectures such as Convolutional Neural Networks (CNN), that are shift-invariant, have shown to be advantageous (LeCun and Bengio, 1995).

The addition of inductive bias is even more relevant for physical systems as training data is limited, and there usually is some knowledge about the system a priori available. This knowledge can have the form of constraints and invariants or can specify characteristics of the system [1]. Without adding constraints such as mass or energy conservation, the machine learning model could violate these fundamental physical laws during predictions. In general, inductive bias is not only able to prevent predictions from violating known physical constraints and invariants but can also lead to better extrapolative predictions. Moreover, incorporating inductive bias and thus physical information can allow us to work with less data as the machine learning model is already a priori better adapted to the targeted application.

Based on the observation that most physical systems can be highly non-stationary but converge to an equilibrium in the long term, we moreover can impose some restrictions when learning the dynamics of such a system. If we use a priori long-term stable dynamics within our machine learning model, we can also guarantee stability during predictions. This is especially useful for learning a reduced-order description for a high-dimensional system. In general, the knowledge about the reduced order system can be minimal a priori, so only basic physical properties such as the mentioned long-term stability can be used as adequate inductive bias.

During predictions, we are for most physical systems interested in applying both a forward as well as an inverse operator to new initial conditions, i.e. we are interested in finding the solution/output of the system given the system's inputs and simultaneously in reconstructing the inputs if the system's solution/output is given. However, most machine learning models are only designed to learn one of the mentioned operators. It is, therefore, advantageous to construct machine learning models that are invertible and can thus learn both of the operators. This allows us also to make the most of our training data as we can use them for both the forward and the inverse operator.

1.1.2. Probabilistic modeling

Probabilistic models are, in my opinion, necessary in machine learning due to two main reasons:

- In nature, we are usually confronted with randomness across all different disciplines. For instance, in medicine, it is usually only possible to make probabilistic predictions about the development and spread of cancer and subsequently the survival chances (Tomasetti et al., 2017). In physics, the state of an atom can be described only in a probabilistic sense (Flügge, 2012), and in chemistry, the reactions of molecules are stochastic processes (Van Kampen, 1992). Therefore, we have to capture inherent randomness and are thus in need of a probabilistic modeling approach. The uncertainty

[1] A physical system can, for instance, have a dissipative character that directly affects the dynamics of the system.

described here is also known as data uncertainty or aleatoric uncertainty (Hüllermeier and Waegeman, 2021). Uncertainties in the data due to imperfect measurements belong to this category as well.

- Moreover, the models we propose are only approximations of the true reality. We may not have enough training data to fit the model accurately and may not yet know the underlying physical processes, model structure, and model parameter values well enough Gal et al. (2022). It is therefore advantageous to quantify the model error by using a probabilistic approach and thus quantify this so-called model uncertainty or epistemic uncertainty. This uncertainty is, in general, reduced when more data becomes available.

We note that while a probabilistic model allows capturing all the aforementioned uncertainties, it is difficult to actually attribute the uncertainty to the different parts. There are different approaches available regarding this topic (Kiureghian and Ditlevsen, 2009; Wang et al., 2019), but this is beyond the focus of the present work.

1.1.3. Small Data

To generate the necessary training data for our machine learning models, we have to run, in general, expensive simulations or conduct equally expensive experiments. Therefore, a machine learning algorithm that relies only on a small amount of training data is advantageous. Within this work, we train our machine learning models with less than 1000 (for some cases even less than 100) given training data, i.e. input/output pairs generated by running a simulation or conducting an experiment. The dimension of each individual data point can be high which poses an additional challenge for our algorithms due to the 'curse of dimensionality' (Bellman, 2010). Moreover, the training data available can be sparse and our algorithm, therefore, needs to be capable of being trained on sparse data.

To compensate for the lack of data, we use physical knowledge as inductive bias. Figure 1.1 shows the trade-off between data and knowledge. A physical system can either be computed with the help of machine learning and a lot of data or with a more classical simulation approach such as the Finite-Elements-Method (FEM) or Finite-Difference-Method (FD) based completely on the physical knowledge which needs to be formulated in form of the physical equations and a very small amount of data, i.e. the boundary and initial conditions of the system. We target an intermediate area here; we want to use a machine learning approach that makes use of a small amount of training data and physical knowledge simultaneously.

1.2. Related work

Data-driven surrogates
As physical systems are, in general, high-dimensional and thus their solution requires significant computational resources, the identification of reduced order models and surrogates for these systems is a crucial task. Besides well-known methods such as Principal Component Analysis (PCA) (Abdi and Williams, 2010) and Proper Orthogonal Decomposition

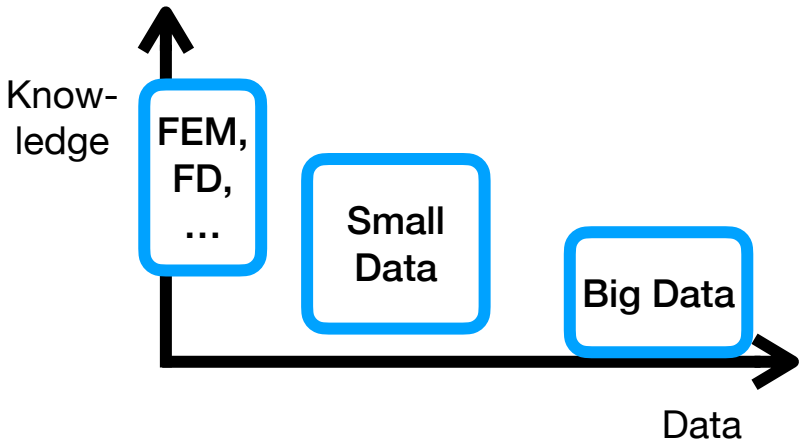


Figure 1.1: Solution approaches based on available data and knowledge

(POD) (Berkooz et al., 1993), approaches based on recent advances in data-driven learning (Ghahramani, 2015; Koutsourelakis et al., 2016; Bhattacharya et al., 2021) have shown to be promising and attracted a lot of attention in the last years. These models are generally trained offline and are then used for various predictive tasks. Especially for multiscale modeling (Koutsourelakis and Bilionis, 2011; Givon et al., 2004), these approaches allow substituting the smallest scales in the computations with an efficient surrogate. Nevertheless, these models can even predict the full fine-grained, i.e. high-dimensional system, if a coarse-to-fine map (Schöberl et al., 2017), i.e. a map from the reduced order description to the high-dimensional system, is used.

Since the availability of automatic differentiation frameworks (Abadi et al., 2015; Bradbury et al., 2018; Paszke et al., 2017), surrogates are often represented by deep neural networks as, under certain conditions, neural networks are universal approximators as shown in Hornik et al. (1989). The Physics-informed Neural Network (PINN) architecture (Lagaris et al., 1998; Raissi et al., 2019) as well as the Deep Operator Network (DeepONet) architecture (Lu et al., 2021; Wang et al., 2021) are prominent examples. The latter also takes input parameters of the underlying Partial Differential Equation (PDE) as an input to the neural network and can generate predictions for different parameter inputs. Other notable approaches are the Fourier Neural Network (Li et al., 2021), which builds upon an integral kernel in Fourier space, and the Learning Operators with Coupled Attention (LOCA) framework (Kissas et al., 2022) that makes use of the attention mechanism. The attention mechanism emphasizes some parts of the input data and diminishes other parts and originated in Natural Language processing (Vaswani et al., 2017).

To address the specific challenges of multiscale systems, methods such as Learning Effective Dynamics (LED) (Vlachas et al., 2022) not only identify a surrogate but employ lifting and restriction operators to switch between scales.

Discovery of dynamics from data
The discovery of dynamics from data is a central task in learning non-stationary physical

systems. Approaches based on the Koopman-Operator theory (Koopman, 1931; Klus et al., 2018) try to identify the most suitable linear representation of the system's dynamics and the associated transformation to a set of linear variables. Due to the restriction of the dynamics to a linear space, most Koopman-based methods (Gin et al., 2021; Lusch et al., 2018; Champion et al., 2019; Lee and Carlberg, 2020; Pan and Duraisamy, 2020) require a large set of linear variables in order to represent a complex dynamic.

Other methods that try to identify a non-linear dynamic can counteract this disadvantage. Such methods can, for instance, be based on the Mori-Zwanzig formalism (Mori, 1965; Zwanzig, 1973). However, Mori-Zwanzig based approaches are restricted to predefined quantities of interest (Kondrashov et al., 2015; Chorin and Stinis, 2007). Other approaches are based on the Sparse Identification of Nonlinear Dynamics (SINDy) algorithm (Brunton et al., 2016; Kaheman et al., 2020) which in general requires derivatives as input data. Approaches based on employing neural networks for the dynamics (Chen et al., 2018; Li et al., 2020) are the most flexible ones but need a large amount of data.

Infusing domain knowledge as inductive bias
Infusing inductive bias into machine learning methods and especially Deep Learning frameworks is a challenging task. A popular approach to employing inductive bias is using physical laws as a regularization term or for augmenting the the loss function (Raissi et al., 2019; Zhu et al., 2019; Rixner and Koutsourelakis, 2021; Wang et al., 2021). The Deep Operator Network (DeepONet) (Lu et al., 2021) for instance can be trained without data by employing a physics-informed loss term (Wang et al., 2021). The performance can subsequently then be further improved by modifying the neural network architecture (Wang et al., 2022d) but also by taking causality into account (Wang et al., 2022c).

Other formulations use neural network architectures that are directly inspired by physics such as the Hamiltonian and Lagrangian Neural Networks (Greydanus et al., 2019; Toth et al., 2020; Lutter et al., 2019; Cranmer et al., 2020) whose architectures are designed such that they incorporate the fundamental laws of Hamiltonian or Lagrangian mechanics. These approaches are, therefore, suitable for physical systems that conserve energy.

1.3. Contribution of this dissertation

This thesis contributes to solving the challenges presented in section 1.1. It consists of four papers which are all addressing different aspects of the mentioned challenges and which are presented below:

Paper A: Incorporating physical constraints in a deep probabilistic machine learning framework for coarse-graining dynamical systems (Kaltenbach and Koutsourelakis, 2020a) (see Section 3.1 and Appendix A):
This paper introduces the concept of virtual observables and thus a tool that can be seamlessly integrated into any probabilistic state-space model to account for physical constraints/invariants or residuals. We apply the novel concept of virtual observables to different coarse-graining problems for dynamical systems and are able to generate good extrapolative

predictions despite working with small data. To identify the dynamics of the coarse-grained system, we rely on a priori defined feature functions combined with sparsity enforcing priors. As we are employing a coarse-to-fine map in our coarse-graining scheme, we can reconstruct the full fine-grained system and do not have to choose the quantities of interest a priori.
With regards to the challenges mentioned in Section 1.1, this paper addresses all three of them. We employ a fully probabilistic framework, account for physical constraints with virtual observables, and train our models in the Small Data regime.

Paper B: Physics-aware, deep probabilistic modeling of multiscale dynamics in the Small Data regime (Kaltenbach and Koutsourelakis, 2021a) (see Section 3.2 and Appendix B):
This paper extends the proposed model order reduction scheme in combination with virtual observables as presented in Paper A to cases where the dynamics of the coarse-grained system are no longer a priori defined by feature functions. Instead, a deep neural network is employed. Despite having to find the optimal set of coefficients for two neural networks during training, e.g. the coarse-to-fine map as well as the aforementioned coarse-grained transition map, the usage of virtual observables still allows us to work in the Small Data regime.
This paper builds upon the results of Paper A and therefore addresses all three challenges mentioned in Section 1.1.

Paper C: Physics-aware, probabilistic model order reduction with guaranteed stability (Kaltenbach and Koutsourelakis, 2021b) (see Section 3.3 and Appendix C):
This paper introduces a physics-aware model order reduction scheme whose stability is guaranteed as the dynamics of the reduced order variables are a priori restricted. In particular, we employ a discretized Ornstein-Uhlenbeck process in the complex plane and enforce a negative real part of the eigenvalues of the system. This is based on the observation that the physical systems of interest converge to some kind of equilibrium. Moreover, we can rate the different reduced order variables according to their slowness. The method is applied to systems of moving particles, correctly identifies the stable state, and can generate predictions for unseen initial conditions.
With regards to the challenges mentioned in Section 1.1, this paper addresses all three of them. However, the main focus of this paper is incorporating inductive bias into a fully probabilistic model order reduction framework. We do not use the concept of virtual observables in this paper but guarantee the stability of the dynamics of the reduced order variables as discussed above. The physics-aware architecture employed allows us to train the model in the Small Data regime.

Paper D: Semi-supervised Invertible Neural Operators for Bayesian Inverse Problems (Kaltenbach et al., 2022) (see Section 3.4 and Appendix D):
This paper introduces the invertible DeepONet. We build upon the DeepONet architecture as introduced by Lu et al. (2021) and combine it with the realNVP network in order to achieve invertibility. This is especially useful for physical systems as we are now able to represent both a forward and an inverse operator with the same neural network. The obtained invertible DeepONet can also easily be used for Bayesian inverse problems as we additionally introduce a way to obtain an approximate posterior without any costly MCMC sampling .

With regards to the challenges mentioned in Section 1.1, this paper addresses mainly the first and third challenges. The invertible DeepONet is not a probabilistic machine learning framework but can readily be used as surrogate for Bayesian inverse problems. However, we can train the invertible DeepONet with only a small amount of (labeled) training data. The framework's architecture was designed with the needs of physical systems in mind as we are, in general, interested in both the forward and the inverse operator.

Additional publications
During my time as PhD candidate, I authored two more papers that at least partly also discuss the challenges mentioned in the section before but are not part of this thesis. One paper takes another look at model order reduction with guaranteed stability (Kaltenbach and Koutsourelakis, 2020b), whereas the other shows that even physics-enhanced neural network architectures such as Hamiltonian Neural Networks can be trained with less data if more physical information is given instead (Eichelsdörfer et al., 2021). The mentioned approach will shortly be discussed in the methodology section, together with the physics-enhanced neural networks.

1.4. Outline of this dissertation

The remainder of this thesis is structured as follows. Chapter 2 introduces the methodology basics of this thesis. We discuss Bayesian probability theory and probabilistic machine learning in general before taking a closer look at different methods to perform (approximate) inference. Moreover, the basics of Deep Learning are introduced within this chapter.
Chapter 3 summarizes the four publications and also outlines my contribution to each of them.
The thesis is subsequently concluded in chapter 4 where the findings of the publications mentioned above are discussed, and an outlook is given.

2. Methodology

2.1. Bayesian probability theory

"Probability is orderly opinion ... inference from data is nothing other than the revision of such opinion in the light of relevant new information." (Edwards et al., 1963)

The Bayesian viewpoint interprets probability as a belief of certainty or reasonable expectation about given information. This resembles a sharp contrast to the frequentist viewpoint, which interprets the probability of an event as the relative frequency of this event in a large number of trials. In this thesis, we will follow the Bayesian interpretation as this interpretation allows us to describe the uncertainty of model parameters and its change given new data. The quote above, taken from a paper about Bayesian inference for psychological research by Edwards et al. (1963), provides a very good summary of this key element of Bayesian probability theory.

Bayes' theorem is the central mathematical theorem of Bayesian probability. Introduced by Thomas Bayes (Bayes, 1763) and later generalized by Pierre-Simon Laplace (de Laplace, 1820), it provides a mathematical description of how the probability of an event/parameter is changed in the light of new information. In more detail, given random parameters $\boldsymbol{\theta}$ and data $\boldsymbol{\mathcal{D}}$ we can derive the conditional probability distribution of the random parameters given the data $p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$ using Bayes' theorem:

$$p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) = \frac{p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{\mathcal{D}})} \tag{2.1}$$

The computed conditional distribution is also called posterior or a-posteriori-distribution because it represents the probability distribution of the parameters after we have seen the data. In contrast, we call $p(\boldsymbol{\theta})$ the prior or a-priori-distribution, i.e. the probability distribution of the parameters before we have seen the data. The conditional distribution of the data given the parameters $p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta})$ is called likelihood and the probability of the data $p(\boldsymbol{\mathcal{D}})$ is called evidence. The latter is rarely relevant in practice as it only acts as a normalization constant and can often be omitted.

The choice of the prior is a crucial step when applying Bayes' theorem. We can choose a very broad and thus uninformative prior or a prior that already constrains our parameters to a specific interval and is thus informative. Especially when working with physical systems, the latter one is generally chosen as we often have some information about the parameters, such as that some parameters have to be larger than zero or physical laws restrict a parameter to a specific interval. If this information is a priori available, it should also be incorporated by choosing a suitable prior. In case we are dealing with a high-dimensional parameter

set and are assuming that only a few parameter values will take values other than zero, sparsity-enforcing a-priori-distributions should be chosen. These priors are heavy-tailed and counteract a possible overfitting by advocating a sparse representation of the parameters. A popular choice is the Automatic Relevance Determination (ARD) framework by Mackay (1995).

2.2. Probabilistic machine learning

Most machine learning algorithms do not consider parameters or other quantities as random variables but instead only as deterministic variables. While this choice can lead to an algorithm that requires less computational resources and is easier to implement, we argue that it neglects important information, and the predictions of such an algorithm are less informative as they do not provide any information regarding the uncertainty of the computed solution.

Therefore, we treat all unknown quantities of interest and parameters as random variables. The reason for this choice is also motivated in more detail in section 1.1.2. In the following sections, we introduce some of the key methods that are required to do probabilistic machine learning. However, the methodology part of this thesis is not meant to provide an extensive description of probabilistic machine learning methods. The interested reader is referred to the books by Christopher Bishop (Bishop and Nasrabadi, 2006) and Kevin Murphy (Murphy, 2012) that provide a detailed introduction to the topic and discuss all relevant methods.

The key element of almost all probabilistic machine learning algorithms is the Bayesian posterior. This posterior has to be derived and subsequently quantities of interest based on this posterior have to be computed afterward. As the posterior can be very high-dimensional and multimodal and is, in general, analytically intractable, the last step can be very challenging. Here, multiple methods have been developed that can be used to do (approximate) inference given a Bayesian posterior. These methods are presented in the next section.

2.3. Approximate inference

The inference of parameters based on the Bayesian posterior can be intractable, and we have to use approximate inference methods in order to resolve the posterior and quantities based upon it.[1] Therefore, several approaches exist to do approximate inference:
Some of those approaches are sampling-based, e.g. we only try to generate samples for the parameters from the posterior distribution because a large enough amount of samples can provide us with enough information about the a posteriori distribution of the parameters and can also be used to calculate relevant quantities based upon the posterior dsitribution. Other approaches turn the inference problem into an optimization problem and identify

[1] In case the prior is conjugate to the likelihood, the posterior belongs to the same distribution family as the prior and can be easily computed. For example, a Gaussian prior combined with a Gaussian likelihood leads to a Gaussian posterior as well.

either point estimates or the optimal approximate posterior distribution within a family of distributions.

2.3.1. Monte Carlo method

The Monte Carlo method (Metropolis and Ulam, 1949) is a well-known algorithm that uses random sampling to compute the outcome of an uncertain event or the approximate solution of an integration or optimization problem. An early version of the Monte Carlo method was already used in the 19th century to solve the so-called Buffon's needle problem (comte de Buffon, 1770). During World War II, Stanislaw Ulam, John von Neumann, and Nicolas Metropolis were involved in the development of the modern version of the Monte Carlo method. As their work on a nuclear weapon project was initially classified, they used the codename "Monte Carlo" for the new algorithm.

We will, in the following, use the Monte Carlo method to compute integrals that are based on our Bayesian posterior. For instance, we are interested in the expectation of some function $h(\boldsymbol{\theta})$ and define the integral I:

$$I = \int h(\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) \, d\boldsymbol{\theta} \tag{2.2}$$

This integral can be solved with the Monte Carlo Method by sampling from the posterior and then evaluating $h(\boldsymbol{\theta})$ for each sample. In more detail, we

1. generate $i = 1, ...N$ independent samples $\boldsymbol{\theta}^i$ from the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$

2. obtain an approximate value for the integral by using the Monte Carlo estimator

$$I_N = \frac{1}{N} \sum_{i=1}^{N} h(\boldsymbol{\theta}^i) \tag{2.3}$$

The Monte Carlo estimator is unbiased and approximately exact in the limit $N \to \infty$. Moreover, the convergence of the mean squared approximation error of the Monte Carlo estimator is $\mathcal{O}\left(1/\sqrt{N}\right)$ and does not depend on the dimension of the integral. This is a significant advantage compared to numerical integration schemes that strongly depend on the dimension of the integral. The computational cost of the Monte Carlo estimator, however, does depend on the evaluation cost of $h(\boldsymbol{\theta})$ and therefore the Monte Carlo method can be computationally challenging in case $h(\boldsymbol{\theta})$ is expensive to evaluate [2] A possibility to lower the cost is parallelization: the Monte Carlo algorithm can be executed in parallel as the separate evaluations can be done independently from each other.

We note that applying the Monte Carlo method without any modifications requires that

[2] We note that the Monte Carlo method is also computationally expensive if generating samples from the posterior is expensive. Especially for a high-dimensional parameter vector $\boldsymbol{\theta}$, MCMC (see Section 2.3.3 can be cumbersome and computationally costly.

samples of the posterior are readily available, which unfortunately is only rarely the case. Therefore we are in the next sections introducing altered versions of this method that allows us to generate samples in a more general setting.

2.3.2. Importance sampling

Importance sampling can be used to compute quantities of interest based on the posterior in case directly sampling from the posterior is not possible or the quality of the samples is low. Instead of directly sampling from the posterior $p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$ we sample from an alternative distribution $q(\boldsymbol{\theta})$. In more detail, Equation 2.2 is rewritten by multiplying both in the nominator and denominator with the alternative density $q(\boldsymbol{\theta})$.

$$I = \int h(\boldsymbol{\theta}) \frac{p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})}{q(\boldsymbol{\theta})} q(\boldsymbol{\theta}) \, d\boldsymbol{\theta} \tag{2.4}$$

We define

$$h^{IS}(\boldsymbol{\theta}) = h(\boldsymbol{\theta}) \frac{p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})}{q(\boldsymbol{\theta})} \tag{2.5}$$

and recover the same structure as in Equation 2.2.

$$I = \int h^{IS}(\boldsymbol{\theta}) q(\boldsymbol{\theta}) \, d\boldsymbol{\theta} \tag{2.6}$$

Therefore, we can define the following two steps procedure similar to the original Monte Carlo method to compute the value of the integral. We

- 1. generate $i = 1, ...N$ independent samples $\boldsymbol{\theta}^i$ from an alternative distribution $q(\boldsymbol{\theta})$
- 2. obtain an approximate value V_N for the integral by using the estimator

$$V_N = \frac{1}{N} \sum_{i=1}^{N} h(\boldsymbol{\theta}^i) \frac{p(\boldsymbol{\theta}^i|\boldsymbol{\mathcal{D}})}{q(\boldsymbol{\theta}^i)} = \frac{1}{N} \sum_{i=1}^{N} h^{IS}(\boldsymbol{\theta}^i) \tag{2.7}$$

  The quotient $\frac{p(\boldsymbol{\theta}^i|\boldsymbol{\mathcal{D}})}{q(\boldsymbol{\theta}^i)}$ is often called importance weight w_i.

Importance sampling works best if the alternative distribution is close to the true posterior. In this case, all importance weights have a similar magnitude, and each sample contributes similarly to the estimator. The quality of importance sampling can be assessed using the Effective Sample Size (ESS) (Martino et al., 2017).

$$\text{ESS} = \frac{(\sum_{i=1}^{N} w_i)^2}{\sum_{i=1}^{N} w_i^2} = \frac{1}{\sum_{i=1}^{N} \bar{w}_i^2} \tag{2.8}$$

with the normalized importance weights

$$\bar{w_i} = \frac{w_i}{\sum_{i=1}^{N} w_i} \tag{2.9}$$

The ESS represents approximately how many direct samples of the actual posterior $p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$ would lead to an estimate of the integral with the same quality. The maximum value of the ESS is N, and the minimum value is 1. In the latter case, the computed approximation is usually clearly off, and another alternative distribution should be selected. We note that, moreover, importance sampling does perform poorly for high-dimensional posteriors. In such a case, we recommend using another inference method.

2.3.3. Markov Chain Monte Carlo methods

Markov Chain Monte Carlo (MCMC) methods provide a general framework to sample from various probability density functions. They combine the Monte Carlo method with Markov chains in order to enable efficient sampling algorithms.
The MCMC method generates samples based on an initial starting point $\boldsymbol{\theta}^0$ and a proposal distribution $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t)$ which proposes new samples $\boldsymbol{\theta}^*$ based on the current sample $\boldsymbol{\theta}^t$. These proposed samples are then either accepted or rejected based on a given criterion. The method is called MCMC, as the generated samples form a Markov chain. The proposal distribution as well as the acceptance criteria has to be selected such that the samples follow the targeted distribution $p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$.

A popular choice is the so called Metropolis algorithm (Metropolis et al., 1953) which uses a symmetric proposal distribution

$$q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t) = q(\boldsymbol{\theta}^t|\boldsymbol{\theta}^*) \tag{2.10}$$

and an acceptance with probability $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t)$:

$$\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t) = \min\left(1, \frac{p(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{D}})}{p(\boldsymbol{\theta}^t|\boldsymbol{\mathcal{D}})}\right) \tag{2.11}$$

To generate N samples with the Metropolis algorithm, we start at $\boldsymbol{\theta}^t$ for $t = 0$. Subsequently,

1. we generate a new sample $\boldsymbol{\theta}^*$ with the given proposal distribution.
2. we check if this new sample is accepted and are therefore sampling a from a uniform distribution $U[0, 1]$. This value a is compared with the acceptance probability $\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t)$:

$$\boldsymbol{\theta}_{t+1} = \begin{cases} \boldsymbol{\theta}^* & \text{if } a < \alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t) \\ \boldsymbol{\theta}^t & \text{otherwise} \end{cases} \tag{2.12}$$

The steps above are then reiterated N times until we have our N samples $\boldsymbol{\theta}^{1:N}$. These

samples are, in general, correlated and not independent. [3] To keep the autocorrelation as small as possible, we have to ensure that the new proposals are distant from the current samples while the acceptance rate is high. Moreover, the Markov chain can be thinned by, for instance, keeping only every third sample and thus reducing the autocorrelation. The samples at the start of the Markov chain are usually disregarded as the Markov Chain needs some samples until it reaches the targeted stationary distribution $p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$.

The MCMC algorithm can only generate successfully samples if the targeted distribution is invariant under the transition kernel $T(\boldsymbol{\theta}^t, \boldsymbol{\theta}^*)$. A sufficient (but not necessary) condition is the detailed balance equation:

$$p(\boldsymbol{\theta}^t|\boldsymbol{\mathcal{D}})T(\boldsymbol{\theta}^t, \boldsymbol{\theta}^*) = p(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{D}})T(\boldsymbol{\theta}^*, \boldsymbol{\theta}^t) \tag{2.13}$$

If this criterion is fulfilled, the Markov chain is reversible and converges asymptotically to the targeted distribution (Bishop and Nasrabadi, 2006).

We can show that the Metropolis algorithm fulfills the detailed balance equation. As a first step, we express the transition kernel with the proposal distribution and acceptance probability:

$$T(\boldsymbol{\theta}^t, \boldsymbol{\theta}^*) = p(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t)\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t) \tag{2.14}$$

Now we can check the detailed balance equation:

$$\begin{aligned}
p(\boldsymbol{\theta}^t|\boldsymbol{\mathcal{D}})T(\boldsymbol{\theta}^t, \boldsymbol{\theta}^*) &= p(\boldsymbol{\theta}^t|\boldsymbol{\mathcal{D}})p(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t)\alpha(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t) \\
&= p(\boldsymbol{\theta}^t|\boldsymbol{\mathcal{D}})p(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t)\min\left(1, \frac{p(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{D}})}{p(\boldsymbol{\theta}^t|\boldsymbol{\mathcal{D}})}\right) \\
&= p(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t)\min\left(p(\boldsymbol{\theta}^t|\boldsymbol{\mathcal{D}}), p(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{D}})\right) \\
&= p(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t)p(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{D}})\min\left(\frac{p(\boldsymbol{\theta}^t|\boldsymbol{\mathcal{D}})}{p(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{D}})}, 1\right) \\
&= p(\boldsymbol{\theta}^t|\boldsymbol{\theta}^*)p(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{D}})\min\left(\frac{p(\boldsymbol{\theta}^t|\boldsymbol{\mathcal{D}})}{p(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{D}})}, 1\right) \\
&= p(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{D}})p(\boldsymbol{\theta}^t|\boldsymbol{\theta}^*)\alpha(\boldsymbol{\theta}^t|\boldsymbol{\theta}^*) \\
&= p(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{D}})T(\boldsymbol{\theta}^*, \boldsymbol{\theta}^t)
\end{aligned} \tag{2.15}$$

We used the property that the proposal distribution of the Metropolis algorithm is symmetric in the fifth line of this proof.

We note that for MCMC algorithms it is sufficient to know the target probability distribution up to a constant as in the algorithm we only need the quotient of this targeted distribution

[3] The integrated autocorrelation time (Roberts and Rosenthal, 2001) can be used as a measure to quantify the autocorrelation between the samples. The higher the autocorrelation time, the larger is the efficiency loss compared to N independent samples.

with itself. Therefore, we are not required to compute the evidence term in the Bayesian posterior and the product of the likelihood and the a-priori-distribution is enough to generate samples.

Having computed the samples $\boldsymbol{\theta}^{1:t}$ via a MCMC algorithm, we can readily evaluate the targeted integral in Equation 2.2 with

$$I_N = \frac{1}{N} \sum_{t=1}^{N} h(\boldsymbol{\theta}^t) \tag{2.16}$$

2.3.4. Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm (Hastings, 1970) extended the Metropolis algorithm to asymmetrical proposal distributions. As a consequence, the acceptance criterion has to be altered and does now include the proposal distribution as well.

$$\alpha_{MH}(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t) = \min\left(1, \frac{q(\boldsymbol{\theta}^t|\boldsymbol{\theta}^*)p(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{D}})}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t)p(\boldsymbol{\theta}^t|\boldsymbol{\mathcal{D}})}\right) \tag{2.17}$$

It is easy to see that if the proposal distribution should be symmetric, i.e. $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t) = q(\boldsymbol{\theta}^t|\boldsymbol{\theta}^*)$, we recover the acceptance criterion of the Metropolis algorithm.
The proof of the detailed balance equation can be done identically to Equation 2.15.
We note that we again do not need a normalized targeted density as we still have only a quotient involved in the acceptance criteria.

2.3.5. Metropolis-adjusted Langevin algorithm

The Metropolis-adjusted Langevin algorithm (MALA) (Besag, 1994; Roberts and Tweedie, 1996) incorporates gradient information in the proposal distribution in order to more efficiently explore the regions of the target distribution with a high probability density.
In more detail, we employ a discretized version of the overdamped Langevin dynamic to generate new proposals:

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}^t + \frac{1}{2}\sigma^2 \, \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}^t|\boldsymbol{\mathcal{D}}) + \sigma\boldsymbol{\epsilon} \tag{2.18}$$

where σ is a parameter that needs to be specified and allows to control the step size and thus the acceptance/rejection rate of the algorithm. $\boldsymbol{\epsilon}$ is sampled from a multivariate standard normal distribution. This leads to the following proposal distribution:

$$q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^t) = \mathcal{N}(\boldsymbol{\theta}^t + \frac{1}{2}\sigma^2 \, \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}^t|\boldsymbol{\mathcal{D}}), \sigma^2) \tag{2.19}$$

As this proposal distribution is asymmetric, the Metropolis-Hastings acceptance ratio has to be computed and proposals have to accepted/rejected accordingly.
If gradient information of the target distribution is available, the MALA algorithm is a reasonable choice and, in general, performs better than a proposal distribution based on a

random walk. However, it can suffer from multimodality of the targeted distribution. Here, other algorithms such as Sequential Monte Carlo (SMC) (Liu and Chen, 1998) could be a better choice.

2.3.6. Point estimates

Instead of trying to find the actual posterior distribution, we can take a simplified approach and only find the parameter values that have the highest probability. This leads to point-based estimates that contain less information than the full posterior but are comparably easy to compute by finding the maximum of the a-posteriori density.

$$\boldsymbol{\theta}_{MAP} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta})\; p(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\; \log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \quad (2.20)$$

We note that instead of finding the maximum of the actual posterior, we can also find the maximum of the log posterior. As the log-function is strictly monotonically increasing, the location of the maximum does not change. Finding the maximum of the log posterior is usually preferable as the values of the posterior can span several orders of magnitude, especially if its dimension is high.

Point-based estimates can also be computed based on the likelihood and are then called Maximum-Likelihood estimates:

$$\boldsymbol{\theta}_{MLE} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\; \log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) \quad (2.21)$$

Again the computation is, in general, easier when using the log-likelihood.

As these point-based estimates do not contain any information regarding the uncertainty anymore, the calculated predictive uncertainty based on either $\boldsymbol{\theta}_{MLE}$ or $\boldsymbol{\theta}_{MAP}$ can be too low. This is especially the case if the variance of the actual posterior distribution is high, whereas for very narrow posterior distributions the error is very low.

2.3.7. Variational Bayesian inference

Similar to the point-estimates introduced in the section before, the Variational Bayesian method turns our inference problem into an optimization problem and can therefore also be used to compute an approximation for an analytically intractable posterior $p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$. These methods originated in statistical physics (see e.g. (Opper and Saad, 2001)) and were later adapted for probabilistic machine learning (Neal and Hinton, 1998; Hoffman et al., 2013; Blei et al., 2017). Instead of finding the values with the highest probability, Variational Bayesian inference methods want to identify the distribution $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ from a parameterized family of distribution that approximates our posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$ best. The parameters $\boldsymbol{\phi}$ are used to parameterize the family of distributions.

$$q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}}) \quad (2.22)$$

This distribution is often called the variational distribution. Usually we employ for $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ a family of distributions from which we can easily sample. This allows us to readily use the obtained distribution to compute further predictions. A good choice is, for instance, a multivariate normal distribution.

In order to find this distribution, we first have to define a metric that quantifies the difference between two probability distributions. A popular choice is the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951):

$$KL(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})) = \int q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})} \, d\boldsymbol{\theta} \tag{2.23}$$

We note that the KL-divergence is non-symmetric and always larger than zero. It is only zero if the two distributions are identical.

To find the best possible distribution $q_{\boldsymbol{\phi}^*}(\boldsymbol{\theta})$ we now have to minimize the KL-divergence with respect to the parameters of $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$.

$$q_{\boldsymbol{\phi}^*}(\boldsymbol{\theta}) = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \; KL(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})) \tag{2.24}$$

However, this is in general not possible as it involves the intractable posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})$. Fortunately, we can use Bayes' law and obtain a quantity that can be optimized. We, therefore, decompose the KL-divergence:

$$\begin{aligned} KL(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})) &= \int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})} \, d\boldsymbol{\theta} \qquad (2.25) \\ &= \int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta})(\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})) \, d\boldsymbol{\theta} \\ &= \int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta})(\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) - \log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) + \log p(\mathcal{D})) \, d\boldsymbol{\theta} \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} \left[\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) - \log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) + \log p(\mathcal{D})\right] \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} \left[\log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) - \log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) - \log p(\boldsymbol{\theta})\right] + \log p(\mathcal{D}) \qquad (2.26) \end{aligned}$$

We first used here the properties of the logarithm and subsequently the Bayes' theorem. Finally, we can directly compute the expected value of the last term as this term is only a constant.
We rearrange the resulting equation:

$$\log p(\mathcal{D}) - KL(q_{\boldsymbol{\theta}}(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\boldsymbol{\mathcal{D}})) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} \left[\log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})\right] \tag{2.27}$$

As the log-evidence $\log p(\mathcal{D})$ is constant with respect to our variational distribution and the KL-divergence is always larger than zero, we can maximize the terms on the left and thus minimize the KL-divergence by maximizing the tractable term on the right-hand side of the equa-

tion. This term is called Evidence Lower Bound (ELBO) $\mathcal{F}(q(\boldsymbol{\theta}))$:

$$\mathcal{F}(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}\left[\log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})\right] \tag{2.28}$$

We note that we can show that the ELBO is the lower bound of the log-evidence by using Jensen's inequality (Bishop and Nasrabadi, 2006):

$$\begin{aligned}
\log p(\boldsymbol{\mathcal{D}}) &= \log \int_{\boldsymbol{\theta}} p(\boldsymbol{\mathcal{D}}, \boldsymbol{\theta})\, d\boldsymbol{\theta} \\
&= \log \int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \frac{p(\boldsymbol{\mathcal{D}}, \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}\, d\boldsymbol{\theta} \\
&\geq \int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\mathcal{D}}, \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}\, d\boldsymbol{\theta} \\
&= \int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}\, d\boldsymbol{\theta} \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}\left[\log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})\right] = \mathcal{F}(q_{\boldsymbol{\phi}}(\boldsymbol{\theta}))
\end{aligned} \tag{2.29}$$

Moreover, we can obtain another expression for the ELBO by introducing the KL-divergence of the prior with the variational distribution:

$$\begin{aligned}
\mathcal{F}(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}\left[\log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})\right] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}\left[\log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta})\right] - KL(q_{\boldsymbol{\phi}}(\boldsymbol{\theta})||p(\boldsymbol{\theta}))
\end{aligned} \tag{2.30}$$

This result is perfectly in accordance with the Bayesian probability theory. Our best possible distribution is found as a trade-off between being close to the prior and accounting for the data via the likelihood term.

The obtained ELBO $\mathcal{F}(q(\boldsymbol{\theta}))$ can be maximized using stochastic optimization algorithms (see Section 2.4) and thus the optimal parameter values $\boldsymbol{\phi}^*$ are determined. The quality of the obtained approximate distribution $q_{\boldsymbol{\phi}^*}(\boldsymbol{\theta})$ depends on how close the actual posterior distribution is to the chosen variational distribution family. The variational inference method can only arrive at the exact posterior if the posterior belongs to the chosen variational distribution family. Therefore, we have to carefully choose our variational distribution family as it strongly influences the accuracy of the inference algorithm.

We note that the point-based approximations presented before in section 2.3.6 can be described as a special case of variational inference with a Dirac delta function as the variational distribution.

2.4. Stochastic optimization

In this section, the basics of stochastic optimization are presented. As an illustrative example, we choose the ELBO $\mathcal{F}(q_{\boldsymbol{\phi}}(\boldsymbol{\theta}))$ (see Equation 2.30) which needs to be optimized during variational inference. Before introducing two stochastic optimization algorithms in more

detail, we, therefore, show first how to compute gradients of the ELBO with respect to the parameters $\boldsymbol{\phi}$:

$$\nabla_{\boldsymbol{\phi}} \mathcal{F}(q(\boldsymbol{\theta})) = \nabla_{\boldsymbol{\phi}} \left(\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} \left[\log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) \right] - KL(q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) || p(\boldsymbol{\theta})) \right) \tag{2.31}$$

We note that the KL-term and its gradient can be computed analytically for most cases as both the prior and the approximative distribution are usually tractable (Kingma and Welling, 2014). In the following, we therefore focus on the gradient $\nabla_{\boldsymbol{\phi}} \left(\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} \left[\log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) \right] \right)$.
This gradient and the expectation involved can be computed with the help of a Monte Carlo algorithm (Paisley et al., 2012; Kingma and Welling, 2014):

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}} \left(\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} \left[\log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) \right] \right) &= \int_{\boldsymbol{\theta}} \log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) \nabla_{\boldsymbol{\phi}} \, q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \, d\boldsymbol{\theta} \\
&= \int_{\boldsymbol{\theta}} \log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\phi}} \, \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \, d\boldsymbol{\theta} \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} \left[\log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \right] \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}^{\boldsymbol{i}}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}^i) \;\; \text{with} \;\; \boldsymbol{\theta}^i \sim q_{\boldsymbol{\phi}}(\boldsymbol{\theta})
\end{aligned}$$

We used the identity $\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) = q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$. This gradient estimator unfortunately has a very high variance and is therefore not a good choice (Paisley et al., 2012). To reduce the variance, we are applying the reparameterization trick (Kingma and Welling, 2014), i.e. we define our samples in terms of a new distribution $p(\boldsymbol{\epsilon})$ that does not depend on the parameters $\boldsymbol{\phi}$. The function $f_{\boldsymbol{\phi}}(\boldsymbol{\epsilon})$ maps $\boldsymbol{\epsilon}$ to $\boldsymbol{\theta}$. In particular,

$$\boldsymbol{\theta} = f_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}) \tag{2.32}$$

with

$$\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \tag{2.33}$$

Given the reparameterization trick, we can now compute the targeted gradient with a different Monte Carlo estimator:

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}} \left(\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} \left[\log p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) \right] \right) &= \nabla_{\boldsymbol{\phi}} \left(\mathbb{E}_{p(\boldsymbol{\epsilon})} \left[\log p(\boldsymbol{\mathcal{D}}|\boldsymbol{f_{\phi}}(\boldsymbol{\epsilon})) \right] \right) \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\phi}} \left(\log p(\boldsymbol{\mathcal{D}}|\boldsymbol{f_{\phi}}(\boldsymbol{\epsilon}^{\boldsymbol{i}})) \right) \;\; \text{with} \;\; \boldsymbol{\epsilon}^i \sim p(\boldsymbol{\epsilon}^i)
\end{aligned} \tag{2.34}$$

This formulation leads to gradients that still have some noise but typically less than the original formulation. Moreover, the noise of the gradients is increased by usually not processing all data $\mathcal{D}$ but only a subset of the data, i.e. a minibatch, in each gradient step.

After we have obtained our potential noisy gradients, which we will call $\boldsymbol{g}$ in the following, we can state a general update rule for a stochastic optimization algorithm:

$$\boldsymbol{\phi}^{t+1} = \boldsymbol{\phi}^t + \nu^t \boldsymbol{g} \tag{2.35}$$

Here, ν^t is the learning rate at step t. This learning rate has to be always larger than zero. We note that it would be possible to define different learning rates for each dimension, which can be beneficial if the length scales of the different dimensions differ a lot.

2.4.1. Robbins-Monro stochastic optimization

The Robbins-Monro algorithm (Robbins and Monro, 1951) is one of the first stochastic optimization algorithms and is guaranteed to converge to equilibrium for

$$\sum_{t=1}^{\infty} \nu^t = \infty \text{ and } \sum_{t=1}^{\infty} \left(\nu^t\right)^2 < \infty \tag{2.36}$$

A popular choice for the learning rate is

$$\nu^t = \frac{\alpha}{(\beta + t)^\rho} \tag{2.37}$$

with the tunable parameters $\alpha > 0, \beta > 0$ and $\rho \in (0.5, 1]$

2.4.2. Adam stochastic optimization

Adaptive moment estimation (Adam) (Kingma and Ba, 2014) is a stochastic optimization algorithm that estimates both the first and second moment of the gradient using a moving average and uses it to automatically adapt the learning rate. As it requires little to no user input, ADAM is currently one of the most popular algorithms in stochastic optimization and is heavily used within the Deep Learning community.

In each step, the algorithm computes an estimate for the first moment $\boldsymbol{m}$ and second moment $\boldsymbol{v}$ using an exponentially decaying moving average

$$\boldsymbol{m}^t = \beta_1 \boldsymbol{m}^{t-1} + (1 - \beta_1)\boldsymbol{g}^t \qquad \boldsymbol{v}^t = \beta_2 \boldsymbol{v}^{t-1} + (1 - \beta_2)\boldsymbol{g} \odot \boldsymbol{g} \tag{2.38}$$

where $\odot$ indicates the element-wise product. Afterward, a bias correction because of the starting values $\boldsymbol{m}^0, \boldsymbol{v}^0 = 0$ has to be applied:

$$\hat{\boldsymbol{m}}^t = \frac{\boldsymbol{m}}{1 - \beta_1^t} \qquad \hat{\boldsymbol{v}}^t = \frac{\boldsymbol{v}}{1 - \beta_2^t} \tag{2.39}$$

Finally, we can update the parameters:

$$\boldsymbol{\phi}^{t+1} = \boldsymbol{\phi}^t + \frac{\alpha \hat{\boldsymbol{m}}}{\sqrt{\hat{\boldsymbol{v}}^t} + \boldsymbol{\epsilon}} \tag{2.40}$$

For a detailed description of the algorithm as well as suggested values for the parameters α, β_1, β_2, and ϵ, we refer to the original paper by Kingma and Ba (2014).

2.5. Deep Learning & Neural Networks

Deep Learning is considered a subset of machine learning. In particular, Deep Learning algorithms employ Neural Networks (NN) with multiple hidden layers to learn from an, in general, large amount of data. Due to the high amount of parameters involved, Deep Learning methods need large computational resources and have especially benefited from the technical development regarding GPUs in the last years and the availability of automatic differentiation frameworks (Abadi et al., 2015; Paszke et al., 2017; Bradbury et al., 2018).
This section introduces the three most popular Neural Networks architectures: Feed-forward neural networks, convolutional neural networks and recurrent neural network. We note that this is only a tiny subset of the currently available Deep Learning architecures. A more detailed summary of Deep Learning is presented in LeCun et al. (2015) or Goodfellow et al. (2016).
Neural Networks, also known as Artificial Neural Networks (ANNs), are the key element of most Deep Learning algorithms. They were originally developed with the intention to mimic the human brain (McCulloch and Pitts, 1943). A neural network usually employs a large number of parameters and thus requires a comparably large training data set. The most significant advantage of NN is their flexibility as they can be universal approximators (Hornik et al., 1989).

2.5.1. Feed-forward Neural Network

The most popular neural network is the feed-forward neural network that takes a value $\boldsymbol{x}^0$ as an input and processes it using multiple hidden layers until an output $\boldsymbol{y}$ is reached. A possible architecture is shown in Figure 2.1.
From a mathematical point of view, the following operations take place within such a feed-forward neural network. We start with a d_0-dimensional input $\boldsymbol{x}^0$. The values at the first hidden layer are then computed using a weight matrix $\boldsymbol{W}^1$, a bias vector $\boldsymbol{b}^1$, and an element-wise activation function h^1. We assume a d_1-dimensional first hidden layer with values x_j^1 for $j = 1, .., d_1$.

$$x_j^1 = h^1\left(\boldsymbol{W}_{j,i}^1 x_i^0 + b_j^1\right) \quad \text{for} \quad j = 1, ..., d_1 \tag{2.41}$$

Subsequently, the values at the d_2-dimensional second hidden layer are computed similarly, but with a new weight matrix and bias vector as well as possibly another activation function.

$$x_j^2 = h^2\left(\boldsymbol{W}_{j,i}^2 x_i^1 + b_j^2\right) \quad \text{for} \quad j = 1, ..., d_2 \tag{2.42}$$

Finally, we can compute the d_3-dimensional output with yet again another weight matrix,

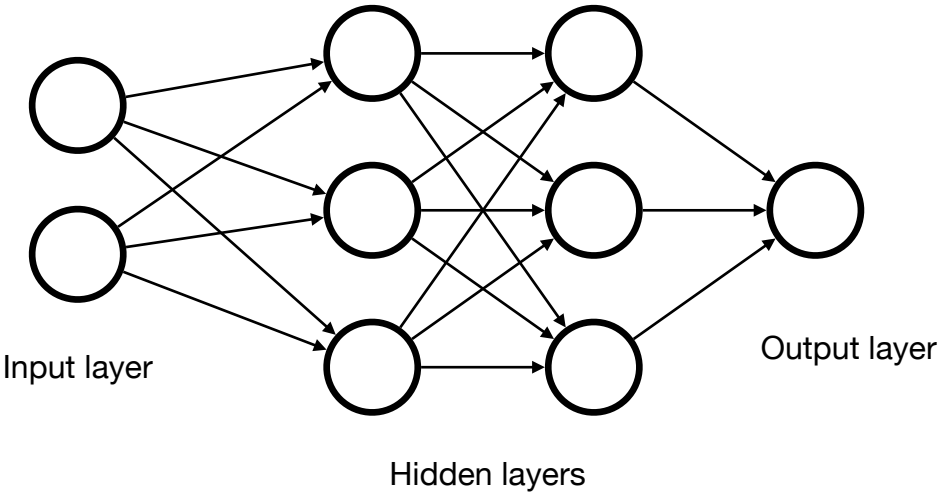


Figure 2.1: Illustrative architecture of a feed-forward neural network

bias vector, and activation function.

$$y_j = h^3\left(\boldsymbol{W}^3_{j,i} x^2_i + b^3_j\right) \quad \text{for} \quad j = 1, ..., d_3 \tag{2.43}$$

The activation functions are usually identical for each layer, but different activation functions could also be employed. Popular activation functions can for instance be found in Bishop and Nasrabadi (2006) and are usually non-linear, differentiable and monotonic. As for each hidden layer, all inputs are connected with all outputs, these layers are also called fully-connected layers.

For the commonly used activation functions, neural networks are differentiable with regards to their parameters using the chain rule. These derivatives are required during the training of a neural network [4] and their easy accessibility is beside their flexibility one of the main reasons for the popularity of neural networks in the machine learning community.

We note that we could also treat all parameters as random variables and thus turn the neural network into a Bayesian neural network. However, the inference is challenging due to the high-dimensional parameter space (Li and Gal, 2017) and the choice of suitable priors for the parameters is not straightforward (Fortuin et al., 2022).

2.5.2. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) (Fukushima, 1980; LeCun et al., 1989) are another popular neural network architecture. CNNs are designed for grid-like data and are therefore primarily used for tasks such as image and video processing and classification but also for discretized time series data. In contrast to the feed-forward neural networks discussed

[4] The training of a neural network is equivalent to solving a (stochastic) optimization problem to obtain the optimal parameter values. The quantity which is minimized is called the loss function.

in Section 2.5.1, they contain convolutional layers and pooling layers in addition to fully-connected layers.

Whereas in a fully-connected layer the outputs for each layer are depending on all inputs, a convolutional layer only has sparse interactions. The outputs of a convolutional layer are convolutions of the inputs with a fixed-sized kernel that is learned from the data. We note that due to employing a convolution with a kernel instead of a general matrix multiplication as in Section 2.5.1, the amount of parameters in a convolutional layer is lower than in the corresponding feed-forward layer. Convolutional layers are moreover shift-invariant.
Pooling Layers are used to enforce invariance of the outputs with regards to small transformations (Goodfellow et al., 2016). For instance, a max pooling layer reports the maximum value within a fixed-size neighborhood.

For a more detailed description the author refers the interested reader to Bishop and Nasrabadi (2006) and Goodfellow et al. (2016) or similar textbooks with a focus on image analysis and pattern recognition.

2.5.3. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are neural networks for processing a sequential input data and learning a propagator for the data. The recurrent neural networks employ an internal memory state $\boldsymbol{h}$, which is updated for new information that is received by the network. Given an initial internal memory state $\boldsymbol{h}_0$ and a time-series $\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_T$ as input for the RNN, the RNN updates in each step the internal hidden state given the current input:

$$\boldsymbol{h}_{t+1} = \mathcal{F}(\boldsymbol{h}_t, \boldsymbol{x}_t) \tag{2.44}$$

Subsequently, the network predicts the desired output, e.g. the the next value for the given time-series, based upon the internal hidden memory.

$$\boldsymbol{x}_{t+1} = \mathcal{G}(\boldsymbol{h}_{t+1}) \tag{2.45}$$

Here, $\mathcal{F}$ and $\mathcal{G}$ are two non-linear functions. By taking into account memory when predicting new states a RNN is capable of representing non-Markovian dynamics. Depending on how exactly $\mathcal{F}$ and $\mathcal{G}$ are chosen, different variations are possible (Goodfellow et al., 2016).

Unfortunately, the computation of gradients and thus the training of RNNs can be complicated as gradients can become very small or very large as the gradient computation of the state at a time t depends on all steps taken before (Goodfellow et al., 2016). A possible remedy is to learn when to incorporate new information into the hidden state and when to forget old information in order to ensure non-zero gradients. This is done in the Long-Short-Term-memory network (LSTM) (Hochreiter and Schmidhuber, 1997) by introducing a self-loop into the architecture.

2.6. Physics-aware Neural Networks

This section introduces four of the most popular physics-aware neural network architectures: the Physics-informed Neural Networks (PINNs) framework, Deep Operator Networks (DeepONets), Learning an Effective Dynamics (LED) as well as the Hamiltonian Neural Network (HNN).

2.6.1. Physics-informed Neural Networks

PINNs were introduced by Raissi et al. (2019) and have quickly become one of the most used physics-aware neural network architectures. They model the solution of a PDE with a neural network and incorporate the residual of the PDE as an additional term in the loss function, which is minimized to find the optimal parameter values. The method, therefore, can be described as a collocation method that employs a neural network as the trial function. Given a PDE

$$\boldsymbol{u}_t + \mathcal{N}(\boldsymbol{u}) = 0 \;\text{ with }\; t \in [0,T], \boldsymbol{x} \in \Omega \tag{2.46}$$

with the non-linear differential operator $\mathcal{N}$, we model the solution $\boldsymbol{u}_{\boldsymbol{\phi}}(\boldsymbol{x},t)$ with a deep neural network with parameters $\boldsymbol{\phi}$. The neural network is trained using a loss that consists of two parts. The first part MSE_{BC} minimizes the error at the initial and boundary conditions, and the second part the error within the domain MSE_{Res}. Given N_f collocation points $\{\boldsymbol{x}_f^i, t_f^i\}_{i+1}^{N_f}$ and N_u points at the boundary $\{\boldsymbol{x}_u^i, t_u^i, \boldsymbol{u}^i\}_{i+1}^{N_u}$, we can state the two loss terms (Raissi et al., 2019):

$$\text{MSE}_{BC} = \frac{1}{N_u}\sum_{i=1}^{N_u} |\boldsymbol{u}_{\boldsymbol{\phi}}(\boldsymbol{x}_u^i, t_u^i) - \boldsymbol{u}^i|^2 \tag{2.47}$$

and

$$\text{MSE}_{Res} = \frac{1}{N_u}\sum_{i=1}^{N_f} |\mathcal{N}(\boldsymbol{u}_{\boldsymbol{\phi}}(\boldsymbol{x}_f^i, t_f^i)) + (\boldsymbol{u}_{\boldsymbol{\phi}}(\boldsymbol{x}_f^i, t_f^i))_t|^2 \tag{2.48}$$

2.6.2. Deep Operator Networks

This section is based upon the DeepONet review in Kaltenbach et al. (2022):

DeepONets (Lu et al., 2021) have been developed to solve parametric PDEs and significantly extend the Physics-Informed Neural Network (PINNs, Raissi et al. (2019)) framework as no additional training phase is required if the input parameters of the PDE are changed. We consider a, potentially nonlinear and time-dependent, PDE with an input function $u \in \mathcal{U}$ and solution function $s \in \mathcal{S}$ where $\mathcal{U}, \mathcal{S}$ are appropriate Banach spaces. The former

can represent e.g. source terms, boundary or initial conditions, material properties. Let:

$$\mathcal{N}(u,\ s)(\boldsymbol{\xi}) = 0 \tag{2.49}$$

denote the governing PDE where $\mathcal{N} : \mathcal{U} \times \mathcal{S} \to \mathcal{V}$ is an appropriate differential operator and $\boldsymbol{\xi}$ the spatio-temporal coordinates. Furthermore, let:

$$\mathcal{B}(u,\ s)(\boldsymbol{\xi}) = 0 \tag{2.50}$$

denote the operator $\mathcal{B} : \mathcal{U} \times \mathcal{S} \to \mathcal{V}$ associated with the boundary or initial conditions. Assuming that the solution s for each $u \in \mathcal{U}$ is unique, we denote with $\mathcal{G} : \mathcal{U} \to \mathcal{S}$ the solution operator that maps from any input u to the corresponding solution s. The goal of DeepONets is to approximate it with an operator $G_{\boldsymbol{\theta}}$ that depends on tunable parameters $\boldsymbol{\theta}$. The latter can yield an approximation to the actual solution at any spatio-temporal point $\boldsymbol{\xi}$ which we denote by $G_{\boldsymbol{\theta}}(\boldsymbol{\xi})$. It is based on a separated representation (Lu et al., 2021). We omit the NN parameters $\boldsymbol{\theta}$ on the right-hand side in order to simplify the notation:

$$G_{\boldsymbol{\theta}}(u)(\boldsymbol{\xi}) = \sum_{j=1}^{Q} b_j \left(\underbrace{u(\boldsymbol{\eta}_1), ..., u(\boldsymbol{\eta}_F)}_{\boldsymbol{u}} \right) t_j(\boldsymbol{\xi}) \tag{2.51}$$

and consists of the so-called branch network whose terms b_j depend on the values of the input function u at F fixed spatio-temporal locations $\{\boldsymbol{\eta}_l\}_{l=1}^{F}$ which we summarily denote with the vector $\boldsymbol{u} \in \mathbb{R}^F$, and the so-called trunk network whose terms t_j depend on the spatio-temporal coordinates $\boldsymbol{\xi}$ (see Figure 2.2). Both networks have trainable weight and bias parameters which we denote collectively by $\boldsymbol{\theta}$.

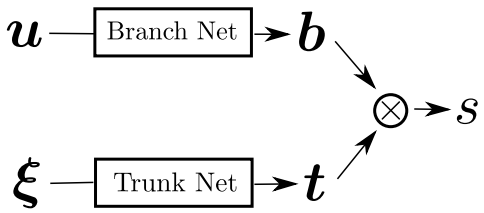


Figure 2.2: DeepONet architecure Kaltenbach et al. (2022)

We emphasize that, once trained, the DeepONet can provide predictions of the solution at any spatio-temporal location $\boldsymbol{\xi}$, a feature that is very convenient in the context of inverse problems as the same DeepONet can be used for solving problems with different sets of observations.
DeepONets are trained using a loss-function that can both incorporate given data points as well as a physics-informed term (Wang et al., 2021) similar to PINNs.

2.6.3. Learning Effective Dynamics framework

The Learning the Effective Dynamics (LED) framework (Vlachas et al., 2022) resemble a combination of an auto-encoder with a recurrent neural network.
A high-dimensional input is mapped to a lower-dimensional representation using the Encoder part of the Autoencoder (Kingma and Welling, 2014) and than propagated in time using a recurrent neural network. The full high-dimensional state is then restored using the Decoder.

From a mathematical perspective, given a high-dimensional time series $\boldsymbol{x}_n$ with $n = 1, ..., T$ the encoder maps the high-dimensional representation to the latent space as:

$$\boldsymbol{z}_n = \boldsymbol{E}_{\boldsymbol{\theta_{AE}}}(\boldsymbol{x}_n) \tag{2.52}$$

The decoder reconstructs the high-dimensional representation based on the latent variables as:

$$\boldsymbol{x}_n = \boldsymbol{D}_{\boldsymbol{\theta_{AE}}}(\boldsymbol{z}_n) \tag{2.53}$$

We denote the parameters of the encoder as well as the decoder by $\boldsymbol{\theta_{AE}}$. Subsequently, a RNN with parameters $\boldsymbol{\theta}_{RNN}$ is learned as a propagator for the obtained low-dimensional time series $\boldsymbol{z}_n$ with $n = 1, ..., T$. More details on recurrent neural network can be found in Section 2.5.3. In the original LED publication (Vlachas et al., 2022), a CNN based Autoencoder is used and a LSTM is employed as the recurrent neural network. The two sets of parameters can be either trained separately by first learning the Autoencoder and afterwards the LSTM or together by adding up the loss-terms for the Autoencoder and the LSTM.

An important advantage of the LED framework is that if the quality of the LED prediction should be declining, the decoder can be used to lift a lower-dimensional representation back to its high-dimensional state. Now, based on the obtained high-dimensional state the time-series can be propagated in case the underlying high-dimensional PDE is known and more training data could be generated. The additional data can be used to retrain the LED framework and the obtained new high-dimensional state can be used as a starting point for new predictions. Vlachas et al. (2022) have obtained very accurate predictions by constantly switching between the low-dimensional representation and the high-dimensional representation. Moreover, the computational costs were significantly reduced compared to solving the system only in the high-dimensional representation.
Unfortunately, it is difficult to decide when to switch back to the high-dimensional representation as a metric for the accuracy of the lower-dimensional states (generated by using the LSTM propagator) has to be obtained without access to reference data. Kičić et al. (2023) have introduced a metric based on UQ and an ensemble of probabilistic networks that is used to decide when to switch to the high-dimensional representation.

2.6.4. Hamiltonian Neural Networks

HNNs were introduced by Greydanus et al. (2019). This architecture does not include physical information within the loss term during training but incorporates the principle of

Hamiltonian mechanics directly as inductive bias into the architecture. In particular, a differential representation of the Hamiltonian of a physical system is learned. The Hamiltonian equations are then invoked to obtain the equations for the generalized coordinates $\boldsymbol{q}$ and momenta $\boldsymbol{p}$, i.e.

$$\dot{\boldsymbol{q}} = \frac{\partial H}{\partial \boldsymbol{p}}, \ \dot{\boldsymbol{p}} = -\frac{\partial H}{\partial \boldsymbol{q}} \tag{2.54}$$

The Hamiltonian H expresses the total energy of the system, which is conserved during the dynamic evolution. HNNs learn the Hamiltonian based on N observations of the state variables $\{\boldsymbol{q}^i, \boldsymbol{p}^i\}_{i+1}^N$ and their derivatives $\{\dot{\boldsymbol{q}}^i, \dot{\boldsymbol{p}}^i\}_{i+1}^N$ with the following loss function $\mathcal{L}(\boldsymbol{\phi})$:

$$\mathcal{L}(\boldsymbol{\phi}) = \frac{1}{N}\sum_{i=1}^{N} \left| \frac{\partial H_{\boldsymbol{\phi}}(\boldsymbol{q}^i, \boldsymbol{p}^i)}{\partial \boldsymbol{p}} - \dot{\boldsymbol{q}}^i \right|^2 + \left| \frac{\partial H_{\boldsymbol{\phi}}(\boldsymbol{q}^i, \boldsymbol{p}^i)}{\partial \boldsymbol{q}} + \dot{\boldsymbol{p}}^i \right|^2 \tag{2.55}$$

Here, $\boldsymbol{\phi}$ are the parameters of the neural network involved.

Eichelsdörfer et al. (2021) showed that the performance of HNNs, especially in the Small Data regime, can be improved by adding an additional, physics-informed, regularization term which penalizes the difference between the learned Hamiltonian $H_{\boldsymbol{\phi}}$ and target values of the total energy level of the system $\hat{H}$. This additional information has to be collected only once for each trajectory. The augmented loss function $\hat{\mathcal{L}}(\boldsymbol{\phi})$ is given by

$$\hat{\mathcal{L}}(\boldsymbol{\phi}) = \frac{1}{N}\sum_{i=1}^{N} \left| \frac{\partial H_{\boldsymbol{\phi}}(\boldsymbol{q}^i, \boldsymbol{p}^i)}{\partial \boldsymbol{p}} - \dot{\boldsymbol{q}}^i \right|^2 + \left| \frac{\partial H_{\boldsymbol{\phi}}(\boldsymbol{q}^i, \boldsymbol{p}^i)}{\partial \boldsymbol{q}} + \dot{\boldsymbol{p}}^i \right|^2 + \lambda_H (H_{\boldsymbol{\phi}}(\boldsymbol{q}^i, \boldsymbol{p}^i) - \hat{H}_i)^2 \tag{2.56}$$

This augmented loss function involves an additional hyperparameter λ_H, which is problem specific and has to be determined by cross-validation or by using an augmented Lagrangian method during the optimization of the loss function.

3. Summary of Publications

3.1. Paper A: Incorporating physical constraints in a deep probabilistic machine learning framework for coarse-graining dynamical systems

3.1.1. Summary

This paper introduces a data-based probabilistic machine learning framework for the effective discovery of coarse-grained (CG) models of high-dimensional dynamical systems. The proposed algorithm is especially suitable for multiscale problems and enables the quantification of predictive uncertainty. We formulate the coarse-graining process by employing a probabilistic state-space model and account for physical constraints with the novel concept of virtual observables. The primary utility of such constraints stems from the undisputed physical laws such as conservation of mass, energy etc. that they represent. We assume that these constraints are virtually observed and thus give rise to a virtual likelihood that can be seamlessly integrated into the computation of the Bayesian posterior. Apart from leading to physically realistic predictions, they can significantly reduce the requisite amount of training data which for high-dimensional, multiscale systems are expensive to obtain (Small Data regime). We, moreover, employ deep neural nets in combination with probabilistic inference tools to identify the coarse-grained variables and their evolution model without a restriction (fine-to-coarse) projection and the availability of time-derivatives of the state variables. We advocate a sparse Bayesian learning perspective that avoids overfitting and reveals the most salient features in the coarse-grained evolution law. The formulation adopted enables the quantification of predictive uncertainty due to information loss which is a crucial and often neglected component in the CG process. Furthermore, we are capable of reconstructing the evolution of the full, high-dimensional system, and therefore, the observables of interest need not be selected a priori. We demonstrate the efficacy of the proposed framework by applying it to systems of interacting particles with advection-diffusion and Burger's type dynamics and a series of images of a nonlinear pendulum. In all cases, we identify the underlying coarse dynamics and can generate extrapolative predictions, including the formation and propagation of a shock for the particle systems and a stable trajectory in the phase space for the pendulum.

3.1.2. Contribution

I am the first author of this paper. I developed the methodological framework together with Phaedon-Stelios Koutsourelakis. The implementation was done by me, and I carried out all simulations. I discussed the results with Phaedon-Stelios Koutsourelakis and then

drafted the manuscript. Phaedon-Stelios Koutsourelakis contributed to the manuscript and reviewed the draft.

3.2. Paper B: Physics-aware, deep probabilistic modeling of multiscale dynamics in the Small Data regime

3.2.1. Summary

This paper presents a probabilistic perspective that simultaneously identifies predictive, lower-dimensional CG variables as well as their dynamics given training data of high-dimensional, multiscale systems. We make use of the expressive ability of deep neural networks in order to represent the right-hand side of the CG evolution law and to represent a coarse-to-fine map from the lower-dimensional coordinates to the high-dimensional system. Furthermore, we incorporate domain knowledge that is very often available in the form of physical constraints (e.g.conservation laws) in our model with the concept of virtual observables. The usage of virtual observables enables physically realistic predictions and can significantly reduce the requisite amount of training data. This enables us to work in the Small Data regime and reduces the amount of required, computationally expensive multiscale simulations, although the model employs the aforementioned two deep neural networks. We demonstrate the efficacy of the proposed framework in a high-dimensional system of moving particles. We are capable of making accurate extrapolative predictions and quantifying the predictive uncertainty as well as reconstructing the evolution of the full, fine-scale system, which allows us to select the quantities of interest a posteriori.

3.2.2. Contribution

I am the first author of this paper. I developed the methodological framework together with Phaedon-Stelios Koutsourelakis. The implementation and simulations were done by me. I discussed the results with Phaedon-Stelios Koutsourelakis and then drafted the manuscript. Phaedon-Stelios Koutsourelakis contributed to the manuscript and reviewed the draft.

3.3. Paper C: Physics-aware, probabilistic model order reduction with guaranteed stability

3.3.1. Summary

We propose a generative framework for learning an effective, lower-dimensional, coarse-grained dynamical model that is predictive of the fine-grained system's long-term evolution but also its behavior under different initial conditions. The model only requires (small amounts of) time-series data from a high-dimensional, fine-grained, multiscale dynamical system. We target those high-dimensional, fine-grained models as they arise in physical applications (e.g. molecular dynamics, agent-based models), the dynamics of which are strongly

non-stationary but their transition to equilibrium is governed by unknown slow processes which are largely inaccessible by brute-force simulations. The generative framework proposed employs a flexible prior on the complex plane for the latent, slow processes and an intermediate layer of physics-motivated latent variables that reduces reliance on data and imbues inductive bias. The flexible prior on the complex plane is inspired by probabilistic Slow Feature analysis (Turner and Sahani, 2007) and enables us to rank the identified processes according to their slowness. It moreover makes use of a discretized version of an Ornstein-Uhlenbeck process in order to be able to restrict the eigenvalues of the system to negative real parts and thus restrict the coarse-grained system to a dynamic with guaranteed stability. We address simultaneously the tasks of dimensionality reduction and model estimation and thus are not required to a priori define projection operators or encoders. We demonstrate its efficacy and accuracy in multiscale physical systems of particle dynamics where probabilistic, long-term predictions of phenomena not contained in the training data are produced. We are able to make predictions for much more time steps than contained in the training data and our model converges to the correct steady state.

3.3.2. Contribution

I am the first author of this paper. I developed the methodological framework together with Phaedon-Stelios Koutsourelakis. The implementation and all simulations were done by me. I discussed the results with Phaedon-Stelios Koutsourelakis and then drafted the manuscript. Phaedon-Stelios Koutsourelakis contributed to the manuscript and reviewed the draft.

3.4. Paper D: Semi-supervised Invertible Neural Operators for Bayesian Inverse Problems

3.4.1. Summary

This paper employs physics-informed Neural Operators in the context of high-dimensional, Bayesian inverse problems. Neural Operators offer a powerful, data-driven tool for solving parametric PDEs by learning operators, i.e. maps between infinite-dimensional function spaces. In particular, we extend the Deep Operator Network framework by employing a realNVP architecture which yields an invertible and differentiable map between the parametric input and the branch net output. This allows us to not only solve forward and inverse problems with one neural network architecture but also to construct accurate approximations of the full posterior in Bayesian inverse problems, which can be readily adapted irrespective of the number of observations and the magnitude of the observation noise. As a result, no additional forward solves are required, nor is there any need for costly sampling procedures. We demonstrate the efficacy and accuracy of the proposed methodology in the context of inverse problems based on an anti-derivative equation, reaction-diffusion dynamics, and flow through porous media. We employ a physics-informed loss term during training and train our model for each example with unlabeled training data and no or only a small amount of labeled training data.

3.4.2. Contribution

I am the first author of this paper. I developed the methodological framework together with Paris Perdikaris and Phaedon-Stelios Koutsourelakis. Paris Perdikaris especially contributed regarding the invertible DeepoNets and their numerical parameters, whereas Phaedon-Stelios Koutsourelakis contributed regarding their application to Bayesian Inverse Problems. The implementation and the simulations were done by me. I discussed the results with Paris Perdikaris and Phaedon-Stelios Koutsourelakis and then drafted the manuscript. Phaedon-Stelios Koutsourelakis and Paris Perdikaris reviewed the draft and contributed to the manuscript.

4. Conclusions & Outlook

4.1. Conclusions and Discussion

In this thesis, different frameworks have been presented that enable probabilistic machine learning for physical systems in the Small Data regime. These frameworks incorporate physical constraints as virtual observables (Kaltenbach and Koutsourelakis, 2020a, 2021a), guarantee stability (Kaltenbach and Koutsourelakis, 2021b) or involve invertible Neural Operators to learn both forward and inverse operators simultaneously in a semi-supervised setting (Kaltenbach et al., 2022).

All frameworks have in common that we are employing a Bayesian perspective and thus are able to quantify the predictive uncertainty. This is a large advantage compared to deterministic machine learning frameworks that are, in general, only capable of computing point estimates. The information loss during model order reduction is also incorporated into the aforementioned uncertainty as we simultaneously carry out the model and dimensionality reduction. A separate model order reduction step would make consistent uncertainty quantification much more difficult. However, it would allow us to split the algorithm into two parts and thus also split the computational workload.
Due to the probabilistic perspective employed, we have to compute a Bayesian posterior and carry out approximate inference. We have therefore developed inference algorithms that are suitable for the Small Data regime and can handle complex random variables and predefined dependencies between variables. These algorithms are based on variational inference (Blei et al., 2017), but we could also employ advanced gradient-based MCMC methods such as the No-U-Turn Sampler (NUTS) (Hoffman and Gelman, 2014). The latter would be computationally more challenging due to the high-dimensional posterior distributions of the models, but would provide asymptotically exact results. In case MCMC is employed, the computations involved should be done in a massively parallel way.

Moreover, all models can be trained in the Small Data regime, which is advantageous as data from physical systems can be sparse and expensive to obtain. Numerical simulations are computationally costly as the dimensions involved are, in general, very high, whereas the time scales of interest are very small. Experiments are equally expensive and can be infeasible given the spatial scales of the system. The frameworks do not require derivatives as input data in contrast to, for instance, the SINDy framework (Brunton et al., 2016). As derivatives are for many systems not readily available and can be noisy, this is, in our opinion, another advantage of our physics-aware probabilistic machine learning algorithms.

The incorporation of physical knowledge is one of the key aspects of this work. While we firmly believe that inductive bias improves machine learning algorithms, it can also be

problematic if the inductive bias should be wrong. For instance, if we enforce an incorrect physical model within our machine learning algorithm, we would potentially need a very large amount of data to overrule the inductive bias. If the structure of the neural network architecture is designed according to the incorrect physical model, even a larger amount of data can, in general, not correct the wrong inductive bias, and the predictive quality of the model will be very low. We have developed the novel concept of Virtual Observables (Kaltenbach and Koutsourelakis, 2020a) to account for physical constraints. This allows us to integrate the constraints seamlessly into our fully probabilistic machine learning framework and decide how strong the constraints are enforced by setting the relevant standard deviation of the virtual observable. This shares similarities with Raissi et al. (2019) as the virtual observables act as a regularization term similar to the physics-informed part of the loss function used for PINNs. The disadvantage of this approach is that we can not guarantee that the constraints are exactly enforced as the virtual likelihood is optimized together with the likelihood based on the data and the solution is a trade-off between both likelihoods. Another possibility would be to directly integrate the constraints into the neural network architecture to ensure that the constraint is enforced all the time. This is possible for energy conservation with Hamiltonian Neural Networks (Greydanus et al., 2019) or for symmetry invariance with equivariant networks (Wang et al., 2022b). Wang et al. (2022a) have shown that the concept of equivariant neural networks can even be extended to non-perfect symmetries.

In Kaltenbach and Koutsourelakis (2021b) we have shown that imposing some constraints regarding the dynamics of the reduced-order variables can significantly enhance the predictive accuracy for long-term predictions. In our work, the obtained reduced-order dynamic not only guarantees stability but also provides interpretability. In particular, we can rank the obtained processes according to their slowness due to the chosen prior. As the dynamics of the reduced order variables are restricted to a discretized Ornstein-Uhlenbeck process, our coarse-to-fine map has to be able to map from reduced order variables with such a dynamic to the high-dimensional system. This can lead to a more complex coarse-to-fine map than in the case of a more flexible dynamic. We, therefore, introduced an intermediate layer of physics-motivated variables to facilitate the learning of this map. Moreover, we believe that the gained stability and interpretability outweigh the slightly more complex coarse-to-fine map. The interpretability obtained with this method is linked to a priori defining a parametrized reduced-order dynamics. In our approach we choose an Ornstein-Uhlenbeck process to directly enforce stability, but choices based on the Koopman-formalism (Pan et al., 2021) or the Mori-Zwanzig approach (Menier et al., 2023) are also possible. Pan et al. (2021) learn the most informative Koopman-invariant subspace, whereas Menier et al. (2023) combine a linear reduced-order dynamics with a non-linear closure term. These methods are all different from for instance the Neural ODEs Chen et al. (2018) or the LED framework Vlachas et al. (2022) which employ a neural network for the reduced-order dynamics without any restrictions and are therefore not interpretable. The LED framework is based on the equation-free framework (Kevrekidis et al., 2004) and, due to the high flexibility of the LSTM employed for the reduced-order dynamics, can be a more promising approach in case a large amount of training data is available and interpretability or stability are not required.

Our work on invertible Neural Operators shows the large potential of invertible architectures in machine learning for physical systems. We are not only able to obtain both a forward and inverse operator for the targeted physical system, but we are also able to use the invertible DeepONet to solve Bayesian Inverse problems in a very fast and accurate way. In case the DeepONet is trained on unlabeled training data only, we can compute an approximate posterior for a Bayesian inverse problem without ever having to solve the underlying PDE. The addition of some labeled training data is rewarded with higher predictive accuracy. We strongly believe that invertible Neural Operators have, therefore, immense potential regarding Bayesian inverse problems. In our work, we incorporated the invertible neural network into a DeepONet, but invertible architectures could enhance other frameworks as well. We note that the DeepONet approach was developed to learn Neural Operators and does not necessarily learn a lower-dimensional representation and thus not uses any latent variables. All aforementioned approaches, however, target dimensional reduction and learning a low-dimensional latent space. In Table 4.1 a comparison between DeepONets and a framework based on Autoencoders such as LED can be found.

The DeepONet framework has been extended recently and some of these advantages could also be used to further improve the invertible DeepONet architecture. For instance, Wang et al. (2022c) suggested account for causality during training which could be directly incorporated into the ivnertible architecture. Seidman et al. (2022) replaced the dot product which combines the output of the branch and the trunk network with a non-linear mapping allowing to also learn non-linear manifolds. This is unfortunately with the current invertible DeepONet architecture not possible as this architecture requires currently a linear combination in the last layer. Moreover, other Neural Operators based on Variational Autoencoders Seidman et al. (2023) have recently been introduced.

Another line of research pertains to the interpretability of DeepONets, which could be improved using a Bayesian formalism (Moya et al., 2023) or incorporating elements from Singular-Value-Decomposition Venturi and Casey (2023). This is currently a disadvantage of our invertible architecture whose could be further improved.

	DeepONet	LED and Autoencoder-based architectures
Target	Representing a solution operator for a parametrized PDE	Learning a reduced order dynamical representation of the system
Architecture Summary	Two neural networks: A branch network processes the input parameters whereas a trunk network processes the spatio-temporal location. The final solution is a dot product of both outputs of these neural networks. (for more details see Section 2.6.2)	Three neural networks: An encoder network maps the high-dimensional data to a latent space, a decoder network maps the latent representation back to the high-dimensional space and a third neural network is used to represent the temporal dynamics for the latent space. (see Section 2.6.3)
Latent representation	No latent representation is learned	Yes
Temporal dynamics	No explicit temporal dynamics are learned	Yes
Parametric dependency	The parameters of the system are the input for the branch network and can be changed during predictions	Not possible with the vanilla version, the parametric dependency would need to be added to the neural networks involved.
Invertible architecture	Yes, invertible DeepONets are possible	No, all temporal dynamics employed are non invertible.

Table 4.1: Comparison of DeepONets and LED

4.2. Outlook

We have presented physics-aware probabilistic machine learning algorithms that can be trained in the Small Data regime and have shown promising results in the application examples. However, some challenges remain currently unsolved:

4.2.1. Outstanding challenges

- One of the most difficult questions in model order reduction/ coarse-graining is how to identify the right number of reduced order state variables that are needed. Whereas we have for physical problems in general an idea which variables would be suitable, we never have a guarantee that those variables are sufficient. Therefore, augmenting our frameworks such that they could discover additional and possibly non-physical reduced order variables could be a very interesting direction for future work. In Kaltenbach and Koutsourelakis (2021b) we have employed an intermediate layer of physical variables, which could be combined with a small amount of other non-physical states that are constructed solely based on the data.
- Another outstanding challenge pertains to the interpretability of our machine learning framework. Although in Kaltenbach and Koutsourelakis (2020a, 2021b) we have already presented approaches that are interpretable up to a certain degree, we would like to increase this even further. Especially the DeepONet formalism does not provide much interpretability besides the split into trunk and branch network.
- In Kaltenbach and Koutsourelakis (2021b) we have developed a model order reduction method with guaranteed stability. However, the reduced order temporal dynamic is discrete, which can be problematic for sparse or irregularly sampled data. We, therefore, think that an extension to a continuous dynamic could enhance the potential of this method even further.
- Apart from the invertible DeepONet, our frameworks can not account for dependencies of the targeted system's output on input parameters. However, this could be changed by integrating these input parameters into our reduced-order models. They could either directly affect the dynamics of the reduced-order model or the mapping between the reduced-order model and the high-dimensional system. Such parametric dependencies have for instance been studied in Kalia et al. (2021).
- We have developed surrogates that rely on Deep Neural Networks to represent the solutions of PDEs. For large scale systems the training of these surrogates can be computationally expensive. Recently, Karnakov et al. (2022) have shown that a machine learning framework without neural networks can be computationally more efficient due to the reduced number of model parameters. The idea proposed could, therefore, potentially be integrated into our frameworks and could lead to faster algorithms that are potentially able to work with even less data due to the aforementioned reduced number of parameters.
- The developed frameworks have so far shown very good results for all the systems they have been applied to. There are however still many application areas left where the frameworks have not been applied so far. An additional challenge would be applying the frameworks to more areas in order to show that they are general applicable. This

could involve small modifications to tailor them to the respective application area. For example, for medical applications interpretability would be even more important.

4.2.2. Future work

We propose the following next steps to resolve some of the aforementioned outstanding challenges:

- We want to use the ELBO, i.e. the lower bound to the model evidence, as a possible selection criterion for how many reduced-order variables are needed and to select the best set of reduced order variables. This could address the issue for how to select the coarse-grained variables. We suggest to start with a set of physics-inspired reduced-order variables and then subsequently add additional latent variables until the improvement in the ELBO is very small.
- The learned dynamics for our reduced order models have currently been based on either predefined feature functions or neural networks. Yin et al. (2021) have proposed a framework that augments incomplete physical knowledge with machine learning. A similar approach could be applied to learning the dynamics of reduced order variables as we have some knowledge available due to fundamental physical laws. This knowledge is, in general, not enough to entirely prescribe the dynamics but can help us to select specific essential components of the dynamics as the physical system should not be able to violate energy conservation and usually converges to some sort of equilibrium. We therefore want to split our reduced-order dynamics into on part based on a physical law and a purely data-driven part and thus increase the interpretability of our reduced-order dynamics. To accomplish this we have to solve a closure problem in the reduced-order dynamics. The closure term can be modeled with a neural network. We suggest to apply a weight regularization to this network such that as much as possible of the dynamics are represented by the given physical law.
- Another possibility to increase the interpretability is to base our neural network architecture on the Mori-Zwanzig formalism. The Mori-Zwanzig formalism provides an exact equation for the dynamics of a reduced-order. An architecture inspired by this formalism would split the reduced-order dynamics in a linear part and a non-linear part representing the memory term of the Mori-Zwanzig formalism. To ensure computational feasibility, we recommend to include only a limited amount of previous time steps into this memory computation.
- Employing a continuous dynamics in the latent space can be done by learning a Stochastic Differential Equations. This requires modifications to our inference algorithms. A possible starting point is described in Li et al. (2020). Their methodology could be adapted for our frameworks to ensure fast and efficient training of an SDE. An alternative approach for a continuous latent space could employ a Gaussian Process.
- Another goal for a future work is to add a parametric dependency to the presented algorithms. This could be done by adding another parameter in the temporal dynamics. In case a LSTM was employed, we could add any additional parameters to the internal hidden state. In a more general setting, we could add another neural network that processes the parameters before they are combined with the neural network responsible for the temporal dynamics. This idea is based on the DeepONet structure (Lu et al.,

2021). According to Seidman et al. (2022) a non-linear recombination between the two neural network architectures could be beneficial.

- We plan to modify the autoencoder that is currently used in our reduced-order frameworks. In more detail we want to introduce a novel mutltiscale autoencoder that is able to process input data at different resolutions. For instance for physical simulations, all simulations rarely have exact the same configurations. Therefore a learning framework that can be trained using data with different discretizations can be beneficial. Moreover, we strongly belief that such a framework can significantly increase the interpretability as the influence of each latent variable on the the different resolution levels can be revealed.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org, 2015. URL `https://www.tensorflow.org/`.

Hervé Abdi and Lynne J. Williams. Principal component analysis. WIREs Computational Statistics, 2(4):433–459, 2010. ISSN 1939-0068. doi: 10.1002/wics.101.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261, 2018. doi: 10.48550/arXiv.1806.01261.

Thomas Bayes. An essay towards solving a problem in the doctrine of chances. Philosophical Transactions of the Royal Society of London, 53:370–418, January 1763. doi: 10.1098/rstl.1763.0053.

Richard E Bellman. Dynamic programming. Princeton university press, 2010.

Gal Berkooz, Philip Holmes, and John L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. Annual review of fluid mechanics, 25:539–575, 1993. doi: 10.1146/annurev.fl.25.010193.002543.

J. Besag. Comments on representations of knowledge in complex systems by u. grenander and mi miller. Journal of the Royal Statistical Society, pages 591–592, 1994.

Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. Model Reduction and Neural Networks for Parametric PDEs. arXiv preprint arXiv:2005.03180, June 2021. doi: 10.48550/arXiv.2005.03180.

Christopher M. Bishop and Nasser M. Nasrabadi. Pattern recognition and machine learning, volume 4. Springer, 2006.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. Journal of the American Statistical Association, 112(518):859–877, April 2017. ISSN 0162-1459. doi: 10.1080/01621459.2017.1285773.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs. Software available from https://github.com/google/jax, 2018. URL `http://github.com/google/jax`.

Andy Brock, Soham De, Samuel L. Smith, and Karen Simonyan. High-Performance Large-Scale Image Recognition Without Normalization. In Proceedings of the 38th International Conference on Machine Learning, pages 1059–1071. PMLR, July 2021. URL `https://proceedings.mlr.press/v139/brock21a.html`. ISSN: 2640-3498.

Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences, 113(15):3932–3937, April 2016. doi: 10.1073/pnas.1517384113.

Kathleen P. Champion, Steven L. Brunton, and J. Nathan Kutz. Discovery of Nonlinear Multiscale Systems: Sampling Strategies and Embeddings. SIAM Journal on Applied Dynamical Systems, 18(1):312–333, January 2019. doi: 10.1137/18M1188227.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In Advances in neural information processing systems, volume 31, Montréal, Canada, 2018. URL `https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf`.

Alexandre Chorin and Panagiotis Stinis. Problem reduction, renormalization, and memory. Communications in Applied Mathematics and Computational Science, 1(1):1–27, May 2007. ISSN 2157-5452. doi: 10.2140/camcos.2006.1.1.

Georges Louis Leclerc comte de Buffon. Histoire naturelle: générale et particulière, volume 16. De l'Imprimerie royale, 1770.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian Neural Networks. arXiv preprint arXiv:2003.04630, July 2020. doi: 10.48550/arXiv.2003.04630.

Pierre Simon de Laplace. Théorie analytique des probabilitÃ©s, volume 7. Courcier, 1820.

Ward Edwards, Harold Lindman, and Leonard J. Savage. Bayesian statistical inference for psychological research. Psychological Review, 70(3):193–242, 1963. ISSN 1939-1471. doi: 10.1037/h0044139.

Jonas Eichelsdörfer, Sebastian Kaltenbach, and Phaedon-Stelios Koutsourelakis. Physics-enhanced Neural Networks in the Small Data Regime. In Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021), 2021. URL `https://ml4physicalsciences.github.io/2021/files/NeurIPS_ML4PS_2021_77.pdf`.

Siegfried Flügge. Practical Quantum Mechanics. Springer Science & Business Media, Berlin, Heidelberg, 2012. ISBN 978-3-540-65035-5 978-3-642-61995-3. doi: 10.1007/978-3-642-61995-3.

Vincent Fortuin, Adriá Garriga-Alonso, Sebastian W. Ober, Florian Wenzel, Gunnar Rätsch, Richard E. Turner, Mark van der Wilk, and Laurence Aitchison. Bayesian Neural Network Priors Revisited. arXiv preprint arXiv:2102.06571, March 2022. doi: 10.48550/arXiv.2102.06571. Issue: arXiv:2102.06571 arXiv:2102.06571 [cs, stat].

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological cybernetics, 36(4):193–202, 1980.

Yarin Gal, Petros Koumoutsakos, Francois Lanusse, Gilles Louppe, and Costas Papadimitriou. Bayesian uncertainty quantification for machine-learned models in physics. Nature Reviews Physics, 4(9):573–577, September 2022. ISSN 2522-5820. doi: 10.1038/s42254-022-00498-4.

Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. Nature, 521 (7553):452–459, May 2015. ISSN 1476-4687. doi: 10.1038/nature14541.

Craig Gin, Bethany Lusch, Steven L. Brunton, and J. Nathan Kutz. Deep learning models for global coordinate transformations that linearise PDEs. European Journal of Applied Mathematics, 32(3):515–539, June 2021. ISSN 0956-7925, 1469-4425. doi: 10.1017/S0956792520000327.

Dror Givon, Raz Kupferman, and Andrew Stuart. Extracting macroscopic dynamics: model problems and algorithms. Nonlinearity, 17(6):R55–R127, August 2004. ISSN 0951-7715. doi: 10.1088/0951-7715/17/6/R01.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. `http://www.deeplearningbook.org`.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian Neural Networks. In Advances in Neural Information Processing Systems, volume 32, 2019. URL `https://proceedings.neurips.cc/paper/2019/file/26cd8ecadce0d4efd6cc8a8725cbd1f8-Paper.pdf`.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. Biometrika, 57(1):97–109, 1970. ISSN 0006-3444. doi: 10.1093/biomet/57.1.97.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

Matthew D. Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. Journal of Machine Learning Research, 15(47): 1593–1623, 2014. URL `http://jmlr.org/papers/v15/hoffman14a.html`.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. Journal of Machine Learning Research, 14:1303–1347, 2013.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5):359–366, January 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. Machine Learning, 110(3):457–506, March 2021. ISSN 1573-0565. doi: 10.1007/s10994-021-05946-3.

Kadierdan Kaheman, J. Nathan Kutz, and Steven L. Brunton. SINDy-PI: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 476(2242):20200279, October 2020. doi: 10.1098/rspa.2020.0279.

Manu Kalia, Steven L Brunton, Hil GE Meijer, Christoph Brune, and J Nathan Kutz. Learning normal form autoencoders for data-driven discovery of universal, parameter-dependent governing equations. arXiv preprint arXiv:2106.05102, 2021. doi: 10.48550/arXiv.2106.05102.

S. Kaltenbach and P.-S. Koutsourelakis. Physics-Aware, Deep Probabilistic Modeling of Multiscale Dynamics in the Small Data Regime. 14th WCCM-ECCOMAS Congress 2020, March 2021a. ISSN 2696-6999. doi: 10.23967/wccm-eccomas.2020.280.

Sebastian Kaltenbach and Phaedon-Stelios Koutsourelakis. Incorporating physical constraints in a deep probabilistic machine learning framework for coarse-graining dynamical systems. Journal of Computational Physics, 419, October 2020a. ISSN 0021-9991. doi: 10.1016/j.jcp.2020.109673.

Sebastian Kaltenbach and Phaedon-Stelios Koutsourelakis. Physics-aware, data-driven discovery of slow and stable coarse-grained dynamics for high-dimensional multiscale systems. In 1st NeurIPS workshop on Interpretable Inductive Biases and Physically Structured Learning, 2020b. URL `https://inductive-biases.github.io/papers/22.pdf`.

Sebastian Kaltenbach and Phaedon Stelios Koutsourelakis. Physics-aware, probabilistic model order reduction with guaranteed stability. In International Conference on Learning Representations (ICLR), March 2021b. URL `https://openreview.net/forum?id=vyY0jnWG-tK`.

Sebastian Kaltenbach, Paris Perdikaris, and Phaedon-Stelios Koutsourelakis. Semi-supervised Invertible DeepONets for Bayesian Inverse Problems. arXiv preprint arXiv:2209.02772, September 2022. doi: 10.48550/arXiv.2209.02772.

Petr Karnakov, Sergey Litvinov, and Petros Koumoutsakos. Optimizing a DIscrete Loss (ODIL) to solve forward and inverse problems for partial differential equations using machine learning tools. arXiv preprint arXiv:2205.04611, May 2022. doi: 10.48550/arXiv.2205.04611.

Ioannis G Kevrekidis, C William Gear, and Gerhard Hummer. Equation-free: The computer-aided analysis of complex multiscale systems. AIChE Journal, 50(7):1346–1355, 2004.

Ivica Kičić, Pantelis R Vlachas, Georgios Arampatzis, Michail Chatzimanolakis, Leonidas Guibas, and Petros Koumoutsakos. Adaptive learning of effective dynamics: Adaptive real-time, online modeling for complex systems. arXiv preprint arXiv:2304.01732, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980, January 2014. doi: 10.48550/arXiv.1412.6980.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. arXiv preprint arXiv:1312.6114, May 2014. doi: 10.48550/arXiv.1312.6114.

Georgios Kissas, Jacob H Seidman, Leonardo Ferreira Guilhoto, Victor M Preciado, and George J Pappas. Learning Operators with Coupled Attention. Journal of Machine Learning Research, 23:1–63, 2022. URL https://www.jmlr.org/papers/volume23/21-1521/21-1521.pdf.

Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? Does it matter? Structural Safety, 31(2):105–112, March 2009. ISSN 0167-4730. doi: 10.1016/j.strusafe.2008.06.020.

Stefan Klus, Feliks Nüske, Péter Koltai, Hao Wu, Ioannis Kevrekidis, Christof Schütte, and Frank Noé. Data-Driven Model Reduction and Transfer Operator Approximation. Journal of Nonlinear Science, 28(3):985–1010, June 2018. ISSN 1432-1467. doi: 10.1007/s00332-017-9437-7.

Dmitri Kondrashov, Mickaël D. Chekroun, and Michael Ghil. Data-driven non-Markovian closure models. Physica D: Nonlinear Phenomena, 297:33–55, March 2015. ISSN 0167-2789. doi: 10.1016/j.physd.2014.12.005.

B. O. Koopman. Hamiltonian Systems and Transformation in Hilbert Space. Proceedings of the National Academy of Sciences, 17(5):315–318, May 1931. doi: 10.1073/pnas.17.5.315.

P. S. Koutsourelakis, N. Zabaras, and M. Girolami. Special Issue: Big data and predictive computational modeling. Journal of Computational Physics, 321:1252–1254, September 2016. ISSN 0021-9991. doi: 10.1016/j.jcp.2016.03.028.

Phaedon-Stelios Koutsourelakis and Elias Bilionis. Scalable Bayesian Reduced-Order Models for Simulating High-Dimensional Multiscale Dynamical Systems. Multiscale Modeling & Simulation, 9(1):449–485, January 2011. ISSN 1540-3459. doi: 10.1137/100783790.

S. Kullback and R. A. Leibler. On Information and Sufficiency. The Annals of Mathematical Statistics, 22(1):79–86, 1951. ISSN 0003-4851. URL https://www.jstor.org/stable/2236703.

I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. IEEE Transactions on Neural Networks, 9(5):987–1000, September 1998. ISSN 1941-0093. doi: 10.1109/72.712178.

Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. In M.A. Arbib, editor, The handbook of brain theory and neural networks, volume 3361. MIT Press, 1995. Publisher: Cambridge, MA USA.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551, 1989.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553): 436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539.

Kookjin Lee and Kevin T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. Journal of Computational Physics, 404, March 2020. ISSN 0021-9991. doi: 10.1016/j.jcp.2019.108973.

Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable Gradients for Stochastic Differential Equations. In Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, volume 108, pages 3870–3882. PMLR, June 2020. URL `https://proceedings.mlr.press/v108/li20i.html`. ISSN: 2640-3498.

Yingzhen Li and Yarin Gal. Dropout Inference in Bayesian Neural Networks with Alpha-divergences. In Proceedings of the 34th International Conference on Machine Learning, volume 70, pages 2052–2061. PMLR, July 2017. URL `https://proceedings.mlr.press/v70/li17a.html`. ISSN: 2640-3498.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. arXiv prerint arXiv:2010.08895, May 2021. doi: 10.48550/arXiv.2010.08895.

Jun S. Liu and Rong Chen. Sequential Monte Carlo Methods for Dynamic Systems. Journal of the American Statistical Association, 93(443):1032–1044, September 1998. ISSN 0162-1459. doi: 10.1080/01621459.1998.10473765.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. Nature Machine Intelligence, 3(3):218–229, March 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00302-5.

Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. Nature Communications, 9(1):4950, November 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-07210-0.

Michael Lutter, Christian Ritter, and Jan Peters. Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning. arXiv preprint arXiv:1907.04490, July 2019. doi: 10.48550/arXiv.1907.04490.

David J. C. Mackay. Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks. Network: Computation in Neural Systems, 6(3):469–505, January 1995. ISSN 0954-898X. doi: 10.1088/0954-898X/6/3/011.

Luca Martino, VÃctor Elvira, and Francisco Louzada. Effective sample size for importance sampling based on discrepancy measures. Signal Processing, 131:386–401, February 2017. ISSN 0165-1684. doi: 10.1016/j.sigpro.2016.08.025.

Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4):115–133, December 1943. ISSN 1522-9602. doi: 10.1007/BF02478259.

Emmanuel Menier, Michele Alessandro Bucci, Mouadh Yagoubi, Lionel Mathelin, and Marc Schoenauer. Cd-rom: Complemented deep-reduced order model. Computer Methods in Applied Mechanics and Engineering, 410:115985, 2023.

Nicholas Metropolis and S. Ulam. The Monte Carlo Method. Journal of the American Statistical Association, 44:335–341, September 1949. ISSN 0162-1459. doi: 10.1080/01621459.1949.10483310.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. The Journal of Chemical Physics, 21:1087–1092, June 1953. ISSN 0021-9606. doi: 10.1063/1.1699114.

Hazime Mori. Transport, Collective Motion, and Brownian Motion. Progress of Theoretical Physics, 33:423–455, March 1965. ISSN 0033-068X. doi: 10.1143/PTP.33.423.

Christian Moya, Shiqi Zhang, Guang Lin, and Meng Yue. Deeponet-grid-uq: A trustworthy deep operator framework for predicting the power grid's post-fault trajectories. Neurocomputing, 535:166–182, 2023. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.03.015. URL https://www.sciencedirect.com/science/article/pii/S0925231223002503.

Kevin P. Murphy. Machine learning: a probabilistic perspective. MIT press, 2012.

Radford M. Neal and Geoffrey E. Hinton. A View of the Em Algorithm that Justifies Incremental, Sparse, and other Variants. In Learning in Graphical Models, NATO ASI Series, pages 355–368. Springer Netherlands, Dordrecht, 1998. ISBN 978-94-011-5014-9. doi: 10.1007/978-94-011-5014-9_12.

Manfred Opper and David Saad. Advanced mean field methods: Theory and practice. MIT press, 2001.

John Paisley, David Blei, and Michael Jordan. Variational Bayesian Inference with Stochastic Search. arXiv preprint arXiv:1206.6430, June 2012. doi: 10.48550/arXiv.1206.6430.

Shaowu Pan and Karthik Duraisamy. Physics-Informed Probabilistic Learning of Linear Embeddings of Nonlinear Dynamics with Guaranteed Stability. SIAM Journal on Applied Dynamical Systems, 19(1):480–509, January 2020. doi: 10.1137/19M1267246.

Shaowu Pan, Nicholas Arnold-Medabalimi, and Karthik Duraisamy. Sparsity-promoting algorithms for the discovery of informative koopman-invariant subspaces. Journal of Fluid Mechanics, 917:A18, 2021.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In Neural Information Processing System (NeurIPS, October 2017. URL https://openreview.net/forum?id=BJJsrmfCZ.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. 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, February 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045.

Maximilian Rixner and Phaedon-Stelios Koutsourelakis. A probabilistic generative model for semi-supervised training of coarse-grained surrogates and enforcing physical constraints through virtual observables. Journal of Computational Physics, 434:110218, June 2021. ISSN 0021-9991. doi: 10.1016/j.jcp.2021.110218.

Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. The Annals of Mathematical Statistics, 22:400–407, 1951. ISSN 0003-4851. URL `https://www.jstor.org/stable/2236626`.

Gareth O. Roberts and Jeffrey S. Rosenthal. Optimal scaling for various Metropolis-Hastings algorithms. Statistical Science, 16(4):351–367, November 2001. ISSN 0883-4237, 2168-8745. doi: 10.1214/ss/1015346320.

Gareth O. Roberts and Richard L. Tweedie. Exponential Convergence of Langevin Distributions and Their Discrete Approximations. Bernoulli, 2:341–363, 1996. ISSN 1350-7265. doi: 10.2307/3318418.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. arXiv preprint arXiv:2205.11487, May 2022. doi: 10.48550/arXiv.2205.11487.

Markus Schöberl, Nicholas Zabaras, and Phaedon-Stelios Koutsourelakis. Predictive coarse-graining. Journal of Computational Physics, 333:49–77, March 2017. ISSN 0021-9991. doi: 10.1016/j.jcp.2016.10.073.

Jacob Seidman, Georgios Kissas, Paris Perdikaris, and George J Pappas. Nomad: Nonlinear manifold decoders for operator learning. Advances in Neural Information Processing Systems, 35:5601–5613, 2022.

Jacob H Seidman, Georgios Kissas, George J Pappas, and Paris Perdikaris. Variational autoencoding neural operators. arXiv preprint arXiv:2302.10351, 2023.

Cristian Tomasetti, Lu Li, and Bert Vogelstein. Stem cell divisions, somatic mutations, cancer etiology, and cancer prevention. Science, 355(6331):1330–1334, March 2017. doi: 10.1126/science.aaf9011.

Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sebastien Racaniere, Aleksandar Botev, and Irina Higgins. Hamiltonian Generative Networks. arXiv preprint arXiv:1909.13789, February 2020. doi: 10.48550/arXiv.1909.13789.

Richard Turner and Maneesh Sahani. A maximum-likelihood interpretation for slow feature analysis. Neural computation, 19(4):1022–1038, 2007. doi: 10.1162/neco.2007.19.4.1022.

Nicolaas Godfried Van Kampen. Stochastic processes in physics and chemistry, volume 1. Elsevier, 1992.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In Advances in Neural Information Processing Systems, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Simone Venturi and Tiernan Casey. Svd perspectives for augmenting deeponet flexibility and interpretability. Computer Methods in Applied Mechanics and Engineering, 403:115718, 2023.

Pantelis R. Vlachas, Georgios Arampatzis, Caroline Uhler, and Petros Koumoutsakos. Multiscale simulations of complex systems by learning their effective dynamics. Nature Machine Intelligence, 4(4):359–366, April 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00464-w.

Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sébastien Ourselin, and Tom Vercauteren. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. Neurocomputing, 338:34–45, 2019.

Rui Wang, Robin Walters, and Rose Yu. Approximately equivariant networks for imperfectly symmetric dynamics. In International Conference on Machine Learning, pages 23078–23091. PMLR, 2022a.

Rui Wang, Robin Walters, and Rose Yu. Data augmentation vs. equivariant networks: A theory of generalization on dynamics forecasting. arXiv preprint arXiv:2206.09450, 2022b.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. Science Advances, 7(40), September 2021. doi: 10.1126/sciadv.abi8605.

Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks. arXiv preprint arXiv:2203.07404, March 2022c. doi: 10.48550/arXiv.2203.07404.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Improved Architectures and Training Algorithms for Deep Operator Networks. Journal of Scientific Computing, 92(2), June 2022d. ISSN 1573-7691. doi: 10.1007/s10915-022-01881-0.

Yuan Yin, Vincent Le Guen, Jérémie Dona, Emmanuel de Bézenac, Ibrahim Ayed, Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting\ast. Journal of Statistical Mechanics: Theory and Experiment, 2021(12), December 2021. ISSN 1742-5468. doi: 10.1088/1742-5468/ac3ae5.

Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. Journal of Computational Physics, 394:56–81, 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2019.05.024.

Robert Zwanzig. Nonlinear generalized Langevin equations. Journal of Statistical Physics, 9(3):215–220, November 1973. ISSN 1572-9613. doi: 10.1007/BF01008729.

A. Paper A

Reprinted from Kaltenbach and Koutsourelakis (2020a) according to the author rights in Elsevier's journals [1].

[1] https://www.elsevier.com/about/policies/copyright

Journal of Computational Physics 419 (2020) 109673

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Incorporating physical constraints in a deep probabilistic machine learning framework for coarse-graining dynamical systems

Sebastian Kaltenbach, Phaedon-Stelios Koutsourelakis *

Professorship of Continuum Mechanics, Technical University of Munich, Germany

ARTICLE INFO

Article history:
Received 6 January 2020
Received in revised form 9 May 2020
Accepted 16 June 2020
Available online 29 June 2020

Keywords:
Bayesian machine learning
Virtual observables
Multiscale modeling
Reduced order modeling
Coarse graining

ABSTRACT

Data-based discovery of effective, coarse-grained (CG) models of high-dimensional dynamical systems presents a unique challenge in computational physics and particularly in the context of multiscale problems. The present paper offers a data-based, probabilistic perspective that enables the quantification of predictive uncertainties. One of the outstanding problems has been the introduction of physical constraints in the probabilistic machine learning objectives. The primary utility of such constraints stems from the undisputed physical laws such as conservation of mass, energy etc. that they represent. Furthermore and apart from leading to physically realistic predictions, they can significantly reduce the requisite amount of training data which for high-dimensional, multiscale systems are expensive to obtain (Small Data regime). We formulate the coarse-graining process by employing a probabilistic state-space model and account for the aforementioned equality constraints as virtual observables in the associated densities. We demonstrate how deep neural nets in combination with probabilistic inference tools can be employed to identify the coarse-grained variables and their evolution model without ever needing to define a fine-to-coarse (restriction) projection and without needing time-derivatives of state variables.

We advocate a sparse Bayesian learning perspective which avoids overfitting and reveals the most salient features in the CG evolution law. The formulation adopted enables the quantification of a crucial, and often neglected, component in the CG process, i.e. the predictive uncertainty due to information loss. Furthermore, it is capable of reconstructing the evolution of the full, fine-scale system and therefore the observables of interest need not be selected a priori. We demonstrate the efficacy of the proposed framework by applying it to systems of interacting particles and a series of images of a nonlinear pendulum. In both cases we identify the underlying coarse dynamics and can generate extrapolative predictions including the forming and propagation of a shock for the particle systems and a stable trajectory in the phase space for the pendulum.

* Corresponding author.
E-mail addresses: sebastian.kaltenbach@tum.de (S. Kaltenbach), p.s.koutsourelakis@tum.de (P.-S. Koutsourelakis).

https://doi.org/10.1016/j.jcp.2020.109673
0021-9991/

1. Introduction

High-dimensional, nonlinear dynamical systems are ubiquitous in applied physics and engineering. The computational resources needed for their solution can grow exponentially with the dimension of the state-space as well as with the smallest time-scale that needs to be resolved and which determines the discretization time-step. Hence the ability to construct reduced, *coarse-grained* descriptions and models that are nevertheless predictive of various observables and at time-scales much larger than the inherent ones, is an important task [1].

One strategy for learning such coarse-grained (CG) models is based on data generated by simulations of the fine-grained (FG) system. This can yield an automated solution especially in cases where domain knowledge is limited or absent. The derivation of CG models from data is also particularly relevant in domains where FG models are not available, such as in social sciences or biophysics, but data abound [2,3]. Data-based methodologies have also been fueled by recent advances in statistical- [4] or machine-learning [5] which, in large part, have been enabled by large datasets (and the computational means to leverage them). We note nevertheless that coarse-graining tasks based on FG simulation data exhibit some fundamental differences [6]. Firstly, the acquisition of FG simulation data is by definition expensive and the reduction of the required FG simulations is one of the objectives of CG model development. Secondly, in physical applications, significant information about the underlying physical/mathematical structure of the problem, and of the CG model in particular, is available. This information might come in the form of constraints that reflect e.g. undisputed physical principles such as conservation laws (e.g. mass, momentum, energy). Injecting this prior information into the CG models in combination with FG data in an automated fashion represents a significant challenge [7], especially in the context of *probabilistic* models [8]. Such a capability would be instrumental not only in reducing the required amount of FG data, but more importantly, in enabling predictions under *extrapolative* settings as those arising e.g. when the initial conditions of the FG system are different from the ones in the training data.

In this paper, we propose a generative, probabilistic (Bayesian) machine learning framework [9] which employs FG simulation data augmented by *virtual observables* to account for constraints. The latter concept which we elucidate in the sequel, enables the incorporation of domain knowledge in probabilistic models and represents, in our opinion the most novel contribution of this paper. Furthermore and within the Bayesian framework advocated, it allows us to introduce appropriate priors that promote the discovery of *slow-varying* CG state-variables which is a highly-desirable feature for multiscale systems [10]. In contrast to most existing techniques which consider the problems of CG state variable discovery and CG model construction in two or more steps [11–14], we address both simultaneously [15]. The framework proposed consists of two building blocks: a probabilistic coarse-to-fine map [16] and an evolution law for the CG dynamics. The former can be endowed with great flexibility in discovering appropriate CG variables when combined with deep neural nets [17–19], which is especially challenging if the number of training data is small.[1] We demonstrate nevertheless the efficacy of such an approach when physical information is incorporated a-priori into the model. The CG variables identified are not restricted to indicator functions of sub-domains of the state-space as in other generative models [20,13,21] and which are difficult to learn when the simulation data is limited and has not sufficiently populated all important regions of the state-space.

The second component of the proposed framework pertains to the discovery of the CG evolution law which is learned by employing a large vocabulary of feature functions and sparsity-inducing priors. This leads to interpretable solutions [22], even in the *Small Data* regime that avoid overfitting and reveal salient characteristics of the CG system [23]. The premise of sparsity [24] has been employed in the past for the discovery of the CG dynamics as e.g. in the SINDy method [25–27]. This however requires the availability of time-derivatives of the CG variables and does not directly lead to a posterior on the model parameters that can reflect inferential uncertainties. Nonparametric models for the CG dynamics have also been proposed [28] but have been restricted to low dimensions. The learned CG dynamics are in general nonlinear in contrast to efforts based on transfer operators [29] and particularly the Koopman operator [30–32]. While the associated theory guarantees the existence of a linear operator, this is possible in the infinite dimensional space of observables, it does not specify how many should be used to obtain a good approximation, and more importantly, how one can predict future FG states given predictions on the evolution of those observables i.e. the reconstruction step.

The latter constitutes the main difference of the proposed model with non-generative ones based e.g. on information-theoretic concepts [33–35] or on the Mori-Zwanzig (MZ) formalism [36–38]. Apart from the difficulties in approximating the right-hand-side of the MZ-prescribed CG dynamics, and particularly the memory term [39,40], this can only guarantee correct predictions of the CG variables' evolution. If observables not depending on CG variables are of interest, then a reconstruction operator would need to be added. In contrast, in the proposed model this reconstruction operator is represented by the probabilistic coarse-to-fine map which is simultaneously learned from the data and can quantify predictive uncertainties associated with the information loss that unavoidably takes place in any CG process as well as due to the fact that finite (and preferably, small) data has been used for training.

The enabling computational technology for training the proposed model is based on probabilistic inference. In order to resolve the intractable posterior on latent variables and model parameters in our Bayesian framework, we make use

[1] In the dynamical systems investigated the size of the dataset depends on the length of the FG time-sequences as well as the number of such sequences employed for training.

of Stochastic Variational Inference [41] as MCMC is cumbersome in high dimensions. We operate on the discretized time domain [42] and demonstrate how amortized [43,44] and non-amortized approximations can be employed.

The remainder of the paper is structured as follows: In Section 2 we present the general methodological framework with special attention on the two building blocks of the state-space model proposed i.e. the transition law for the CG dynamics and the incorporation of virtual observables (section 2.2), as well as the emission law which provides the link between CG and FG description through a probabilistic *coarse-to-fine* map (section 2.3). Computational aspects related to inference and prediction are discussed in sections 2.4 and 2.5 respectively. Section 3 contains illustrative applications involving coarse-graining of high-dimensional systems of interacting particles (section 3.1) as well as learning the dynamics of a nonlinear pendulum (section 3.2) from a sequence of images. We conclude in section 4 which also contains a discussion on possible extensions.

2. Methodology

In general, we use the subscript f or lower-case letters to denote variables associated with the (high-dimensional) fine-grained(FG)/full-order model and the subscript c or upper-case letters for quantities of the (lower-dimensional) coarse-grained(CG)/reduced-order description. We also use a circumflex $\hat{}$ to denote observed/known variables. We begin with the presentation of the FG and the CG model and subsequently explain the essential ingredients of the proposed formulation.

2.1. The FG and CG models

We consider a, generally high-dimensional, FG system with state variables $\boldsymbol{x}$ of dimension d_f ($d_f >> 1$) such that $\boldsymbol{x} \in \mathcal{X}_f \subset \mathbf{R}^{d_f}$. The dynamics of the FG system are dictated by system of deterministic or stochastic ODEs i.e.,

$$\dot{\boldsymbol{x}}_t = \boldsymbol{f}(\boldsymbol{x}_t, t), \quad t > 0 \tag{1}$$

The initial condition $\boldsymbol{x}_0$ might be deterministic or drawn from a specified distribution. In the following we do not make explicit use of the FG dynamics but rely purely on FG data i.e. time sequences simulated from Equation (1) with a time-step, say δt. That is, our observables consists of n data sequences over $T+1$ FG time-steps δt i.e.,

$$\mathcal{D}_{T,n} = \{\hat{\boldsymbol{x}}_{0:T\delta t}^{(1:n)}\} \tag{2}$$

We denote the (unknown) CG state variables by $\boldsymbol{X}$ and assume $\boldsymbol{X} \in \mathcal{X}_c \subset \mathbf{R}^{d_c}$, where d_c is the dimension of the CG system. We presuppose Markovian dynamics[2] for the CG system of the form:

$$\dot{\boldsymbol{X}}_t = \boldsymbol{F}(\boldsymbol{X}_t, t) \tag{3}$$

which we discretize using a linear multistep method and a CG time step Δt:

$$\boldsymbol{R}_l(\boldsymbol{X}) = \sum_{k=0}^{K} \left(\alpha_k \boldsymbol{X}_{(l-k)\Delta t} + \Delta t \beta_k \boldsymbol{F}(\boldsymbol{X}_{(l-k)\Delta t})\right) = 0, \qquad l = K, K+1, \ldots \tag{4}$$

where α_k, β_k are the parameters of the discretization scheme and $\boldsymbol{R}_l$ the corresponding residual at time step l [45]. We note that depending on the values of the parameters K, α_k, β_k, several of the well-known, explicit/implicit, numerical time-integration schemes can be recovered. In this work, our goal is two-fold:

a) to identify the CG state-variables $\boldsymbol{X}$ and their relation with the FG description $\boldsymbol{x}$,
b) to identify the right-hand side of Equation (3),

in view of enabling predictions of the FG system over longer time horizons. Traditionally, the aforementioned tasks are *not* considered simultaneously. Usually the CG state variables are specified a priori using domain-knowledge (physical insight) or based on the observables of interest [34]. In other efforts, linear or non-linear dimensionality reduction procedures are first employed in order to identify such a lower-dimensional set of collective variables $\boldsymbol{X}$ (e.g. [46]). In both of these cases, $\boldsymbol{X}$ are defined using a *fine-to-coarse*, projection map e.g. $\boldsymbol{X} = \Pi(\boldsymbol{x})$ where $\Pi : \mathcal{X}_f \subset \mathbb{R}^{d_f} \to \mathcal{X}_c \subset \mathbb{R}^{d_c}$. Irrespective of whether this map is prescribed from the physics or learned from data, it is generally a many-to-one function that does not have an inverse i.e. if the CG states $\boldsymbol{X}$ are known one cannot readily reconstruct $\boldsymbol{x}$ [47].

We note that this has nothing to do with the quality of the CG evolution law (problem b) above). Even if the Mori-Zwanzig (MZ) formalism were employed, which in principle provides an exact, closed system of evolution equations for any observable of the FG states and therefore for $\boldsymbol{X} = \Pi(\boldsymbol{x})$, even if all the terms in the right-hand side were available, one would simply be able to predict the future evolution of $\boldsymbol{X}$ but not $\boldsymbol{x}$. This might be sufficient for a lot of problems

[2] As discussed in section 3, this assumption can be relaxed.

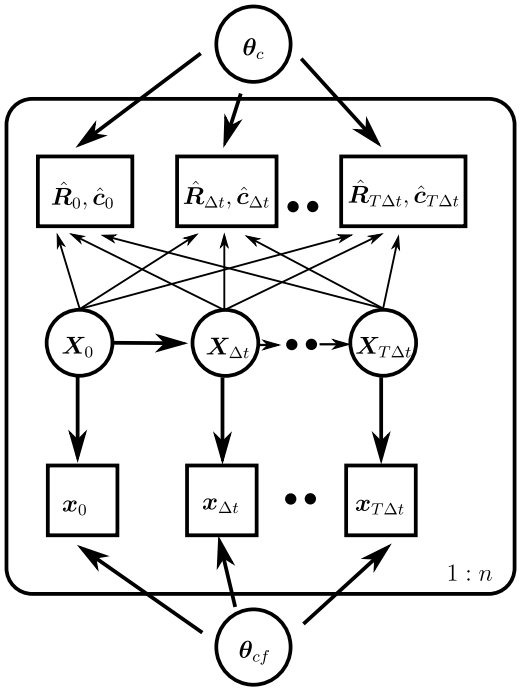


Fig. 1. Proposed probabilistic graphical model. The CG variables $\boldsymbol{X}$ are latent and are inferred together with the parameters $\boldsymbol{\theta}_c$ and $\boldsymbol{\theta}_{cf}$. Apart from the FG states $\boldsymbol{x}$, the observables are augmented by *virtual observables* $\hat{\boldsymbol{R}}, \hat{\boldsymbol{c}}$ (see section 2.2). These virtual observables can depend on all CG variables but more often this dependence is restricted to only a few of them.

of practical interest where the CG variables (or observables thereof) are of sole interest. Our goal however is a bit more ambitious, i.e. we seek to find a $\boldsymbol{X}$ that would allow us to reconstruct as accurately as possible the whole FG vector $\boldsymbol{x}$ into the future. As with any coarse-graining process, we recognize that this would unavoidably imply some information loss which in turn will give rise to predictive uncertainty [48]. In this work, we advocate a probabilistic framework that quantifies this uncertainty.

With regards to problem b) above, we note that its solution hinges upon the CG variables $\boldsymbol{X}$ employed (problem a)). Irrespective of the breadth of the model forms considered (i.e. functions $\boldsymbol{F}$ in Equation (3)), the evolution of some $\boldsymbol{X}$ might fall outside this realm. For example, it is known from MZ theory that memory terms can become significant for certain observables. It is well-known that such memory terms can be substituted or approximated by additional variables [49] which would in turn imply an augmented CG description $\boldsymbol{X}$ in Equation (3) that contains these auxiliary internal state variables [50].

We address problems a) and b) in the coarse-graining process *simultaneously* by employing a probabilistic state-space model. This consists of two densities i.e.

- the transition law which dictates the evolution of the CG variables $\boldsymbol{X}$ (section 2.2). Special attention is paid to the definition of *virtual observables* with which the CG states and their dynamics can be injected with physical information.
- the emission law which provides the link between CG and FG description through a probabilistic *coarse-to-fine* map (section 2.3, [15]).

We emphasize that in our formulation, the CG state-variables $\boldsymbol{X}$ are implicitly defined as latent generators of the FG description $\boldsymbol{x}$. As discussed in detail in the sequel, this enables a straightforward, *probabilistic* reconstruction of $\boldsymbol{x}$ when $\boldsymbol{X}$ is known. The inverse map (analogous to Π above) arises naturally through probabilistic inference as explained in section 2.4. An overview of the essential elements of the proposed model can be seen in the probabilistic graphical model of Fig. 1.

2.2. Transition law: CG dynamics and virtual observables

Typical state-space models [51–53,43] postulate Markovian, stochastic dynamics for the hidden variables $\boldsymbol{X}$, in the form of a diffusion process, which are subsequently discretized *explicitly* using e.g. a Euler-Maruyama scheme with time step Δt. This gives rise to a, generally Gaussian, conditional density $p(\boldsymbol{X}_{(l+1)\Delta t}|\boldsymbol{X}_{l\Delta t})$ which can be stacked over multiple time-instants in order to formulate a generalized prior on the CG-space.

When the CG state-variables $\boldsymbol{X}$ are given (in part or in whole) physical meaning (e.g. as thermodynamic state variables), then some of the equations for their evolution are prescribed by associated physical principles e.g. conservation of mass, momentum, energy. These can be reflected in the residuals $\boldsymbol{R}_l$ of the governing equations as in Equation (4) or alternatively as equality constraints of the form:

$$\boldsymbol{c}_l(\boldsymbol{X}_{l\Delta t}) = \boldsymbol{0}, \quad l = 0, 1, \ldots \tag{5}$$

which must hold at each time-step. The function $\boldsymbol{c}_l : \mathcal{X}_c \subset \mathbb{R}^{d_c} \rightarrow \mathbb{R}^{M_c}$ enforces these known constraints at each time-step (see specific examples in section 3) and the only requirement we will impose is that of differentiability of $\boldsymbol{c}_l$ (see section 2.4). In order to account for the aforementioned constraints in the transition law of the CG state variables, we employ the novel (to the best of our knowledge) concept of *virtual observables*. In particular for each of the residuals $\boldsymbol{R}_l$ in Equation (4), we define a new variable/vector $\hat{\boldsymbol{R}}_l$ which relates to $\boldsymbol{R}_l$ as follows:

$$\hat{\boldsymbol{R}}_l = \boldsymbol{R}_l(\boldsymbol{X}) + \sigma_R \boldsymbol{\epsilon}_R, \qquad \boldsymbol{\epsilon}_R \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{6}$$

We further assume that $\hat{\boldsymbol{R}}_l$ have been *virtually observed* and $\hat{\boldsymbol{R}}_l = 0$ leading to an augmented version of the data in Equation (2), by a set of virtual observations and therefore virtual likelihoods of the type:

$$p(\hat{\boldsymbol{R}}_l = \boldsymbol{0} \mid \boldsymbol{X}, \sigma_R) = \mathcal{N}(\boldsymbol{0} \mid \boldsymbol{R}_l(\boldsymbol{X}), \sigma_R^2 \boldsymbol{I}) \tag{7}$$

The "noise" parameter σ_R determines the intensity of the enforcement of the virtual observations and is analogous to the tolerance parameter with which residuals are enforced in a deterministic solution of the dynamics. Similarly, for constraints of the form of Equation (5), additional variables and virtual observables of the type:

$$\boldsymbol{0} = \hat{\boldsymbol{c}}_l = \boldsymbol{c}_l(\boldsymbol{X}_{l\Delta t}) + \sigma_c \boldsymbol{\epsilon}_c, \quad \boldsymbol{\epsilon}_c \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{8}$$

can be defined which would lead to an augmented (virtual) likelihood with terms of the type:

$$p\left(\hat{\boldsymbol{c}}_l = \boldsymbol{0} \,|\boldsymbol{X}_{l\Delta t}, \sigma_c\right) = \mathcal{N}\left(\boldsymbol{0} \mid \boldsymbol{c}_l(\boldsymbol{X}_{l\Delta t}), \sigma_c^2 \boldsymbol{I}\right) \tag{9}$$

where the role of σ_c^2 is analogous to σ_R^2 above.

Since the goal is to identify the right-hand side of the evolution laws in Equation (3), we denote by $\boldsymbol{\theta}_c$ the parameters appearing in $\boldsymbol{F}$ i.e. $\boldsymbol{F}(\boldsymbol{X}_t, t;\ \boldsymbol{\theta}_c)$. Accordingly, the virtual observations in Equation (6) or Equation (8) would depend on $\boldsymbol{\theta}_c$. We defer until section 3 a detailed discussion on the form, the parametrization as well as the prior specifications in the Bayesian setting adopted. The latter plays an important role as with sparsity-inducing priors we can avoid overfitting and obtain a parsimonious and physically-interpretable solution for $\boldsymbol{F}$. We finally remark that physical information taking the form of equalities can also be available for the FG states $\boldsymbol{x}$. While this can be incorporated using appropriate virtual observables as above, the inference framework would exhibit significant differences (in brief, FG states would need to be inferred as well) and in order to avoid confusion we do not discuss such cases here.

2.3. Emission law: coarse-to-fine map

We make use of a probabilistic *generative* model in the definition of the CG state-variables through a *coarse-to-fine* map [15] as opposed to traditional, many-to-one maps from the FG description to the CG one. We denote the associated (conditional) density by:

$$p_{cf}(\boldsymbol{x}_t |\ \boldsymbol{X}_t;\ \boldsymbol{\theta}_{cf}) \tag{10}$$

where $\boldsymbol{\theta}_{cf}$ denote the (unknown) parameters that will be learned from the data. The form of p_{cf} can be adapted to the particulars of the problem and can be endowed with various levels of domain knowledge. In section 3, we provide various examples, from particle-systems where p_{cf} is fully determined by the physics, to a more abstract case where deep neural networks are employed in order to learn the full p_{cf}. We note finally that a (probabilistic) fine-to-coarse map can still be learned in the current setting, and would correspond to the *posterior* of $\boldsymbol{X}_t$ given $\boldsymbol{x}_t$. We discuss this as well as all aspects pertaining to inference and learning in the next section.

2.4. Inference and learning

We start this section by summarizing the main elements of the model presented (i.e. data, latent variables and parameters - see also Table 1) and subsequently describe a fully Bayesian inference scheme based on Stochastic Variational Inference (SVI, [41]) tools.

We adopt an enlarged definition of *data* which we cumulatively denote by $\mathcal{D}$ and which encompasses:

- FG simulation data as in Equation (2) consisting of n sequences of the FG state-variables. As the likelihood model implied by the p_{cf} in Equation (10) involves only the observables at each *coarse* time-step we denote those by $\{\hat{\boldsymbol{x}}_{0:T\Delta t}^{(1:n)}\}$. We assume that the number of observations in each sequence is the same although this is not necessary. In fact, the length of each time-sequence and the number of time-sequences needed could be the subject of an active learning scheme. This would be particularly important in cases where very expensive, high-dimensional FG simulators are employed. The

Table 1
Data, latent variables and model parameters.

Observables $\mathcal{D}$	$\hat{\boldsymbol{x}}^{(1:n)}_{0:T\Delta t}$	FG simulation data
	$\hat{\boldsymbol{R}}^{(1:n)}_{0:T}$	Virtual observables corresponding to CG model residuals
	$\hat{\boldsymbol{c}}^{(1:n)}_{0:T}$	Virtual observables corresponding to CG constraints
Latent variables	$\boldsymbol{X}^{(1:n)}_{0:T\Delta t}$	CG state variable
Model parameters $\boldsymbol{\theta}$	$\boldsymbol{\theta}_{cf}$	parameters in the coarse-to-fine mapping
	$\boldsymbol{\theta}_{c}$	parameters in the CG evolution law

generative, proposed formulation can account for any type of (in)direct or (in)complete/partial, experimental or computational observations relating to FG states which we omit here for simplicity of the presentation. We nevertheless illustrate this capability of the model in the example of section 3.2.

- Virtual observables relating to the CG states $\boldsymbol{X}$ at each time-step l consisting of residuals $\hat{\boldsymbol{R}}_l^{(1:n)}$ as in Equation (6) and/or constraints $\hat{\boldsymbol{c}}_l^{(1:n)}$ as in Equation (8) (the superscript pertains to the time sequence $i = 1, \dots, n$). Assuming they pertain to all time-steps, we denote them by $\left\{\hat{\boldsymbol{R}}^{(1:n)}_{0:T}, \hat{\boldsymbol{c}}^{(1:n)}_{0:T}\right\}$.

The latent (unobserved) variables of the model are represented by the CG state-variables $\left\{\boldsymbol{X}^{(1:n)}_{0:T\Delta t}\right\}$ which relate to the FG data through the p_{cf} (in Equation (10)) and to the virtual observables through Equation (7) or Equation (9).

Finally, the (unknown) parameters of the model which we denote cumulatively by $\boldsymbol{\theta}$ consist of[3]:

- $\boldsymbol{\theta}_c$ which parametrize the right-hand-side of the CG evolution law (see end of section 2.2),
- $\boldsymbol{\theta}_{cf}$ which parametrize the probabilistic coarse-to-fine map (Equation (10)),
- σ_R, σ_c involved in the enforcement of virtual observables in Equation (6) and Equation (8) respectively, and,
- *hyperparameters* associated with the priors employed on the latent variables or the previous parameters.

Following a fully-Bayesian formulation, we can express the posterior of the unknowns (i.e. latent variables and parameters) as follows:

$$p(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{X}^{(1:n)}_{0:T\Delta t}, \boldsymbol{\theta})\ p(\boldsymbol{X}^{(1:n)}_{0:T\Delta t}, \boldsymbol{\theta})}{p(\mathcal{D})} \tag{11}$$

where $p(\boldsymbol{X}^{(1:n)}_{0:T\Delta t}, \boldsymbol{\theta})$ denotes the prior on the latent variables and parameters.

We discuss first the likelihood term $p(\mathcal{D}|\boldsymbol{X}^{(1:n)}_{0:T\Delta t}, \boldsymbol{\theta})$ which can be decomposed into the product of three (conditionally) independent terms, one for each data-type, i.e.:

$$p(\mathcal{D} \mid \boldsymbol{X}^{(1:n)}_{0:T\Delta t}, \boldsymbol{\theta}) = p(\hat{\boldsymbol{x}}^{(1:n)}_{0:T\Delta t} \mid \boldsymbol{X}^{(1:n)}_{0:T\Delta t}, \boldsymbol{\theta})\ p(\hat{\boldsymbol{R}}^{(1:n)}_{0:T} \mid \boldsymbol{X}^{(1:n)}_{0:T\Delta t}, \boldsymbol{\theta})\ p(\hat{\boldsymbol{c}}^{(1:n)}_{0:T} \mid \boldsymbol{X}^{(1:n)}_{0:T\Delta t}, \boldsymbol{\theta}) \tag{12}$$

We further note that (from Equation (10)):

$$p(\hat{\boldsymbol{x}}^{(1:n)}_{0:T\Delta t} \mid \boldsymbol{X}^{(1:n)}_{0:T\Delta t}, \boldsymbol{\theta}) = \prod_{i=1}^{n}\prod_{l=0}^{T} p_{cf}(\boldsymbol{x}^{(i)}_{l\ \Delta t} \mid \boldsymbol{X}^{(i)}_{l\ \Delta t}, \boldsymbol{\theta}_{cf}) \tag{13}$$

and (from Equation (7)):

$$\begin{aligned} p(\hat{\boldsymbol{R}}^{(1:n)}_{0:T}|\boldsymbol{X}^{(1:n)}_{0:T\Delta t}, \boldsymbol{\theta}) &= \textstyle\prod_{i=1}^{n}\prod_{l=0}^{T} \mathcal{N}\left(\boldsymbol{0}|\boldsymbol{R}_l(\boldsymbol{X}^{(i)}), \sigma_R^2\boldsymbol{I}\right) \\ &\propto \textstyle\prod_{i=1}^{n}\prod_{l=0}^{T} \frac{1}{\sigma_R^{dim(\boldsymbol{R})}} \exp\left\{-\frac{1}{2\sigma_R^2}\left|\boldsymbol{R}_l(\boldsymbol{X}^{(i)})\right|^2\right\} \end{aligned} \tag{14}$$

and (from Equation (9)):

$$\begin{aligned} p(\hat{\boldsymbol{c}}^{(1:n)}_{0:T}|\boldsymbol{X}^{(1:n)}_{0:T\Delta t}, \boldsymbol{\theta}) &= \textstyle\prod_{i=1}^{n}\prod_{l=0}^{T} \mathcal{N}(\boldsymbol{0}|\boldsymbol{c}_l(\boldsymbol{X}^{(i)}_{l\ \Delta t}), \sigma_c^2\boldsymbol{I}) \\ &\propto \textstyle\prod_{i=1}^{n}\prod_{l=0}^{T} \frac{1}{\sigma_c^{dim(\boldsymbol{c})}} \exp\left\{-\frac{1}{2\sigma_c^2}\left|\boldsymbol{c}_l(\boldsymbol{X}^{(i)}_{l\ \Delta t})\right|^2\right\} \end{aligned} \tag{15}$$

[3] If any of the parameters in this list are prescribed, then they are omitted from $\boldsymbol{\theta}$.

While the complexity of the expressions involved imply a non-analytic solution for the posterior, we emphasize that the terms above encode actual and virtual observables (constraints) and they are differentiable, a property that is crucial for carrying out Variational Inference.

Before presenting the inference procedure, we mention an interesting possibility for encoding prior information for the latent CG states $\boldsymbol{X}^{(1:n)}_{0:T\Delta t}$ through the prior term $p(\boldsymbol{X}^{(1:n)}_{0:T\Delta t})$. A desirable property of the CG state-variables is that of *slowness* i.e. that they should capture features of the system that evolve over (much) larger time-scales [10]. The discovery of such features has been the goal of several statistical analysis procedures (e.g. Slow Feature Analysis [54]) as well as in physics/chemistry literature (see a recent review in [29]). In this work we promote the discovery of such slow features by appropriate prior selection, and in particular by penalizing the jumps between two successive time-instants, i.e.:

$$\begin{aligned} p(\boldsymbol{X}^{(1:n)}_{0:T\Delta t}) &= \textstyle\prod_{i=1}^{n} p_{c,0}(\boldsymbol{X}^{(i)}_{0}) \prod_{l=0}^{T-1} p(\boldsymbol{X}^{(i)}_{(l+1)\ \Delta t} | \boldsymbol{X}^{(i)}_{l\ \Delta t}, \sigma_X^2 \boldsymbol{I}) \\ &= \textstyle\prod_{i=1}^{n} p_{c,0}(\boldsymbol{X}^{(i)}_{0}) \prod_{l=0}^{T-1} \mathcal{N}(\boldsymbol{X}^{(i)}_{(l+1)\ \Delta t} | \boldsymbol{X}^{(i)}_{l\ \Delta t}, \sigma_X^2 \boldsymbol{I}) \\ &\propto \textstyle\prod_{i=1}^{n} p_{c,0}(\boldsymbol{X}^{(i)}_{0}) \prod_{l=0}^{T-1} \frac{1}{\sigma_X^{dc}} \exp\left\{ -\frac{1}{\sigma_X^2} \left| \boldsymbol{X}^{(i)}_{(l+1)\ \Delta t} - \boldsymbol{X}^{(i)}_{l\ \Delta t} \right|^2 \right\} \end{aligned} \tag{16}$$

where $p_{c,0}$ is a prior density for the initial CG state. We observe that the strength of the penalty is inversely proportional to the hyperparameter σ_X^2 and in the limit $\sigma_X^2 \to 0$ it implies a constant time history of $\boldsymbol{X}_t$. As the appropriate value for σ_X^2 depends on the problem, we include this in the parameter vector $\boldsymbol{\theta}$ that is inferred/learned from the data.

Given the intractability of the actual posterior, we advocate in this work Variational Inference. This operates on a parameterized family of densities, say $q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta})$ and attempts to find the one (i.e. the value of $\boldsymbol{\phi}$) that most closely approximates the posterior by minimizing their Kullback-Leibler divergence. It can be readily shown [55], that the optimal $q_{\boldsymbol{\phi}}$, maximizes the Evidence Lower Bound (ELBO) $\mathcal{F}(q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta}))$ below:

$$\begin{aligned} \log p(\mathcal{D}) &= \log \textstyle\int p(\mathcal{D},\ \boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta})\, d\boldsymbol{X}^{(1:n)}_{0:T\Delta t}\, d\boldsymbol{\theta} \\ &= \log \textstyle\int \frac{p(\mathcal{D}|\ \boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta}) p(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta})} q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta})\, d\boldsymbol{X}^{(1:n)}_{0:T\Delta t}\, d\boldsymbol{\theta} \\ &\geq \textstyle\int \log \frac{p(\mathcal{D}|\ \boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta}) p(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta})} q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta})\, d\boldsymbol{X}^{(1:n)}_{0:T\Delta t}\, d\boldsymbol{\theta} \\ &= \mathcal{F}(q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta})) \end{aligned} \tag{17}$$

In the examples analyzed we decompose the approximate posterior as:

$$\begin{aligned} q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta}) &= q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(1:n)}_{0:T\Delta t})\, q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \\ &= \left[\textstyle\prod_{i=0}^{n} q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(i)}_{0:T\Delta t}) \right]\ q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \end{aligned} \tag{18}$$

where the first line is the so-called mean-field approximation and the second is a direct consequence of the (conditional) independence of the time sequences in the likelihood. We note that evaluations of the ELBO $\mathcal{F}$ involve expectations with respect to $q_{\boldsymbol{\phi}}$ i.e.:

$$\mathcal{F}\left(q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta})\right) = \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p(\mathcal{D}|\ \boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta})\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \frac{p(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(1:n)}_{0:T\Delta t},\ \boldsymbol{\theta})}\right] \tag{19}$$

and in order to maximize it (with respect to $\boldsymbol{\phi}$), gradients of those are needed. Given the intractability of these expectations and their derivatives, we make use of Monte Carlo estimates in combination with stochastic gradient ascent for the $\boldsymbol{\phi}$-updates. In order to reduce the Monte Carlo error in these estimates, we make use of the reparametrization trick [56], for which the differentiability of the residuals/constraints is necessary. We specify the particulars of the algorithm more precisely in the numerical illustration section (see e.g. Algorithm 3 or 4).

We note that maximum likelihood or maximum-a-posteriori (MAP) point estimates for any of the parameters involved can be obtained as a special case of the aforementioned scheme by employing a $q_{\boldsymbol{\phi}}$ that is equal to a Dirac-delta function. Furthermore, amortized versions of the approximate posterior $q_{\boldsymbol{\phi}}$ i.e. forms that explicitly account on the dependence on the data values, can be employed in part or in whole. These have the capability of being able to transfer information across data points and are necessary in the realm of Big Data. We note though that we operate in the *Small Data* regime, i.e. the number of time sequences n (and time-steps T) is not particularly large. Hybrid versions between amortized and non-amortized posteriors could also be employed [57].

We note finally that while the ELBO $\mathcal{F}$ is used purely as the objective function for the determination of the approximate posterior, its role can be quite significant in model validation and refinement. In particular since $\mathcal{F}$ approximates the model evidence (denominator of Equation (11)), once evaluated, it can be used to comparatively assess different models. These

could have different CG states $\boldsymbol{X}$ (in type and/or number) or different parametrizations $\boldsymbol{\theta}$. In this regard, the ELBO $\mathcal{F}$ could serve as the primary driver for the adaptive refinement of the CG model [58] in order to better explain the observables and lead to superior predictions.

2.5. Prediction

An essential feature of the proposed modeling framework is the ability to produce *probabilistic* predictive estimates. These encompass the information-loss due to the coarse-graining process as well as the epistemic uncertainty arising from finite (and small) datasets. We distinguish between two settings:

a) the *"interpolative"* i.e. predictions into the future of a sequence i observed up to time-step T i.e. $\hat{\boldsymbol{x}}^{(i)}_{0:T\Delta t}$ which was used in the training phase - see section 3, or
b) the *"extrapolative"* i.e. predictions for a completely new initial condition $\hat{\boldsymbol{x}}_0$ - see section 3.

We note that any predictions should account for the domain knowledge incorporated in the training through the residuals $\boldsymbol{R}_l$ or constraints $\boldsymbol{c}_l$. Formally that is, one should enlarge the posterior density defined in Equation (11), in order to account for the residuals and/or constraints at future time-steps. This would in turn imply, that future (FG or CG) states should be inferred from such an augmented posterior i.e. prediction would imply an enlarged inference process. In the examples presented we adopt a simpler procedure that retains the essential features (i.e. probabilistic nature) but is more computationally expedient. In particular, for case a) above and if $q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(i)}_{T\Delta t})$ is the (marginal) posterior of the last, hidden CG state and $q(\boldsymbol{\theta})$ the posterior of the model parameters, then we (see also Algorithm 1):

- sample from $q(\boldsymbol{X}^{(i)}_{T\Delta t}), q(\boldsymbol{\theta})$
- for each sample, we propagate the CG dynamics of Equation (3) (e.g. by solving the corresponding residual Equations (4)) in order to obtain $\boldsymbol{X}^{(i)}_{(T+1)\Delta t}, \boldsymbol{X}^{(i)}_{(T+2)\Delta t}, \ldots$, and,
- we sample $\boldsymbol{x}^{(i)}_{(T+1)\Delta t}$ from $p_{cf}(\boldsymbol{x}^{(i)}_{(T+1)\Delta t}|\boldsymbol{X}^{(i)}_{(T+1)\Delta t}, \boldsymbol{\theta}_{cf})$, $\boldsymbol{x}^{(i)}_{(T+2)\Delta t}$ from $p_{cf}(\boldsymbol{x}^{(i)}_{(T+2)\Delta t}|\boldsymbol{X}^{(i)}_{(T+2)\Delta t}, \boldsymbol{\theta}_{cf})$ etc.

We note that this procedure does not necessarily ensure enforcement of the constraints by future CG states. Nevertheless it gives rise to samples of the full FG state evolution from which any observable of interest as well as the predictive uncertainty can be computed.

Algorithm 1: Prediction - algorithm for *interpolative* setting.

Result: Sample of $\boldsymbol{x}^{(i)}_{(T+P)\Delta t}$
Data: $q_{\boldsymbol{\phi}}(\boldsymbol{X}_{T\Delta t}), q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$
Sample from $q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(i)}_{T\Delta t})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$;
while *Time-step* $(T+P)\Delta t$ *of interest not reached* **do**
  Apply the CG evolution law as described in Equation (4);
end
Sample from $p_{cf}(\boldsymbol{x}_{(T+P)\Delta t} \mid \boldsymbol{X}_{(T+P)\Delta t}, \boldsymbol{\theta})$

For the *extrapolative* setting above, i.e. for a new FG initial condition $\hat{\boldsymbol{x}}_0$, the evolution equations of the CG states as well as the emission density p_{cf} can be employed as long as the initial state $\boldsymbol{X}_0$ is specified or better yet inferred. For that purpose, the posterior $p(\boldsymbol{X}_0|\hat{\boldsymbol{x}}_0)$ of $\boldsymbol{X}_0$ needs to be determined which according to Bayes rule will be proportional to:

$$p(\boldsymbol{X}_0 \mid \hat{\boldsymbol{x}}_0) \propto p_{cf}(\hat{\boldsymbol{x}}_0 \mid \boldsymbol{X}_0, \boldsymbol{\theta}_{cf})\; p_{c,0}(\boldsymbol{X}_0) \tag{20}$$

where $p_{c,0}(\boldsymbol{X}_0)$ is the initial state's prior (see also Equation (16)). For each sample of $\boldsymbol{\theta}_{cf}$ from the (approximate) posterior $q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_{cf})$, samples of $\boldsymbol{X}_0$ must be drawn from $p(\boldsymbol{X}_0|\hat{\boldsymbol{x}}_0)$ and subsequently propagated as in the 3 steps above in order to obtain predictive samples of the full FG state vector (see Algorithm 2).

2.6. Computational considerations

We note that in multiscale dynamical systems of physical interest, the computational cost stems primarily from the simulation of the FG system due to its generally very high-dimensional state-vector $\boldsymbol{x}$ and very small time-step δt. Hence, one of the main objectives of this work is to enable the learning of the CG dynamics with the fewest possible and shortest possible FG time-sequences.

We note that once such FG simulation (or experimental) data have been obtained, neither the training phase of the CG model (section 2.4) nor the prediction phase (section 2.5) require any additional FG simulations. The cost of training depends on the dimension of the CG states $\boldsymbol{X}$ as well as the number of parameters $\boldsymbol{\theta}_c$ (for the CG dynamics), $\boldsymbol{\theta}_{cf}$ (for the coarse-to-fine map) and $\boldsymbol{\phi}$ (for the approximate posterior).

Algorithm 2: Prediction - algorithm for *extrapolative* setting.

Result: Sample of $\boldsymbol{x}_{P\Delta t}$
Data: $p_{\boldsymbol{\phi}}(\hat{\boldsymbol{x}}_0), q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$
Apply Bayesian Inference as described in Equation (20) to infer $p(\boldsymbol{X}_0|\hat{\boldsymbol{x}}_0)$;
Sample from $p(\boldsymbol{X}_0|\hat{\boldsymbol{x}}_0)$ and $q(\boldsymbol{\theta})$;
while *Time-step* $P\Delta t$ *of interest not reached* **do**
  Apply the CG evolution law as described in Equation (4);
end
Sample from $p_{cf}(\boldsymbol{x}_{P\Delta t}|\boldsymbol{X}_{P\Delta t}, \boldsymbol{\theta})$

We emphasize that this is a one-time, offline cost i.e. once the CG model has been trained, it can be used to produce probabilistic predictive estimates of the whole FG state-vector into the future without any further recourse to the FG model. One needs only to simulate in such case the CG dynamics which due to the lower-dimensional state-vector $\boldsymbol{X}$ and the much larger CG time-step Δt are much less cumbersome than the FG system.

Finally, if more FG data (e.g. longer or new sequences) become available at a later stage, the SVI algorithm can be re-initialized from the previous values and incorporate the new likelihood terms. If a modest amount of data is introduced, one would expect small (or even no changes for faraway states) changes and therefore rapid convergence. Naturally the introduction of observables at new time instants would introduce additional latent variables for the corresponding CG states.

3. Numerical illustrations

We demonstrate the capabilities of the proposed framework in discovering predictive, coarse-grained evolution laws as well as effective coarse-grained descriptions, on three examples. Two of those involve very high-dimensional systems of stochastically interacting particles (section 3.1, [15]) and the third, a nonlinear pendulum, the dynamics of which we attempt to identify simply from sequences of images (section 3.2, [27]). In the sequel, we specify the elements of the proposed model that were presented generically in the previous sections and concretize parametrizations and their meaning. The goals of the numerical illustrations are:

- to assess the predictive performance of the model under "interpolative" and "extrapolative" conditions (see section 2.5). By "interpolative" we mean the ability to predict the evolution of an FG states-sequence when data from this sequence has been used for training. By "extrapolative", we mean the ability to predict the full FG state evolution from new initial conditions that were *not* used in training.
- to examine the effect of the number n and length T of the data sequences and assess the model's ability to learn the correct structure with small n, T and partial observations.
- to examine the enforcement of the residuals/constraints (e.g. conservation of mass) in the inferred and predicted states.
- to examine the ability of the model to identify *sparse*, interpretable solutions for the CG dynamics.
- to assess the magnitude and time evolution of the predictive uncertainty estimates.
- to assess the ability of the model to learn effective CG state variables and accurate coarse-to-fine maps.

Some of the simulation results as well as the corresponding code will be made available at the following github repository[4] upon publication.

3.1. Particle systems

3.1.1. FG model

The FG model consists of d_f identical particles which can move in the bounded one-dimensional domain $[-1,\ 1]$ (under periodic boundary conditions). The FG variables $\boldsymbol{x}_t$ consist therefore of the coordinates of the particles at each time instant t and the dimension of the system d_f is equal to the number of particles. We consider two types of stochastic dynamics that correspond to an advection-diffusion-type (section 3.1.5) and an inviscid-Burgers-type behavior (section 3.1.6). The particulars of the microscopic dynamics are described in the corresponding sections. In the following, we discuss common aspects of both problems that pertain to the CG description, the CG evolution law and the inference procedures.

3.1.2. CG variables and coarse-to-fine mapping

For the CG representation, we employ the normalized particle density $\rho(s,t),\ s \in [-1,\ 1]$ [59] which we discretize in d_c bins. The state vector $\boldsymbol{X}_t = \{X_{t,j}\}_{j=1}^{d_c}$ contains the particle density values in each of the bins j, i.e. $\sum_{j=1}^{d_c} X_{t,j} = 1$ and $X_{t,j} \geq 0\ \forall t, j$. We emphasize that CG and FG variables are of a different nature (i.e. proportion of particles in each bin vs. coordinates of particles) and, more importantly for practical purposes, of very different dimension.

[4] https://github.com/SebastianKaltenbach/PhysicalConstraints_ProbabilisticCG.git.

The nature of the CG variables $\boldsymbol{X}_t$ suggests a *multinomial* for the coarse-to-fine density p_{cf} (section 2.3) i.e.:

$$p_{cf}(\boldsymbol{x}_t|\boldsymbol{X}_t) = \frac{d_f!}{m_1(\boldsymbol{x}_t)!\, m_2(\boldsymbol{x}_t)! \ldots m_{d_c}(\boldsymbol{x}_t)!} \prod_{j=1}^{d_c} X_{t,j}^{m_j(\boldsymbol{x}_t)}, \tag{21}$$

where $m_j(\boldsymbol{x}_t)$ is the number of particles in bin j. The underlying assumption is that, given the CG state $\boldsymbol{X}_t$, the coordinates of the particles $\boldsymbol{x}_t$ are *conditionally* independent. This does *not* imply that they move independently nor that they cannot exhibit coherent behavior [15]. The practical consequence of Equation (21) is that no parameters need to be learned for p_{cf} (in contrast to section 3.2).

3.1.3. The CG evolution law and the virtual observables

With regards to the evolution law of the CG states (Equation (3)), we postulate a right-hand side $\boldsymbol{F}(\boldsymbol{X}_t;\boldsymbol{\theta}_c) = \left\{F_j(\boldsymbol{X}_t;\boldsymbol{\theta}_c)\right\}_{j=1}^{d_c}$ of the form:

$$\begin{aligned} F_j(\mathbf{X}_t,\boldsymbol{\theta}_c) &= \textstyle\sum_{m=1}^{M} \theta_{c,m}\, \psi_m^{(j)}(\mathbf{X}_t) \\ &= \underbrace{\sum_{h=-H}^{H} \theta_{c,h}^{(1)} X_{t,j+h}}_{\text{1st order}} + \underbrace{\sum_{h_1=-H}^{H} \sum_{h_2\geq h_1}^{H} \theta_{c,\,(h_1,h_2)}^{(2)} X_{t,j+h_1} X_{t,j+h_2}}_{\text{2nd order}} \end{aligned} \tag{22}$$

which consists of first- and second-order interactions over a window of size H with $\boldsymbol{\theta}_c^{(1)}$ and $\boldsymbol{\theta}_c^{(2)}$ denoting the vectors of the corresponding unknown coefficients. In this case, the total number of unknown coefficients $\boldsymbol{\theta}_c$, is $M = dim(\boldsymbol{\theta}_c) = (2H+1) + (H+1)(2H+1)$ and grows quadratically with the neighborhood-size H. Since each of the CG variables $X_{t,j}$ refers to the particle density at bin j (and at time t), the neighborhood size H corresponds to the number of bins to the left or to the right of bin j that affect its evolution in time The feature functions that we generically denote with $\psi_m^{(j)}$ in Equation (22) can also involve higher-order interactions or be of non-polynomial type. Non-Markovian models could be accommodated as well by accounting for memory terms. It is obviously impossible to know a priori which feature functions are relevant in the evolution of the CG states or what types of interactions are essential (e.g. first, second-order etc). At the same time, and especially in the Small Data regime considered, employing a large vocabulary of feature functions can lead to *overfitting*, lack of interpretability and poor predictions, particularly under "extrapolative" conditions. This highly-important *model selection* issue has been of concern in several coarse-graining studies [60]. We propose of automatically addressing this within the Bayesian framework advocated by employing appropriate sparsity-inducing priors for $\boldsymbol{\theta}_c$ [15]. In particular, we make use of the Automatic Relevance Determination (ARD, [61]) model according to which

$$p(\theta_{c,m} \mid \tau_m) = \mathcal{N}(\theta_{c,m} \mid 0, \tau_m^{-1}), \qquad m = 1, 2, \ldots, M = dim(\boldsymbol{\theta}_c). \tag{23}$$

The following hyperprior for the precision hyperparameters $\boldsymbol{\tau} = \{\tau_m\}_{m=1}^{M}$ was used:

$$p(\tau_k \mid \gamma_0, \delta_0) = Gamma(\tau_k \mid \gamma_0, \delta_0) \tag{24}$$

The hyperparameters γ_0 and δ_0 are set to very small values 10^{-9} in all ensuing studies [62]. As we demonstrate in the sequel, the hyperprior proposed can give rise to parsimonious solutions for the CG dynamics even in the Small Data setting considered.

A discretized version of the CG evolution law (Equation (3) and Equation (22)) with time step Δt is considered by employing a forward Euler scheme[5] which implies the following residual vector $\boldsymbol{R}_l$ at each time-step l (Equation (4)):

$$\boldsymbol{R}_l(\boldsymbol{X}) = \boldsymbol{X}_{(l+1)\Delta t,\, j} - \boldsymbol{X}_{l\Delta t,\, j} - \Delta t\ \boldsymbol{F}(\boldsymbol{X}_{l\Delta t,\, j}, \boldsymbol{\theta}_c), \quad \forall\, l \tag{25}$$

and the corresponding virtual observables $\hat{\boldsymbol{R}}_l$ (Equation (6)).

More importantly, the nature of the CG variables suggests a *conservation of mass* constraint that has to be fulfilled at each time step l. In view of the discussion of section 2.2, this suggests the scalar constraint function as in Equation (5):

$$c_l(\boldsymbol{X}_{l\Delta t}) = \sum_{j=1}^{d_c} X_{l\Delta t,\, j}\ - 1 = 0, \quad \forall\, l \tag{26}$$

and the corresponding virtual observables $\hat{c}_l$ (Equation (8)).

[5] This corresponds to a multistep method in Equation (4) with $K = 1$, $a_0 = 1, a_1 = -1, \beta_0 = 0$ and $\beta_1 = -1$.

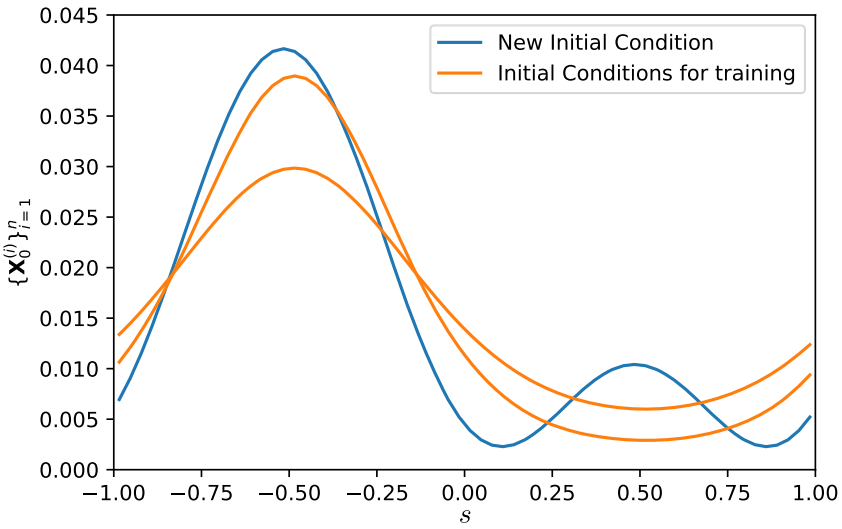


Fig. 2. Sample initial conditions $\left\{\boldsymbol{X}_0^{(i)}\right\}_{i=1}^n$ for the Advection-Diffusion problem (orange) and an initial condition (blue) used for "extrapolative" predictions. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

3.1.4. Inference and learning

Given the multinomial p_{cf} in Equation (21), we employed the following procedure for generating training data which consists of n numerical experiments in which the FG model is randomly initialized and propagated for one coarse time-step Δt i.e. for $T = \frac{\Delta t}{\delta t}$ microscopic time-steps. In particular:

- For $i = 1, \ldots, n$, we:
  - sample CG initial state $\hat{\boldsymbol{X}}_0^{(i)}$ from a density $p_{c,0}(\hat{\boldsymbol{X}}_0^{(i)})$.
  - sample FG initial state $\hat{\boldsymbol{x}}_0^{(i)}$ from $p_{cf}(\hat{\boldsymbol{x}}_0^{(i)}|\boldsymbol{X}_0^{(i)})$.
  - solve the (discretized) FG model for $\frac{\Delta t}{\delta t}$ microscopic time-steps and record final state $\hat{\boldsymbol{x}}_{\Delta t}^{(i)}$

The generated FG data $\{\hat{\boldsymbol{x}}_{\Delta t}^{(i)}\}_{i=1}^n$ over a *single* CG time-step are used subsequently to draw inferences on the CG model states and parameters (section 2.4). We note that longer time sequences could readily be generated (albeit at an increased cost). The number of samples n is also something that can be selected adaptively since inferences and predictions can be updated as soon as more data become available. The density $p_{c,0}(\boldsymbol{X}_0^{(i)})$ from which initial CG states are drawn, can be selected quite flexibly and some indicative samples are shown in Fig. 2 for the advection-diffusion case, and in Fig. 12 for the inviscid-Burgers' case. In summary, the data $\mathcal{D}$ employed, apart from $\{\hat{\boldsymbol{x}}_{\Delta t}^{(i)}\}_{i=1}^n$ above consists of the virtual observables $\{\hat{\boldsymbol{R}}_0^{(1:n)}, \hat{\boldsymbol{c}}_1^{(1:n)}\}$.

As a result of the data employed and the parametrization adopted, we have $\boldsymbol{X}_{\Delta t}^{(1:n)}$ as the sole latent vector and $\boldsymbol{\theta}_c, \boldsymbol{\tau}$ as the unknown (hyper)parameters. Since only a single CG time-step was considered, we omitted the slowness prior (see Equation (16)). Hence we sought an approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{X}_{\Delta t}, \boldsymbol{\theta}_c, \boldsymbol{\tau})$ (Equation (17)) which we factorized as in Equation (18) as follows:

$$q_{\boldsymbol{\phi}}(\boldsymbol{X}_{\Delta t}^{(1:n)}, \boldsymbol{\theta}_c, \boldsymbol{\tau}) = \left[\prod_{i=1}^{n} q_{\boldsymbol{\phi}}(\boldsymbol{X}_{\Delta t}^{(i)})\right] q(\boldsymbol{\theta}_c) q(\boldsymbol{\tau}) \tag{27}$$

Upon substitution in Equation (19), this yields the following ELBO:

$$\begin{aligned} \mathcal{F}(q_{\boldsymbol{\phi}}(\boldsymbol{X}_{\Delta t}^{(1:n)}, \boldsymbol{\theta}_c, \boldsymbol{\tau})) \quad &= \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p(\mathcal{D}|\, \boldsymbol{X}_{\Delta t}^{(1:n)}, \boldsymbol{\theta}_c)\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p(\boldsymbol{\theta}_c \mid \boldsymbol{\tau})\right] \\ &+ \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p(\boldsymbol{\tau})\right] - \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log q_{\boldsymbol{\phi}}\right] \end{aligned} \tag{28}$$

where:

$$p(\mathcal{D}|\boldsymbol{X}_{\Delta t}^{(1:n)}, \boldsymbol{\theta}_c) = p(\hat{\boldsymbol{x}}_{\Delta t}^{(1:n)}|\boldsymbol{X}_{\Delta t}^{(1:n)})\; p(\hat{\boldsymbol{R}}_0^{(1:n)}|\boldsymbol{X}_{\Delta t}^{(1:n)}, \boldsymbol{\theta}_c)\; p(\hat{\boldsymbol{c}}_1^{(1:n)}|\boldsymbol{X}_{\Delta t}^{(1:n)}) \tag{29}$$

Based on Equation (28) the optimal variational posterior densities can be obtained as:

$$\log q^{opt}(\boldsymbol{\theta}_c) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{X}_{\Delta t}^{(1:n)})}\left[\log\; p(\hat{\boldsymbol{R}}_0^{(1:n)}|\boldsymbol{X}_{0:1\Delta t}^{(1:n)}, \boldsymbol{\theta}_c)\right] + \mathbb{E}_{q(\boldsymbol{\tau})}\left[\log\; p(\boldsymbol{\theta}_c \mid \boldsymbol{\tau})\right] \tag{30}$$

Algorithm 3: Inference algorithm for particle systems.

Result: $\{q_{\boldsymbol{\phi}}(\boldsymbol{X}_{\Delta t}^{(i)})\}_{i=1}^{n},\ q(\boldsymbol{\theta}_c),\ q(\boldsymbol{\tau})$
Data: $\{\boldsymbol{X}_0^{(i)}, \hat{\boldsymbol{x}}_{\Delta t}^{(i)}\}_{i=1}^{n}$
Initialize the parameters for the variational distributions;
Set iteration counter w to zero;
Set convergence limit ϵ;
while $||parameters_w - parameters_{w-1}||^2 > \epsilon$ **do**
  for $i \leftarrow 1$ **to** n **do**
    Update $q_{\boldsymbol{\phi}}(\boldsymbol{X}_{\Delta t}^{(i)})$ by maximizing the ELBO (see Equation (28))
  end
  update $q(\boldsymbol{\theta}_c)$ according to Equation (33) and Equation (34) ;
  update $q(\boldsymbol{\tau})$ according to Equation (35) ;
  update the iteration counter by one ;
end

$$\log q^{opt}(\boldsymbol{\tau}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_c)}[\log\ p(\boldsymbol{\theta}_c \mid \boldsymbol{\tau})] + \log\ p(\boldsymbol{\tau}) \tag{31}$$

$$\begin{aligned}\log q_{\boldsymbol{\phi}}^{opt}(X_{\Delta t}^{(i)}) \quad &= \log\ p_{cf}(\boldsymbol{x}_{\Delta t}^{i} \mid \boldsymbol{X}_{\Delta t}^{i}) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_c)}\left[\log\ p(\hat{\boldsymbol{R}}_0^{(i)} | \boldsymbol{X}_{0:1\Delta t}^{(i)}, \boldsymbol{\theta}_c)\right] \\ &\quad + \log\ p(\hat{\boldsymbol{c}}_1^{(i)} | \boldsymbol{X}_{\Delta t}^{(i)})\end{aligned} \tag{32}$$

The equations above are coupled and a closed-form solution can be obtained only for the first two. In particular, the optimal posterior approximation for $\boldsymbol{\theta}_c$ is a multivariate normal with mean $\mu_{\boldsymbol{\theta}_c}$ and covariance $\boldsymbol{S}_{\boldsymbol{\theta}_c}$.

$$\mathbf{S}_{\boldsymbol{\theta}_c}^{-1} = \sigma_R^{-2} \sum_{i=1}^{n} \sum_{j=1}^{d_c} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{X}_{\Delta t}^{(i)})}\left[\boldsymbol{\psi}^{(j)}(\boldsymbol{X}_{\Delta t}^{(i)}) \left(\boldsymbol{\psi}^{(j)}(\boldsymbol{X}_{\Delta t}^{(i)})\right)^T\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\tau})}[diag(\boldsymbol{\tau})] \tag{33}$$

$$\mathbf{S}_{\boldsymbol{\theta}_c}^{-1} \boldsymbol{\mu}_{\boldsymbol{\theta}_c} = \sigma_R^{-2} \sum_{i=1}^{n} \sum_{j=1}^{d_c} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{X}_{\Delta t}^{(i)})}\left[\boldsymbol{\psi}^{(j)}(\boldsymbol{X}_{\Delta t}^{(i)})\right] \tag{34}$$

where the vector $\boldsymbol{\psi}^{(j)}$ consists of the M feature functions $\psi_m^{(j)}$ in Equation (22). The optimal posterior approximation for the vector $\boldsymbol{\tau}$ of the hyperparameters $\{\tau_m\}_{m=1}^{M}$ reduces to a product of independent Gamma-densities [62] with parameters γ_m and δ_m which are given by:

$$\gamma_m = \gamma_0 + 0.5, \qquad \delta_m = \delta_0 + \frac{1}{2}\left(\mu_{\boldsymbol{\theta}_c,m} + S_{\boldsymbol{\theta}_c,(m,m)}\right), \quad m = 0, 1, \ldots, M = dim(\boldsymbol{\theta}_c) \tag{35}$$

Finally and since closed-form updates for the optimal posterior $q_{\boldsymbol{\phi}}^{opt}(X_{\Delta t}^{(i)})$ are impossible, we employed Stochastic Variational Inference (SVI) as detailed in section 2.4 by assuming a multivariate lognormal (in order to ensure positivity of $X_{\Delta t,j}$) with parameters $\boldsymbol{\phi} = \{\boldsymbol{\mu}_i, \boldsymbol{S}_i\}_{i=1}^{n}$.[6] Noisy gradients with respect to the parameters $\boldsymbol{\phi}$ were estimated with Monte Carlo and the reparametrization trick [56] and $\boldsymbol{\phi}$ were updated using stochastic gradient ascent (the ADAM algorithm of [63] in particular). The inference steps are summarized in Algorithm 3.

3.1.5. Advection-diffusion system

For the simulations presented in this section $d_f = 250 \times 10^3$ particles were used, which, at each microscopic time step $\delta t = 2.5 \times 10^{-3}$ performed random, non-interacting, jumps of size $\delta s = \frac{1}{640}$, either to the left with probability $p_{left} = 0.1875$ or to the right with probability $p_{right} = 0.2125$. The positions were restricted in $[-1, 1]$ with periodic boundary conditions. It is well-known [64] that in the limit (i.e. $d_f \to \infty$) the particle density $\rho(s,t)$ can be modeled with an advection-diffusion PDE with diffusion constant $D = (p_{left} + p_{right})\frac{\delta s^2}{2\delta t}$ and velocity $v = (p_{right} - p_{left})\frac{\delta s}{\delta t}$:

$$\frac{\partial \rho}{\partial t} + v\frac{\partial \rho}{\partial s} = D\frac{\partial^2 \rho}{\partial s^2}, \qquad s \in (-1, 1). \tag{36}$$

For the CG description, 64 bins were employed i.e. $d_c = 64$ and a time step $\Delta t = 2$ (see Table 2). Furthermore we employed first- and second-order feature function as in Equation (22) with a neighborhood size $H = 5$ which implies a total of $M = 77$ unknown parameters $\boldsymbol{\theta}_c$. We incorporate virtual observables pertaining to the residuals $\hat{\boldsymbol{R}}_0$ with $\sigma_R^2 = 10^{-6}$

[6] Diagonal covariances $\boldsymbol{S}_i$ were employed.

Table 2
FG/CG state-space dimensions and FG/CG time-steps for particle systems investigated.

	$d_f = dim(\boldsymbol{x})$	$d_c = dim(\boldsymbol{X})$	FG time-step δt	CG time-step Δt
Advection-diffusion	250×10^3	≤ 64	2.5×10^{-3}	2
Inviscid Burgers	500×10^3	≤ 128	2.5×10^{-3}	4

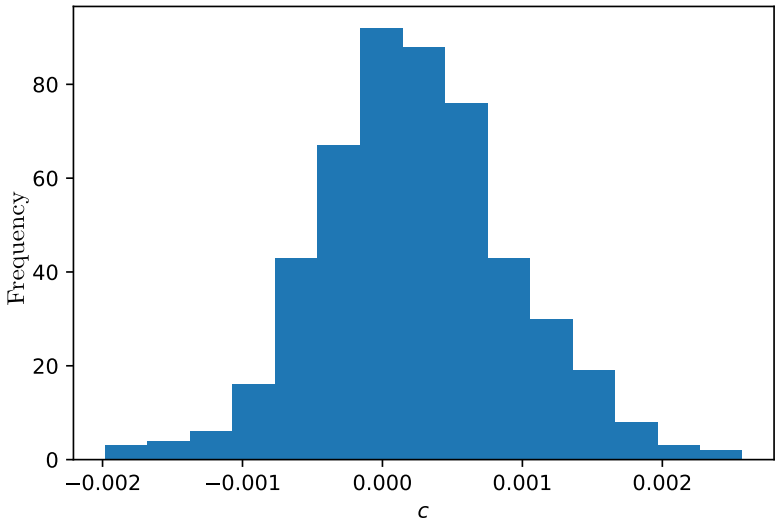


Fig. 3. Histogram of the mass constraint c_1.

(Equation (7))[7] and the virtual observables $\hat{\boldsymbol{c}}_1$ pertaining to the conservation-of-mass constraint with $\sigma_c^2 = 10^{-10}$ (Equation (9)).

We employed $n = 32$ and $n = 64$ time sequences for training that were generated as detailed in section 3.1.4 with initial conditions $\{\boldsymbol{X}_0^{(i)}\}_{i=1}^n$ such as the ones seen in Fig. 2. The initial conditions were generated by sampling the amplitude of a *sine* function, which was shifted up to ensure all values are positive and then normalized.

Fig. 3 provides a histogram of the function values of the conservation-of-mass constraint $\left\{c_1(\boldsymbol{X}_{\Delta t}^{(i)})\right\}_{i=1}^n$ upon convergence. The small values suggest that this has been softly incorporated in the CG states. A similar histogram for the norm of the residuals $\left\{\boldsymbol{R}_0(\boldsymbol{X}^{(i)})\right\}_{i=1}^n$ is depicted in Fig. 4 which also suggests enforcement of the CG evolution with the parameters $\boldsymbol{\theta}_c$ learned from the data. The evolution of the posterior mean $\boldsymbol{\mu}_{\boldsymbol{\theta}_c}$ (Equation (34)) of (a subset of) these parameters over the iterations of the SVI is depicted in Fig. 5. Therein, and more clearly in Fig. 6, one can observe the ability of the ARD prior to deactivate the vast majority of the right-hand-side feature functions and reveal a small subset of non-zero, salient terms. Both with $n = 32$ and $n = 64$ training data sequences, only parameters $\boldsymbol{\theta}_c$ associated with first-order-interactions (Equation (22)) are activated. In particular, these are $\theta_{c,-3}^{(1)}$ and $\theta_{c,1}^{(1)}$ which are associated with the feature functions $X_{t,j-3}$ and $X_{t,j+1}$ respectively in Equation (22). This shares similarities with a finite-difference discretization scheme for the advection-diffusion and could be considered as an upwind scheme. The two identified coefficients do not form a centered difference operator but the center of the operator is shifted to the left and therefore takes into account the direction of the particle movement. As the value of the coefficients is not exactly the same the diffusive part is also captured.

Fig. 7 depicts one of the inferred CG states $\boldsymbol{X}_{\Delta t}^{(i)}$ as well as the associated posterior uncertainty. Once the CG evolution law is learned, this state can be propagated into the future as detailed in section 2.5 in order to generate predictions. Indicative predictions (under "interpolative" conditions) can be seen in Fig. 8 where the particle density $\rho_x(t, s)$ up to $25\Delta t$ into the future is drawn. The latter as well as the associated uncertainty bounds are estimated directly from the reconstructed FG states. As one would expect, the predictive uncertainty grows, the further into the future one tries to predict. Fig. 9 compares the predictive performance as a function of the training data used i.e. $n = 32$ or $n = 64$. In both cases, the ground truth is enveloped and as one would expect, more training data lead to smaller uncertainty bounds.

We also tested the trained model (on $n = 64$) under "extrapolative" conditions i.e. for a different initial condition than the ones included in the training data (Fig. 2). The predictive estimates in Fig. 10 show very good agreement with the reference solution. It is important to point out that the model can correctly advect and diffuse the particle-bump initially introduced around $s = 0.5$ which suggests that the CG dynamics learned reflect the most important features of the problem.

[7] A very interesting possibility which is not explored here would be to learn σ_R^2 i.e. the strength of the enforcement of the CG evolution law from the data. This would increase the flexibility of the model in cases where the vocabulary of the feature functions selected in the right-hand side of the CG dynamics is not rich enough.

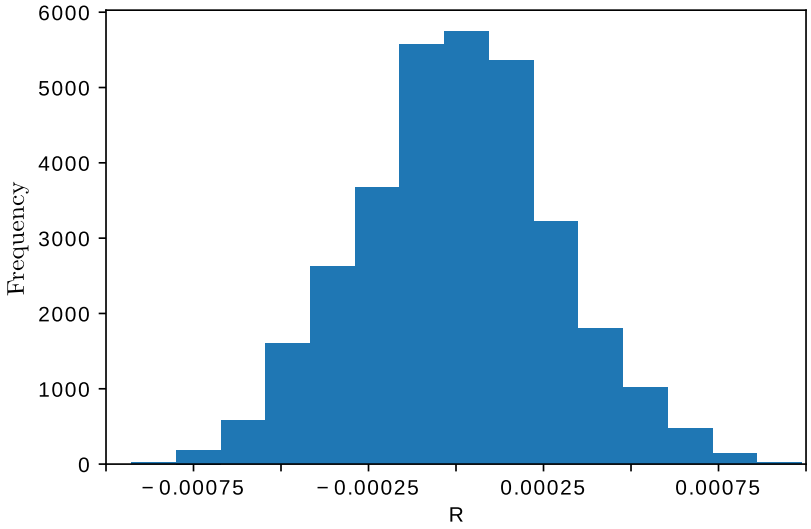


Fig. 4. Histogram of the norm of the residual $\boldsymbol{R}_0$.

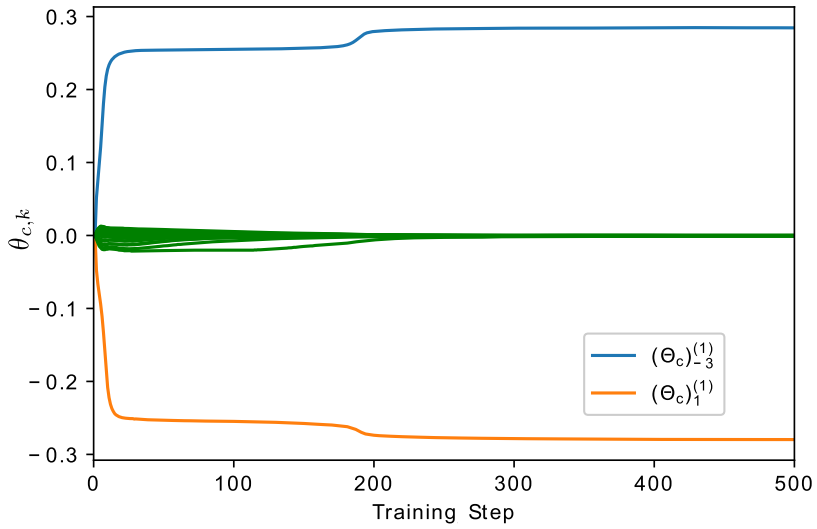


Fig. 5. Evolution of a subset of $\boldsymbol{\theta}_c$ parameters with respect to the iterations of the SVI for $n = 64$.

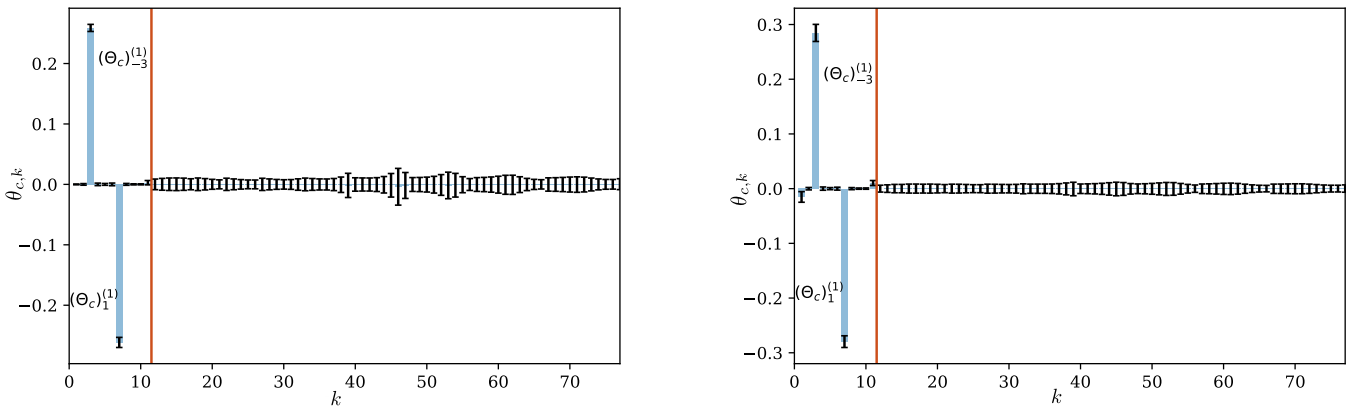


Fig. 6. Comparison of the inferred parameters $\boldsymbol{\theta}_c$ for $n = 32$ (left) and $n = 64$ (right) training data sequences. The black bars indicate $+/-1$ standard deviation. The red vertical line separates first- from second-order coefficients.

Finally, in Fig. 11, the evolution of the mass constraint into the future is depicted and good agreement with the target value is observed.

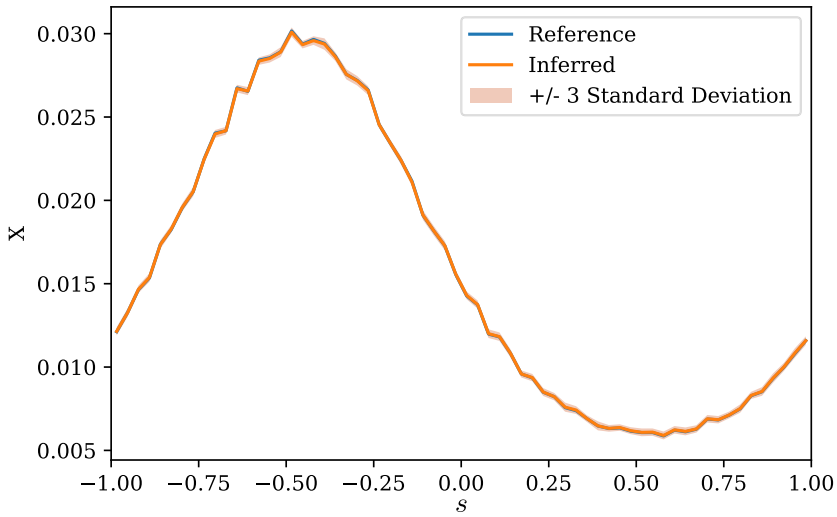


Fig. 7. Inferred CG state $\boldsymbol{X}_{\Delta t}^{(i)}$ for a data sequence i. Reference is obtained by sorting the particles into bins according to their position.

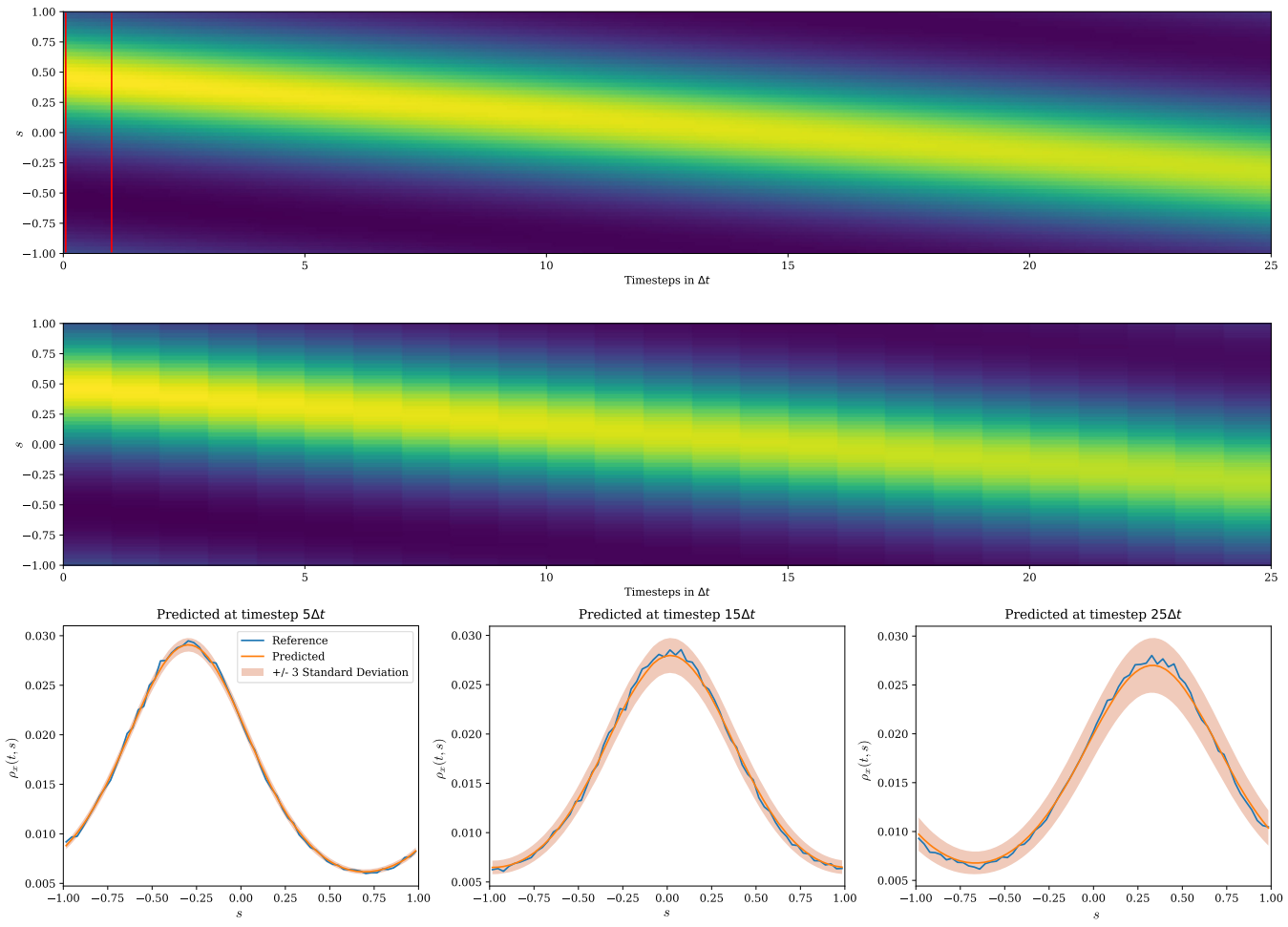


Fig. 8. Prediction based on an initial condition contained in the training data. Top: reference data (the vertical lines indicate the time instances with given data). Middle: predictive posterior mean. Bottom: snapshots at three different time instances.

3.1.6. *Burgers' system*

The second test-case involved an FG system of $d_f = 500 \times 10^3$ particles which perform *interactive* random walks i.e. the jump performed at each fine-scale time-step $\delta t = 2.5 \times 10^{-3}$ depends on the positions of the other walkers. In particular we adopted interactions as described in [65,66,59] so as, in the limit (i.e. when $d_f \to \infty, \delta t \to 0, \delta s \to 0$), the particle density $\rho(s,t)$ follows the inviscid Burgers' equation:

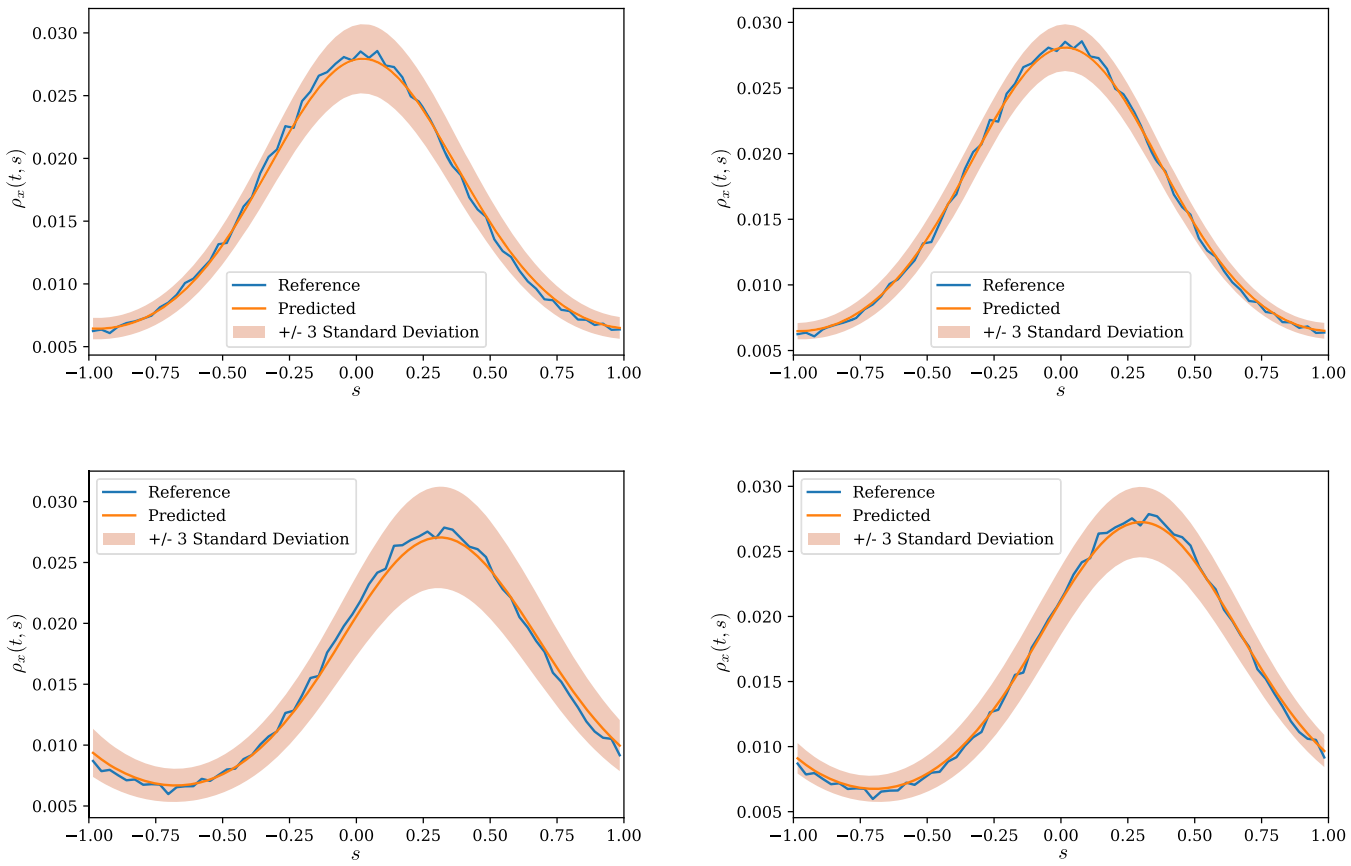


Fig. 9. Comparison of the predictions for $n = 32$ (left) and $n = 64$ (right) at $15\Delta t$ (top) and $25\Delta t$ (bottom).

$$\frac{\partial \rho}{\partial t} + \frac{1}{2}\frac{\partial \rho^2}{\partial s} = 0, \qquad s \in (-1, 1). \tag{37}$$

For the CG description, 128 bins were employed i.e. $d_c = 128$ and a time step $\Delta t = 4$ (see Table 2). As compared with the previous case, we enlarged the neighborhood size H in the first- and second-order interactions to $H = 8$, which yielded $M = 170$ right-hand-side terms in Equation (22). We incorporate virtual observables pertaining to the residuals $\hat{\boldsymbol{R}}_0$ with $\sigma_R^2 = 10^{-7}$ (Equation (7)) and the virtual observables $\hat{\boldsymbol{c}}_1$ pertaining to conservation-of-mass constraint with $\sigma_c^2 = 10^{-10}$ (Equation (9)).

We employed $n = 32$, $n = 64$ and $n = 128$ time sequences for training that were generated as detailed in section 3.1.4 with initial conditions $\{\boldsymbol{X}_0^{(i)}\}_{i=1}^n$ such as the ones seen in Fig. 12. They were generated by randomizing the width and height of a triangular profile.

Fig. 13 provides a histogram of the function values of the conservation-of-mass constraint $\left\{c_1(\boldsymbol{X}_{\Delta t}^{(i)})\right\}_{i=1}^n$ upon convergence. The small values suggest that this has been softly incorporated in the CG states. A similar histogram for the norm of the residuals $\left\{\boldsymbol{R}_0(\boldsymbol{X}^{(i)})\right\}_{i=1}^n$ is depicted in Fig. 14 which also suggests enforcement of the CG evolution with the parameters $\boldsymbol{\theta}_c$ learned from the data. The evolution of the posterior mean $\boldsymbol{\mu}_{\boldsymbol{\theta}_c}$ (Equation (34)) of (a subset of) these parameters over the iterations of the SVI is depicted in Fig. 15. As in the previous example, in Fig. 16 one can observe the ability of the ARD prior model to yield sparse solutions for the right-hand side of the CG evolution law. For all three training datasets with $n = 32, 64, 128$ time-sequences, only parameters $\boldsymbol{\theta}_c$ associated with second-order-interactions (Equation (22)) are activated. In particular, these are the negative coefficient $\boldsymbol{\theta}_{c,(0,0)}^{(2)}$ (in all three cases) as well as different second-order coefficients. In the cases of $n = 32$ and $n = 64$ two coefficients are found with positive mean and high posterior uncertainty, but they also have negative posterior correlation (correlation coefficient of -0.88). As all activated coefficients pertain to feature-functions involving the actual bin or bins to the left, the learned evolution law could be interpreted as an upwind scheme, which takes the direction of the Burgers' flow into account. Such schemes are considered advantageous for numerical simulations of fluid flows.

Fig. 17 depicts one of the inferred CG states $\boldsymbol{X}_{\Delta t}^{(i)}$ as well as the associated posterior uncertainty. Given the learned CG dynamics, this state can be propagated into the future as detailed in section 2.5 in order to generate predictions. Indicative predictions (under "interpolative" conditions) can be seen in Fig. 18 where the particle density up to $25\Delta t$ into the future is drawn. The latter as well as the associated uncertainty bounds are estimated directly from the reconstructed FG states. As in the previous example, the predictive uncertainty grows, the further into the future one tries to predict. Fig. 19 compares

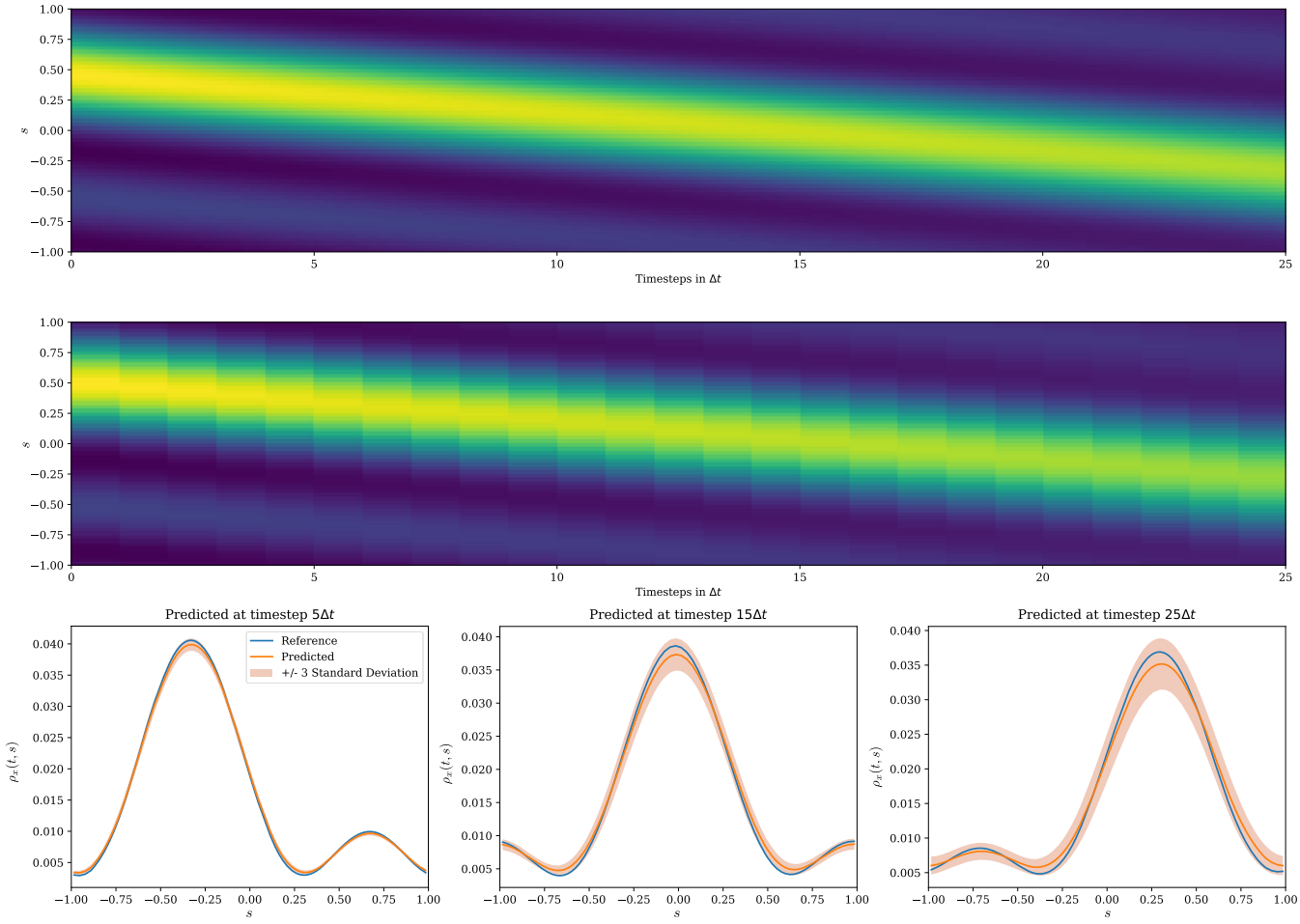


Fig. 10. Prediction based on an initial condition NOT contained in the training data. Top: reference data. Middle: predictive posterior mean. Bottom: snapshots at three different time instances.

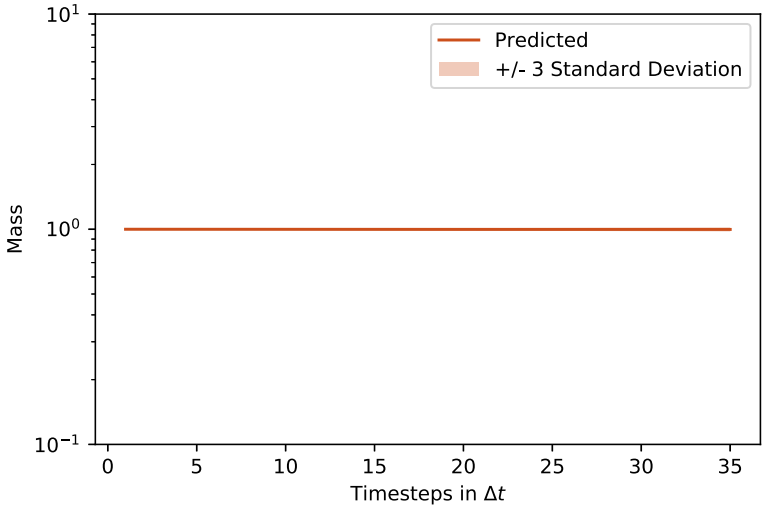


Fig. 11. Evolution of the mass constraint (target value is 1) in time including future time-instants. "Predicted" corresponds to the posterior mean.

the predictive performance as a function of the training data used i.e. $n = 32$ or $n = 64$. The increase in data leads for this example to a better fit of the posterior mean to the reference, which captures the location of the shock more precisely. The predictive uncertainty bounds are particularly large at the location of the shock which is the most challenging component in such systems.

We also test the trained model (on $n = 64$) under "extrapolative" conditions i.e. for a "bimodal" initial condition which was quite different from the ones included in the training data (Fig. 12). The predictive estimates in Fig. 20 show very

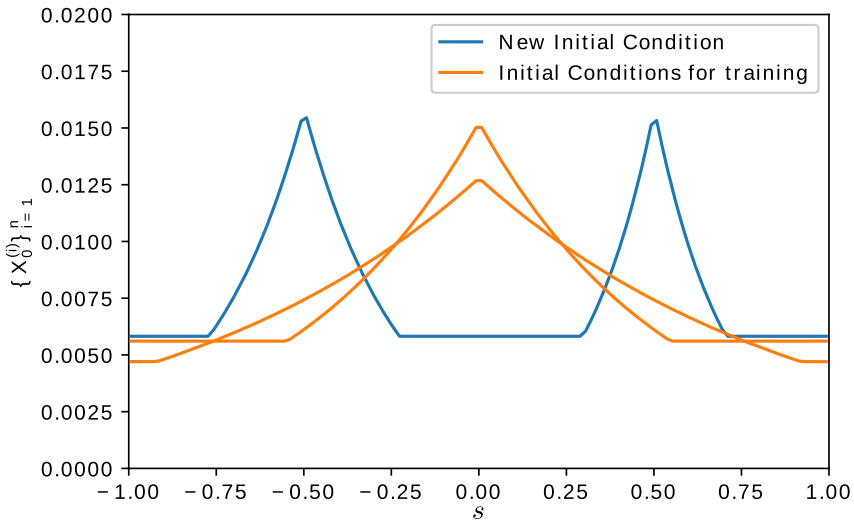


Fig. 12. Sample initial conditions $\left\{\boldsymbol{X}_0^{(i)}\right\}_{i=1}^{n}$ for the Burgers' problem (orange) and an initial condition (blue) used for "extrapolative" predictions.

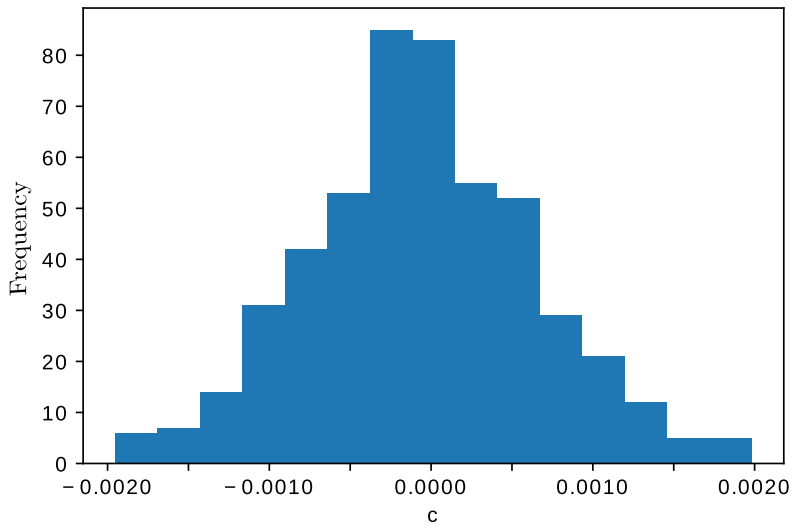


Fig. 13. Histogram of the mass constraint c_1.

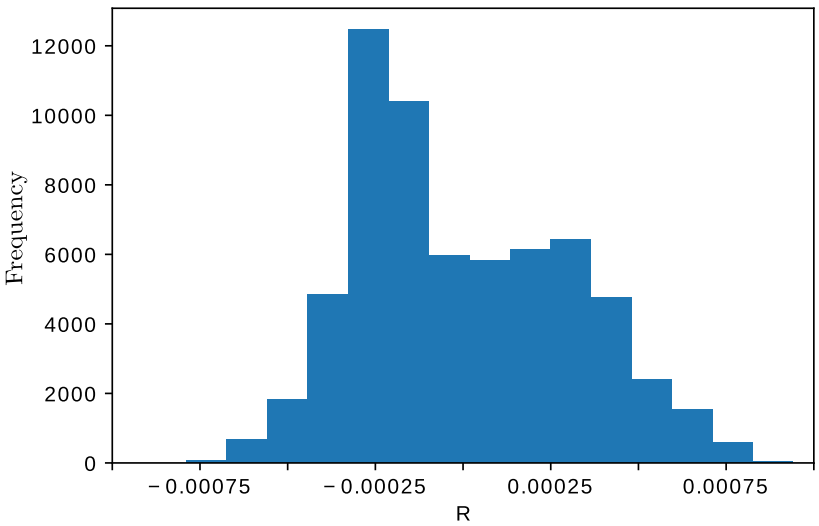


Fig. 14. Histogram of the norm of the residual constraint $\boldsymbol{R}_0$.

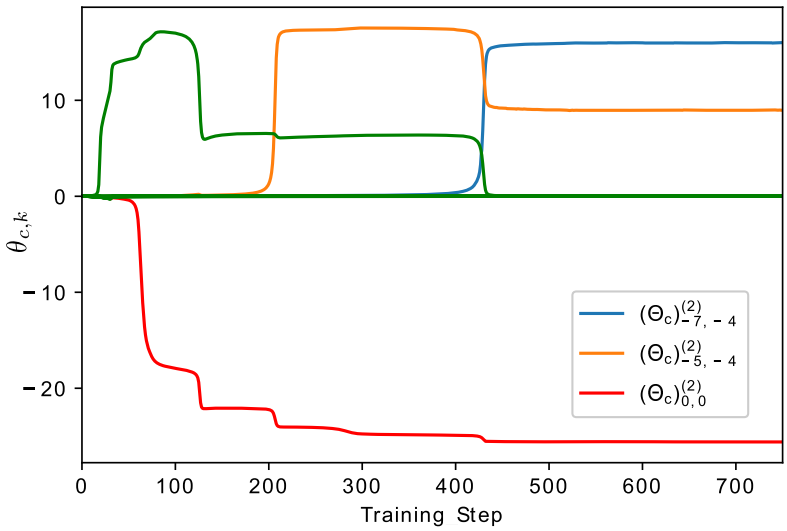


Fig. 15. Evolution of a subset of $\boldsymbol{\theta}_c$ parameters with respect to the iterations of the SVI for $n = 64$.

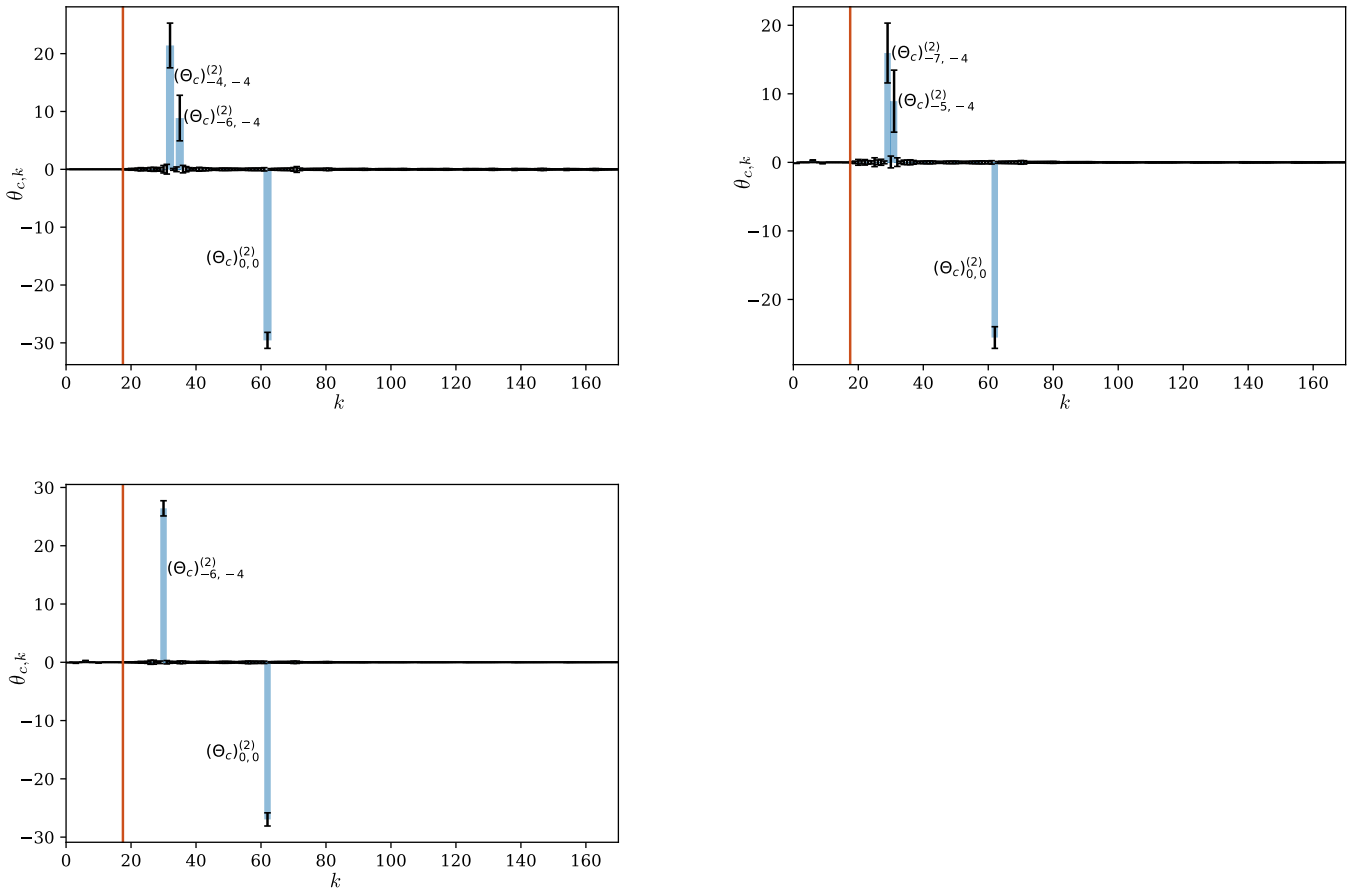


Fig. 16. Comparison of the inferred parameters $\boldsymbol{\theta}_c$ for $n = 32$ (top-left), $n = 64$ (top-right) and $n = 128$ (bottom-left) training data. The black bars indicate $+/-1$ standard deviation. The red vertical line separates first- from second-order coefficients.

good agreement with the reference solution. We want to point out that the trained model is capable of capturing the development, the position as well as the propagation of a shock front. Finally, in Fig. 21, the evolution of the mass constraint into the future is depicted and good agreement with the target value is observed.

3.2. Nonlinear pendulum

In this final example we consider time sequences of images of a nonlinear pendulum in two dimensions as in [27].

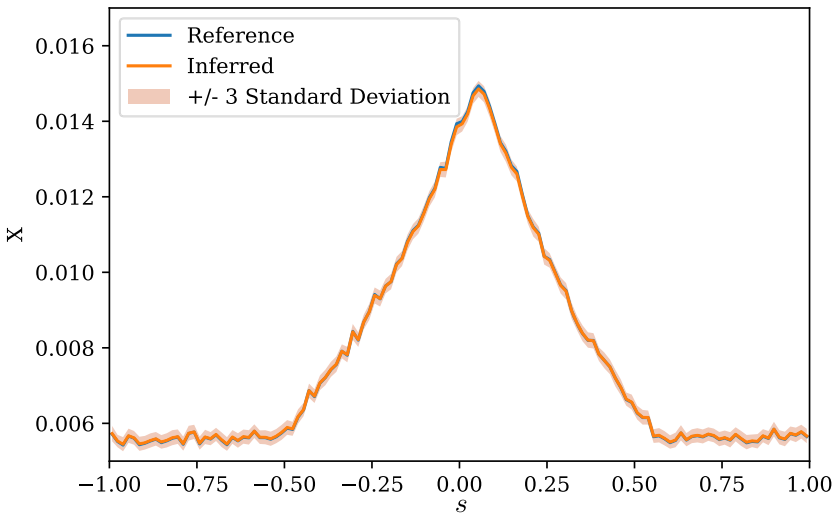


Fig. 17. Example of inferred CG state $\boldsymbol{X}_{\Delta t}^{(i)}$ for data sequence i.

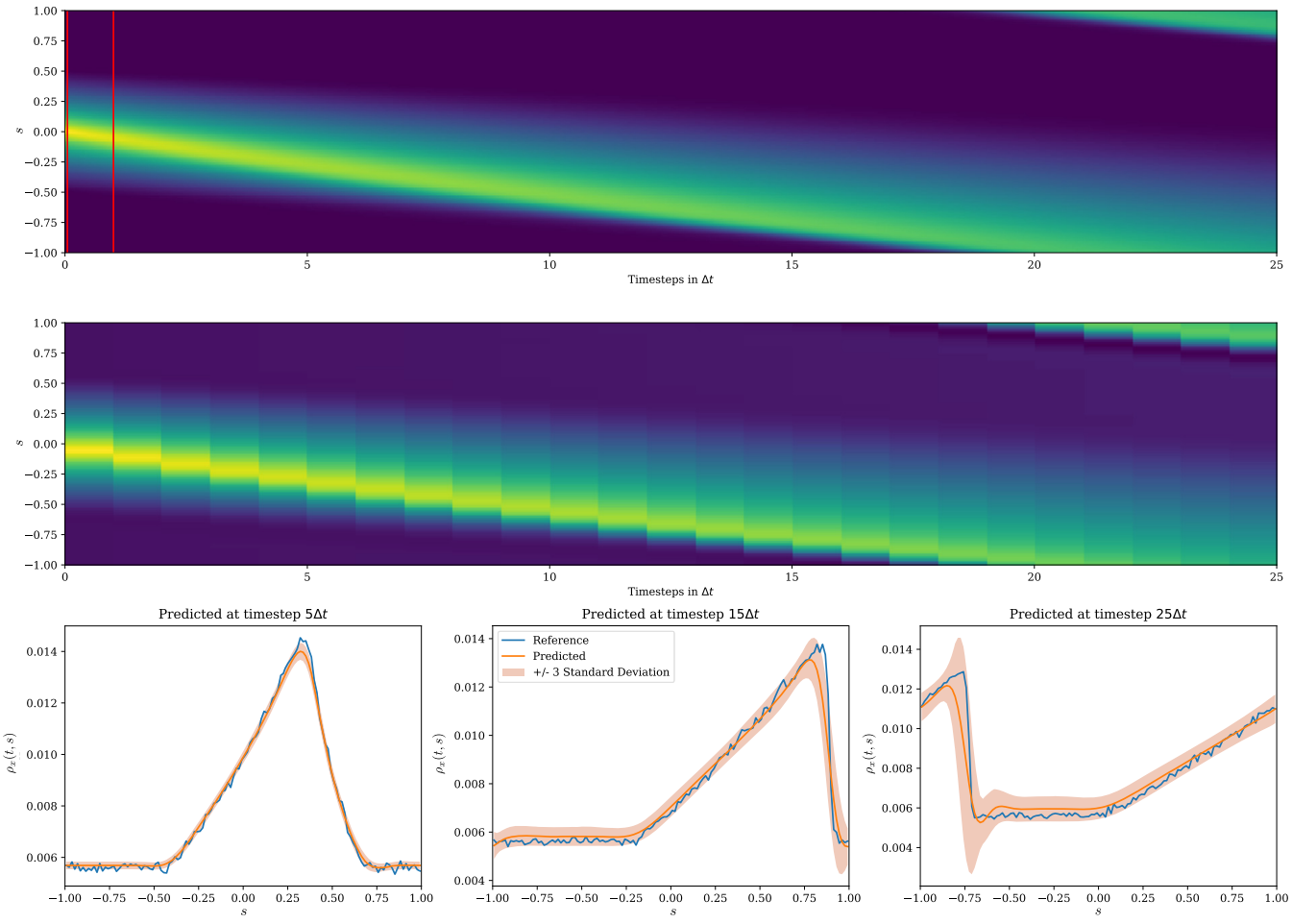


Fig. 18. Prediction based on an initial condition contained in the training data. Top: reference data (the vertical lines indicate the time instances with given data). Middle: predictive posterior mean. Bottom: snapshots at three different time instances.

3.2.1. FG model

For the FG data we generate a series of black-and-white images of a moving disk tied on a string and forming a pendulum (see Fig. 31). Each image consists of 29×29 pixels each and each pixel's value was either 1 (occupied) or -1 (unoccupied). Hence $\boldsymbol{x}_t$ was a $d_f = 29^2 = 581$-dimensional vector of binary variables. The dynamics of the pendulum can be fully described by the rotation angle y_t which follows a nonlinear, second-order ODE of the form:

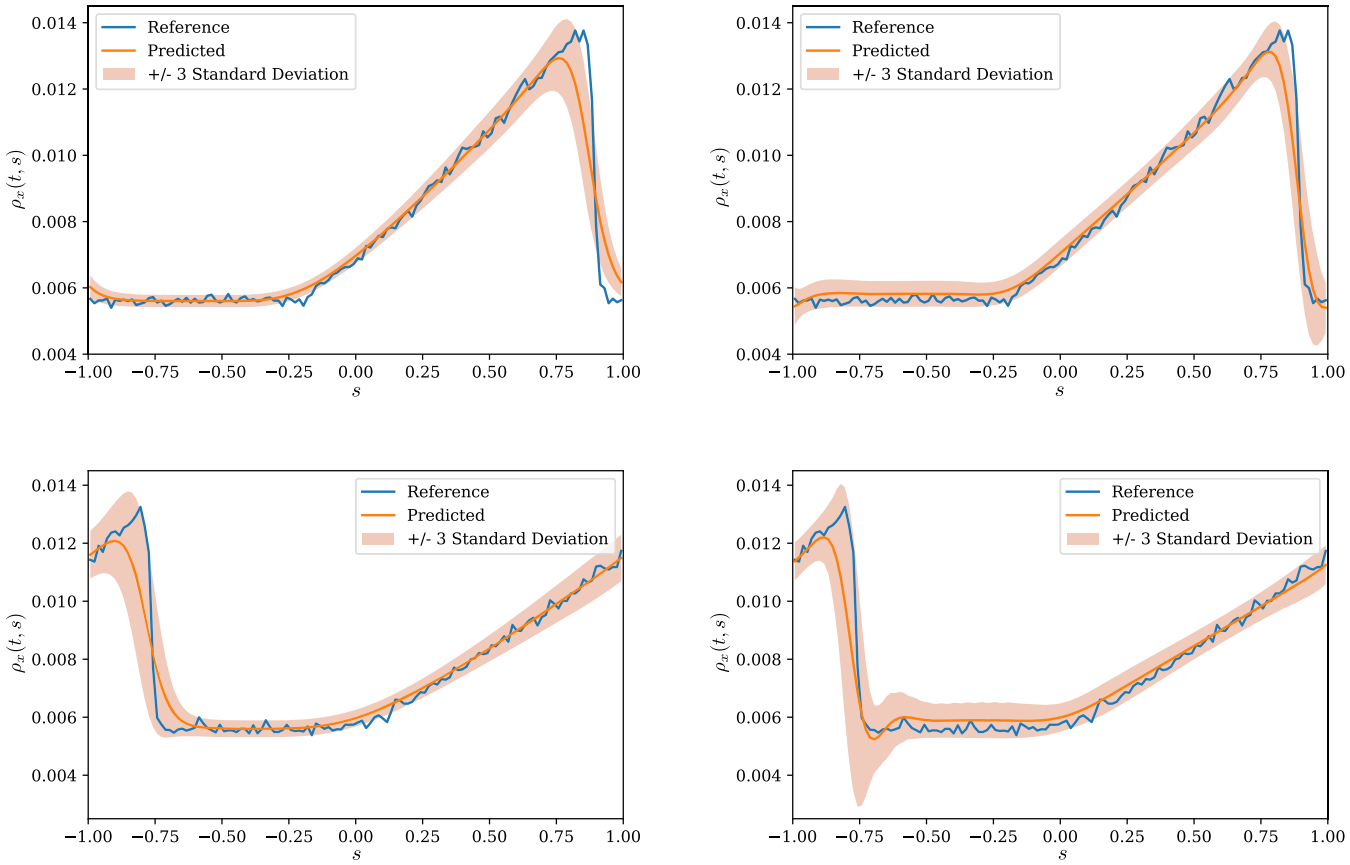


Fig. 19. Comparison of the predictions for $n = 32$ (left) and $n = 64$ (right) training data at $15\Delta t$ (top) and $25\Delta t$ (bottom).

$$\ddot{y}_t + \sin(y_t) = 0 \tag{38}$$

The primary goal is to identify the right CG variables as well as CG dynamics solely from image data i.e. binary vectors $\{\hat{\boldsymbol{x}}^{(i)}_{0:T\Delta t}\}_{i=1}^{n}$ collected over T time-steps as the pendulum is initialized from n states/positions. The length of time sequences in the following numerical results was $T = 74$ and the CG time-step $\Delta t = 0.05$.[8] We also considered the effect of missing data i.e. only observing a subset of the $T + 1$ values in each sequence and present respective results in Section 3.2.6.

3.2.2. CG variables and coarse-to-fine mapping

The only knowledge introduced a priori with regards to the CG variables $\boldsymbol{X}_t$ is that $dim(\boldsymbol{X}) = d_c = 2$. We intend to investigate procedures that can automatically identify d_c i.e. the number of CG variables. We note at this stage that such efforts could be guided by the ELBO $\mathcal{F}$ (e.g. Equation (19)) which approximates the model evidence and therefore provides a natural Bayesian score for comparing models with different numbers of CG variables.

The other pertinent model component is the coarse-to-fine map which is enabled by the $p_{cf}(\boldsymbol{x}_t|\boldsymbol{X}_t)$ (section 2.3). To that end, we employed the following logistic model[9]:

$$p_{cf}(\boldsymbol{x}|\boldsymbol{X}) = \prod_{s=1}^{d_f} p_{cf}(x_s|\boldsymbol{X}) \tag{39}$$

with

$$p_{cf}(x_s|\boldsymbol{X}) = \begin{cases} \dfrac{1}{1+\exp(-G_s(\boldsymbol{X};\boldsymbol{\theta}_{cf}))} & \text{for } x_s = 1 \\ \dfrac{1}{1+\exp(+G_s(\boldsymbol{X};\boldsymbol{\theta}_{cf}))} & \text{for } x_s = 0 \end{cases} \tag{40}$$

[8] For the generation of images a *microscopic* time-step $\delta t = 0.01$ for the integration of Equation (38) was used.

[9] We omit the time-index t for clarity.

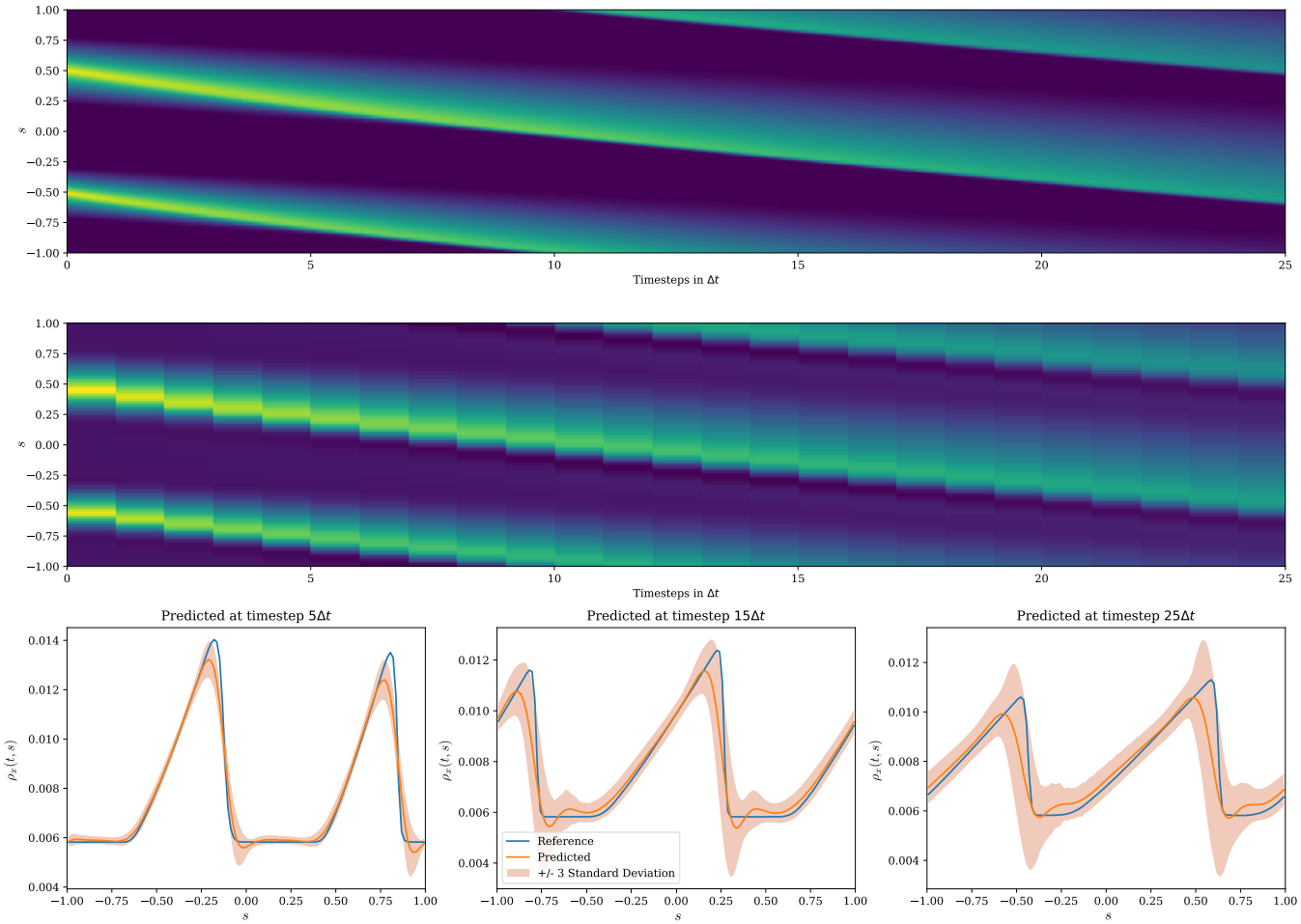


Fig. 20. Prediction based on an initial condition NOT contained in the training data. Top: reference data. Middle: predictive posterior mean. Bottom: snapshots at three different time instances.

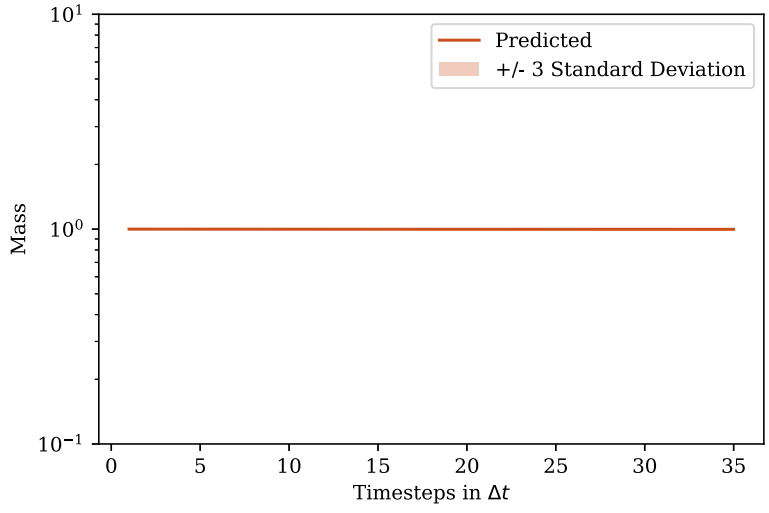


Fig. 21. Evolution of the mass constraint (target value is 1) in time including future time-instants. "Predicted" corresponds to the posterior mean.

where x_s is the value $(1, 0)$ of each of the pixels $s = 1, \ldots, d_f$. For the link functions $\{G_s\}_{s=1}^{d_f}$, we employed a deep neural net with weights $\boldsymbol{\theta}_{cf}$, the details of which are shown in Fig. 22. One fully connected layer followed by two transposed convolutional layers were found to be flexible enough to accurately represent the functions G_s. The CNNs were specifically chosen because of their ability to extract/map features from/to images.

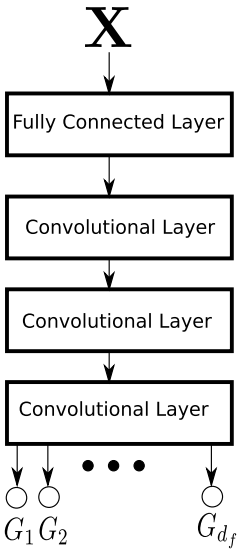


Fig. 22. Deep neural net employed for the link functions G_s (Equation (39)). After one dense layer which $32 \cdot 7 \cdot 7$ nodes and rectified linear unit activation function (ReLU), two two-dimensional transposed convolutional layers with 32 filters and a kernel size of 3 as well as a ReLU activation function are applied followed by one-last two-dimensional transposed convolutional layers with one filter, kernel size 3 and without activation to generate the functions G_s.

3.2.3. The CG evolution law and the virtual observables

With regards to the evolution law of the CG states $\boldsymbol{X}_t = \{X_{t,1}, X_{t,2}\}$, we postulate the following form:

$$\begin{aligned} \dot{X}_{t,1} &= F_1(\boldsymbol{X}_t, \boldsymbol{\theta}_c) = X_{t,2} \\ \dot{X}_{t,2} &= F_2(\boldsymbol{X}_t, \boldsymbol{\theta}_c) = \boldsymbol{\theta}_{\mathbf{c}}^T \boldsymbol{\psi}(X_{t,1}) = \textstyle\sum_{m=0}^{M} \theta_{c,m}\ \psi_m(X_{t,1}) \end{aligned} \tag{41}$$

where $\boldsymbol{\theta}_c$ denote the associated parameters. In total we employed $M = 101$ feature functions of the following type:

$$\psi_m(X) = \begin{cases} 1, & m = 0 \\ \sin(mX), & m = 1, \ldots, M/2 = 50 \\ \cos((m-50)X), & m = 51, \ldots, M = 100 \end{cases} \tag{42}$$

The form of Equation (41) implies a second-order ODE where the second CG variable plays the role of the velocity. With regards to the parameters $\boldsymbol{\theta}_c$, the sparsity-inducing ARD prior detailed in section 3.1.2 was employed.

To enforce the associated dynamics, we made use of the symplectic Euler time-discretization scheme, which is a first-order integrator, that is explicit in the first variable ($X_{t,1}$) and *implicit* in the other ($X_{t,2}$).[10] The associated virtual observables (see Equation (6)) were enforced with $\sigma_R^2 = 10^{-5}$.

3.2.4. Inference and learning

As in the previous examples (Equation (27)), the approximate posterior was factorized as:

$$q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T\,\Delta t}^{(1:n)}, \boldsymbol{\theta}_c, \boldsymbol{\tau}) = \left[\prod_{i=1}^{n} q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T\,\Delta t}^{(i)})\right] q(\boldsymbol{\theta}_c) q(\boldsymbol{\tau}) \tag{43}$$

and closed-form updates were used for $q(\boldsymbol{\theta}_c)$ (see Equations (33) and (34)) and $q(\boldsymbol{\tau})$ (see Equation (35)).

SVI was applied for the posterior densities $q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T\,\Delta t}^{(i)})$ on the vector of the latent CG states $\boldsymbol{X}_{0:T\,\Delta t}^{(i)}$ which we approximated with multivariate Gaussians. Since the posterior reveals the fine-to-coarse map which apart from insight can be used for predictive purposes as well, we employed an *amortized* version of SVI ([56]) i.e. explicitly accounted for the dependence of each $q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T\,\Delta t}^{(i)})$ on the corresponding FG observables $\hat{\boldsymbol{x}}_{0:T\,\Delta t}^{(i)}$ i.e.:

$$q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T\ \Delta t}^{(i)}) = \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\hat{\boldsymbol{x}}_{0:T\,\Delta t}^{(i)})\ ,\ \boldsymbol{S}_{\boldsymbol{\phi}}(\hat{\boldsymbol{x}}_{0:T\,\Delta t}^{(i)})\right) \tag{44}$$

The parameters $\boldsymbol{\phi}$ were the weights of a deep convolutional neural net, the architecture of which is shown in Fig. 23. This was chosen because it mirrors the DNN architecture employed for the coarse-to-fine map in Fig. 22.

[10] This corresponds to a multistep method in Equation (4) with $K = 1$, $a_0 = 1, a_1 = -1$, $\beta_0 = 0$ and $\beta_1 = -1$ for the explicit part and $K = 1$, $a_0 = 1, a_1 = -1$, $\beta_0 = -1$ and $\beta_1 = 0$ for the implicit part.

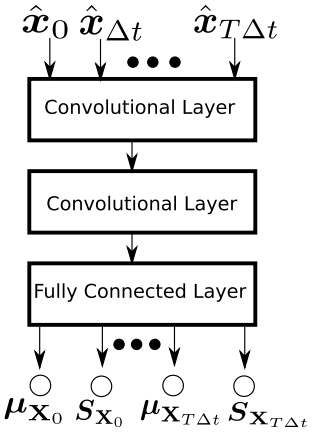


Fig. 23. DNN architecture for approximate posterior $q_{\boldsymbol{\phi}}$. The input consists of a time series of pictures of the pendulum and can therefore considered to be three-dimensional, where the first and second dimension are the number of pixels and the third dimension is the number of time steps available for training. This input is given to a three-dimensional convolutional layer with kernel size $(3, 3, 2)$, 32 filters and a ReLU activation followed by another three-dimensional convolutional layer with kernel size 2 in each dimension, 64 filters and a ReLU activation. The last layer is a fully connected layer with $2d_c \cdot T$ nodes and without activation to generate the mean and variance values for each time step of the inferred $\boldsymbol{X}$ coordinates.

Finally it should be mentioned that the "slowness" prior was employed on the hidden states $\boldsymbol{X}_{0:T\Delta t}^{(1:n)}$ as described in Equation (16).[11] Maximum-likelihood estimates for the hyperparameter $\sigma_{\boldsymbol{X}}^2$ were employed which readily arise by differentiating the ELBO $\mathcal{F}$ and which yield the following update equation:

$$\sigma_{\boldsymbol{X}}^2 = \frac{1}{n\, T\, d_c} \sum_{i=1}^{n} \sum_{l=0}^{T-1} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T\Delta t}^{(i)})} \left[\left| \boldsymbol{X}_{(l+1)\ \Delta t}^{(i)} - \boldsymbol{X}_{l\ \Delta t}^{(i)} \right|^2 \right] \tag{45}$$

Maximum likelihood estimates were also obtained for the parameters $\boldsymbol{\theta}_{cf}$ (Equation (39)) by numerically differentiating the ELBO $\mathcal{F}$ and performing Stochastic Gradient Ascent (SGA).

A general summary of the steps involved for the inference procedure can be found in Algorithm 4. For the implementation we made use of the Tensorflow framework [67].

Algorithm 4: Algorithm for the pendulum system.

Result: $\boldsymbol{\phi}$,$q(\boldsymbol{\theta}_c)$,$q(\boldsymbol{\tau})$,$\boldsymbol{\theta}_{cf}$,$\sigma_{\boldsymbol{X}}$
Data: $\hat{\boldsymbol{x}}_{0:T\Delta t}^{(1:n)}$
Initialize all required parameters;
Set iteration counter w to zero;
while $||ELBO_w - ELBO_{w-1}||^2 > \epsilon$ **do**
  Update the parameters $\boldsymbol{\theta}_{cf}$ and $\boldsymbol{\phi}$ by SGA of the ELBO (Equation (19)) ;
  update $q(\boldsymbol{\theta}_c)$ according to Equation (33) and Equation (34) ;
  update $q(\boldsymbol{\tau})$ according to Equation (35) ;
  update the parameter $\sigma_{\boldsymbol{X}}$ according to Equation (45);
  update the iteration counter by one;
end

3.2.5. Results

Each data sequence $\hat{\boldsymbol{x}}_{0:T\Delta t}^{(i)}$ used consisted of 75 images, i.e. $T = 74$, generated with a time-step $\Delta t = 0.05$ (Fig. 24). We investigated two cases for the number of data sequences i.e. $n = 16$ and $n = 64$. The data generation involved sampling uniformly the initial angle $y_0 \in [-\pi, \pi]$ and assuming zero initial velocity i.e. $\dot{y}_0 = 0$. We emphasize that none of the data sequences contained a complete oscillation of the pendulum i.e. always partial trajectories were observed.

Fig. 25 indicates the posterior means of the inferred $\boldsymbol{\theta}_c$ that parametrize the CG evolution law (Equation (41)) for $n = 16$ and $n = 64$. Of the 101 possible terms, only 2 are activated due the ARD prior.

Fig. 26 illustrates trajectories in the two-dimensional CG state-space obtained with various initial conditions for the CG model identified with $n = 16$ and $n = 64$ data sequences. The blue curves correspond to "interporlative" settings i.e. to the CG states of an observed sequence of images, whereas the orange curves to "extrapolative settings" i.e. to the CG states inferred by initializing the pendulum from an arbitrary position *not* contained in the training data. In Fig. 27 the predicted

[11] For the prior distribution $p_{c,0}(\boldsymbol{X}_0^{(i)})$ a Gaussian mixture distribution with means $+1.5$ and -1.5 and standard deviation 1.5 was used.

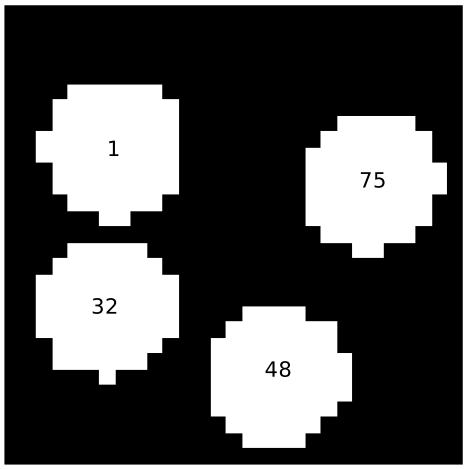


Fig. 24. Indicative positions of the pendulum in a data sequence $\hat{\boldsymbol{x}}^{(i)}_{0:T\,\Delta t}$. The number indicates the corresponding time-step.

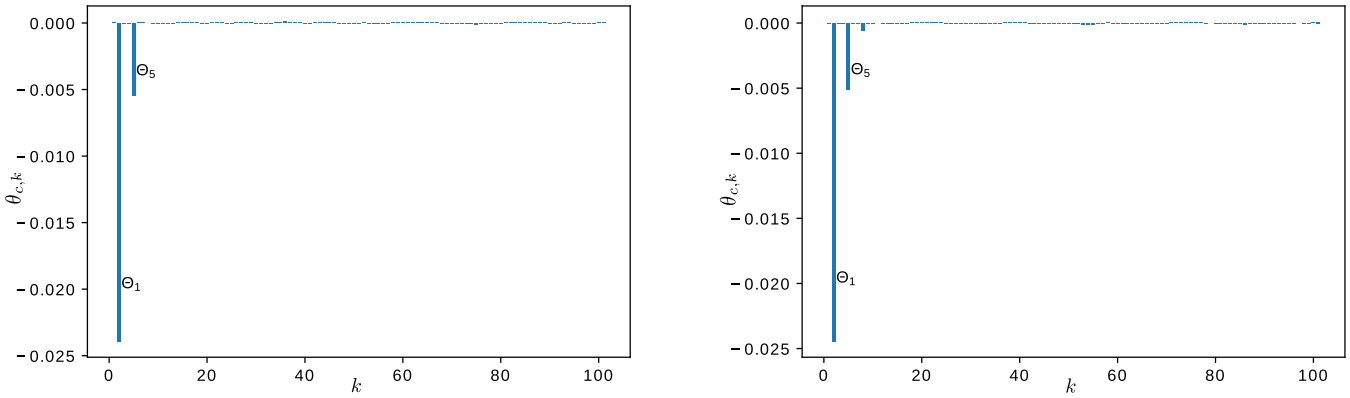


Fig. 25. Posterior means of the inferred $\boldsymbol{\theta}_c$ that parametrize the CG evolution law (Equation (41)) for $n = 16$ (left) and $n = 64$ (right) training data.

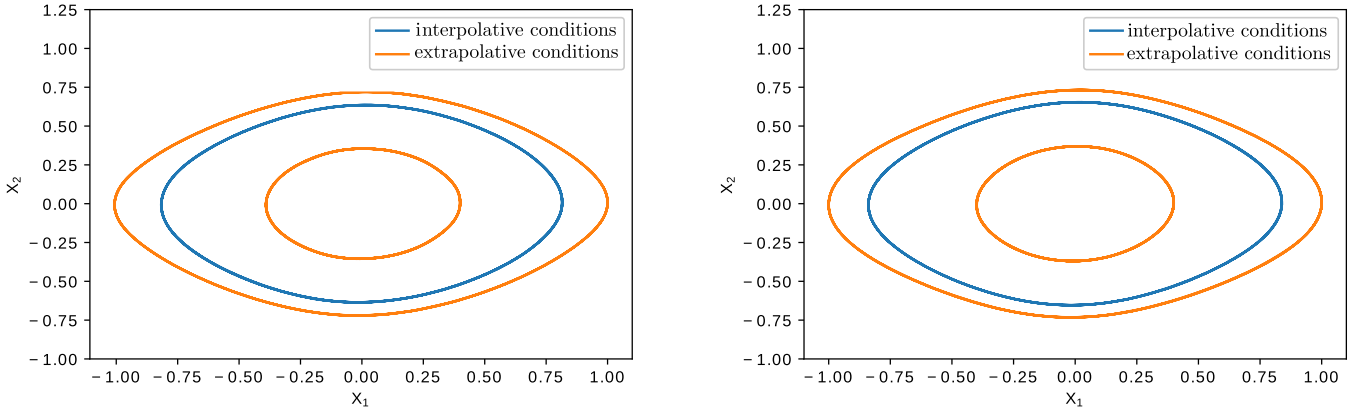


Fig. 26. Comparison of trajectories in state space $\boldsymbol{X}$ of the CG dynamics learned for $n = 16$ (left) and $n = 64$ (right) training data.

evolution in time of both coarse-grained variables is shown. The periodic nature of the CG dynamics is obvious, even though the CG state variables implicitly identified do not correspond to the natural ones i.e. y_t and $\dot{y}_t$.

This can be seen in Fig. 28 where for data-sequences $\boldsymbol{x}^{(i)}_{0:T\,\Delta t}$ (corresponding to the pendulum at various positions i.e. angles $y_{0:T\,\Delta t}$), we compute from the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(i)}_{0:T\,\Delta t}|\boldsymbol{x}^{(i)}_{0:T\,\Delta t})$ (Equation (44)) the mean of the corresponding CG states $\boldsymbol{X}^{(i)}_{0:T\ \Delta t}$ as well as the (in this case negligible) standard deviation. For each time instant $l = 0, 1, \ldots, T$, we plot the pairs of $y_{l\Delta t}$ and (the mean of) $X_{l\Delta t,1}$ (i.e. the first of the CG variables identified) to show the relation between the two variables. While it is obvious from the scales that the first CG variable identified is *not* the angle, it appears to be isomorphic to y. The latter property persists for $n = 64$ even though the sign of the relation has been reversed. The difference between the first CG variable identified and the natural angle y explains the difference between the CG evolution law identified (Fig. 25) and the reference one Equation (38).

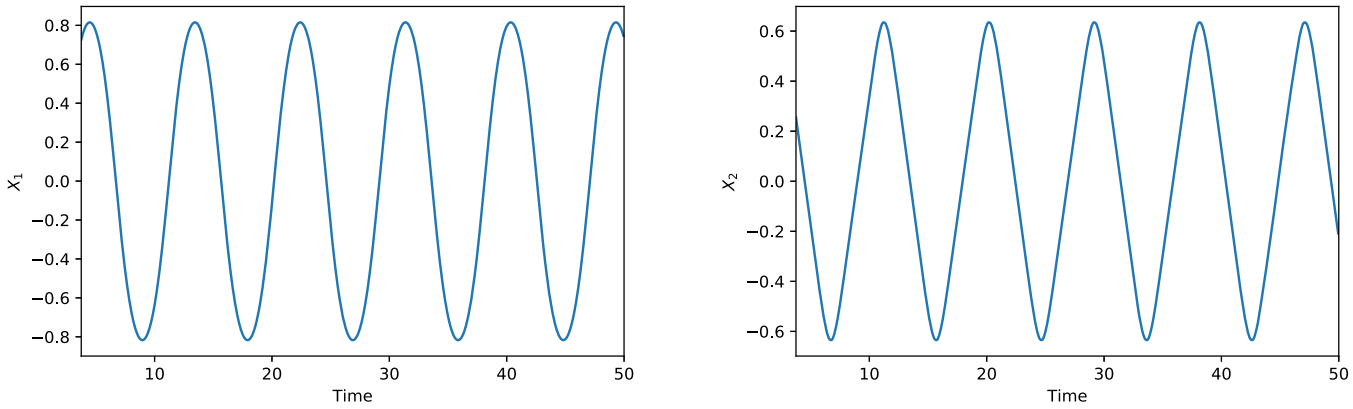


Fig. 27. Predicted posterior mean of CG state variables $\boldsymbol{X}_t$.

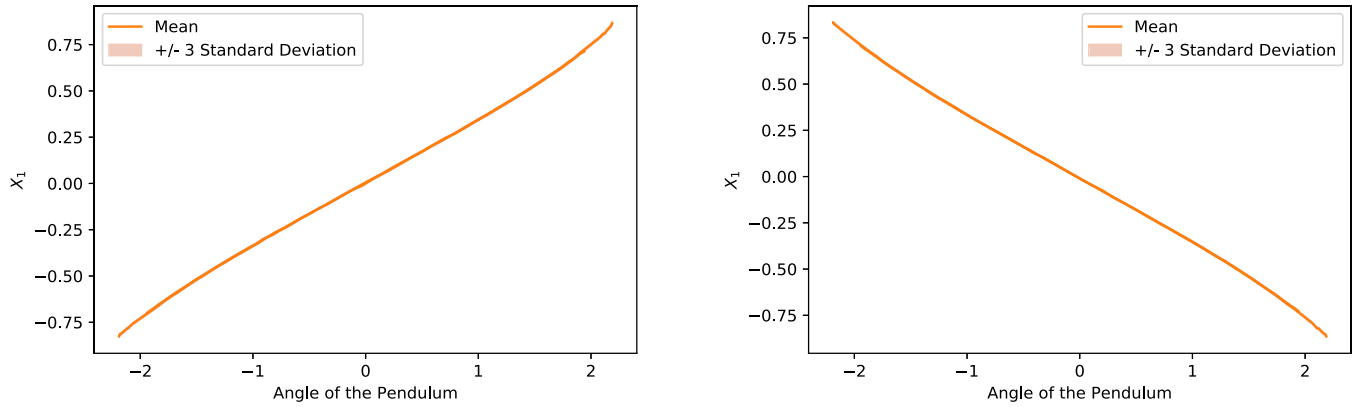


Fig. 28. Mapping between the angle of the pendulum and the coarse-grained coordinates for 32 training data and 64 (right) training data.

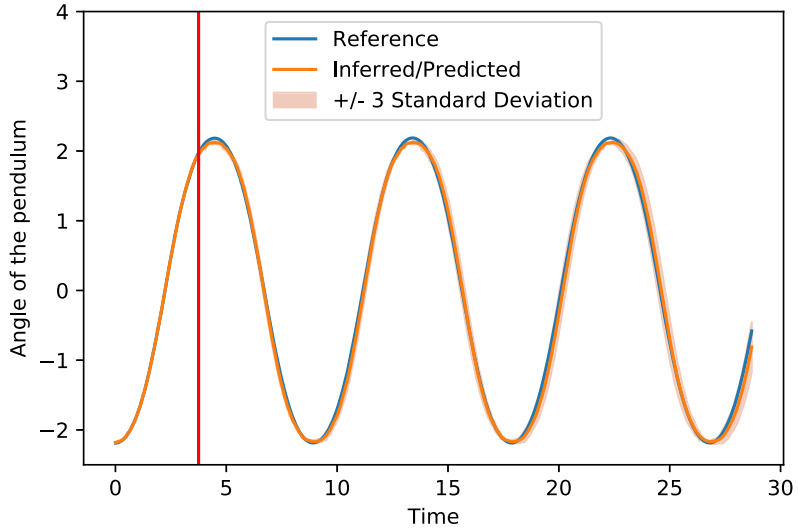


Fig. 29. Inferred/predicted evolution of the center of mass of the pendulum. The vertical line separates the inferred states from the predictions.

Fig. 29 provides predictive estimates of the position of the center of mass in time. These were obtained by propagating the CG variables in time and for each time instant, sampling p_{cf} for corresponding images $\boldsymbol{x}$. From the latter, the center of mass was computed from the activated pixels i.e. the pixels with value 1. Naturally, predictive uncertainty arises due the stochasticity in the initial conditions of $\boldsymbol{X}$ as well as in p_{cf}. The latter is quantified by the standard deviation and plotted in Fig. 29. As in the previous examples, the predictive uncertainty grows, albeit modestly, with time.

Fig. 30 depicts predictions in time for two pixels in the image. One can clearly distinguish the change-points i.e. when the pendulum crosses the pixel and its value is changed from 0 to 1 as well as the predictive uncertainty which is concentrated

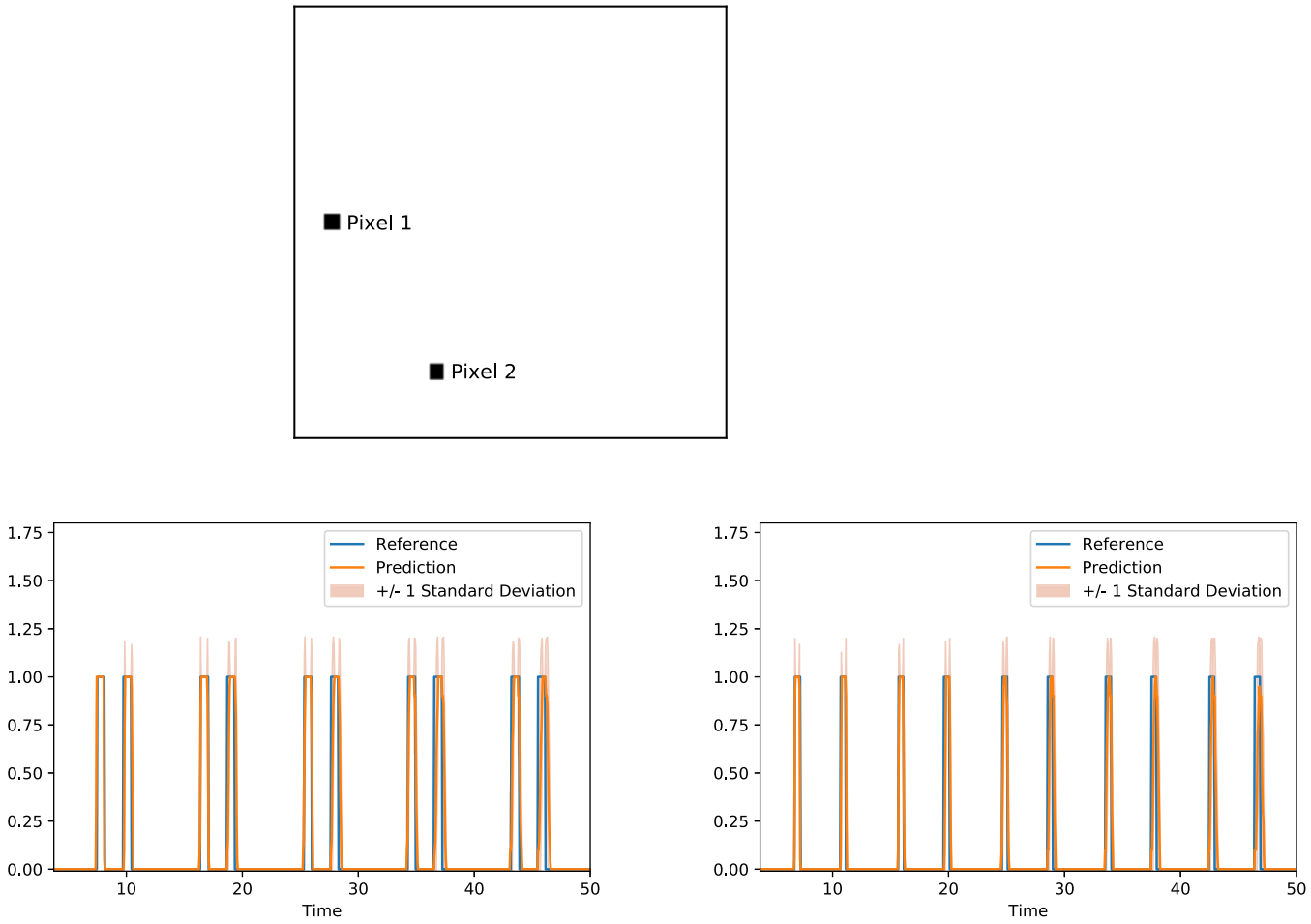


Fig. 30. Predicted time history of a single pixel: Pixel 1 (left) and Pixel 2 (right).

at those change-points. This demonstrates one of the strengths of our approach as due to the coarse-to-fine mapping the whole FG state is reconstructed and every observable can be computed together with the associated predictive uncertainty.

Finally, Fig. 31 compares actual images obtained by the reference dynamics of the pendulum with the predictive posterior mean obtained by the CG model and p_{cf} trained on the data. Even though these extend up to 875 time-steps i.e. more than 11 times longer than the time-window over which observations were available, they match the reference quite accurately, a strong indication that the right CG variables and CG dynamics have been identified. An animation containing all frames can be found by following this link.

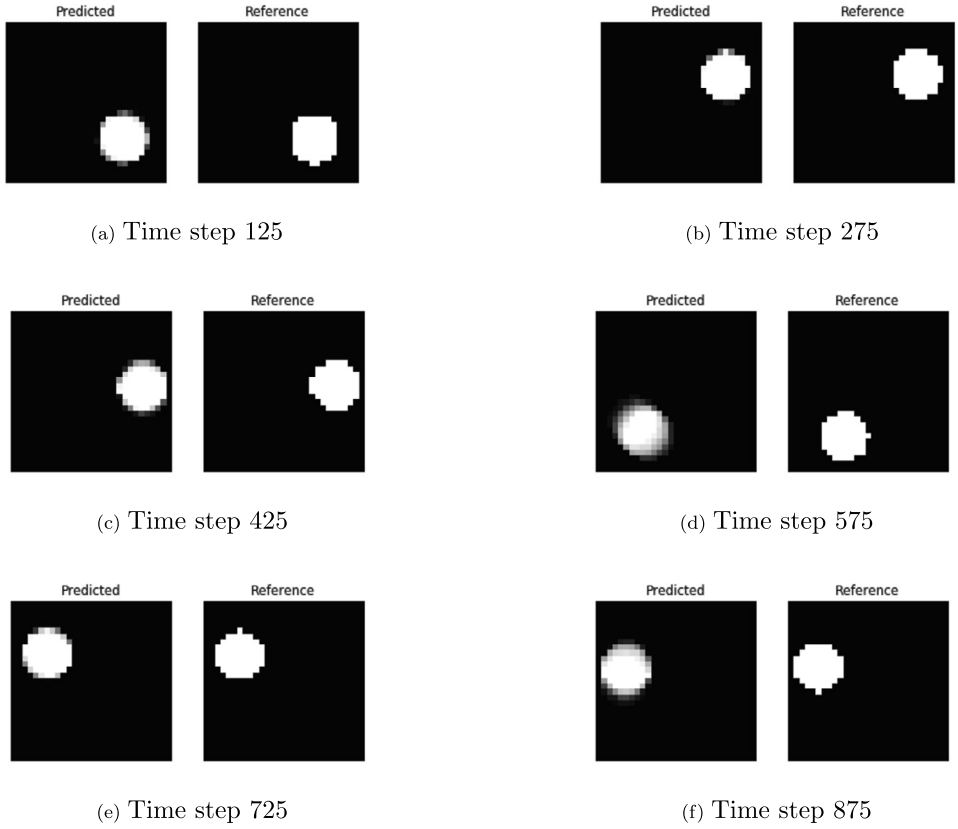
3.2.6. Missing data

The generative nature of the proposed model makes it highly suitable for handling missing FG data either in the form of partial observations of the FG state vector $\boldsymbol{x}_t$ or observations over a portion/subset of the time-sequence considered. We investigate the latter case in this section but note that in both situations the only modification required is removing the likelihood terms corresponding to the missing data from Equation (13).

In particular, we investigated the performance of the model when every second FG state $\boldsymbol{x}_t$ in the training sequences was *not* observed i.e. the FG observables consisted of $\{\boldsymbol{x}_0^{(i)}, \boldsymbol{x}_{2\Delta t}^{(i)}, \boldsymbol{x}_{4\Delta t}^{(i)}, \ldots, \boldsymbol{x}_{T\Delta t}^{(i)}\}$ for each data sequence i (where $T = 74$ as before). As one would expect, fewer observations lead to higher inferential uncertainties as seen when comparing Fig. 28 (fully observed case) with Fig. 32 (partially observed case). More importantly, fewer observations lead to higher predictive uncertainty as seen when comparing the predictions for the center of pendulum in Fig. 29 (fully observed case) with Fig. 33 (partially observed case).

4. Conclusions

We proposed a probabilistic generative model for the automated discovery of coarse-grained variables and dynamics based on fine-grained simulation data. The FG simulation data are augmented in a fully Bayesian fashion by virtual observables that enable the incorporation of physical constraints at the CG level that appear in the form of equalities. These could be residuals of the CG evolution law or more importantly conservation laws that are available when CG variables have physical meaning. This is particularly important in the context of physical modeling as in many cases such domain



(a) Time step 125 (b) Time step 275

(c) Time step 425 (d) Time step 575

(e) Time step 725 (f) Time step 875

Fig. 31. Predictive posterior means of images of the pendulum compared to the reference data.

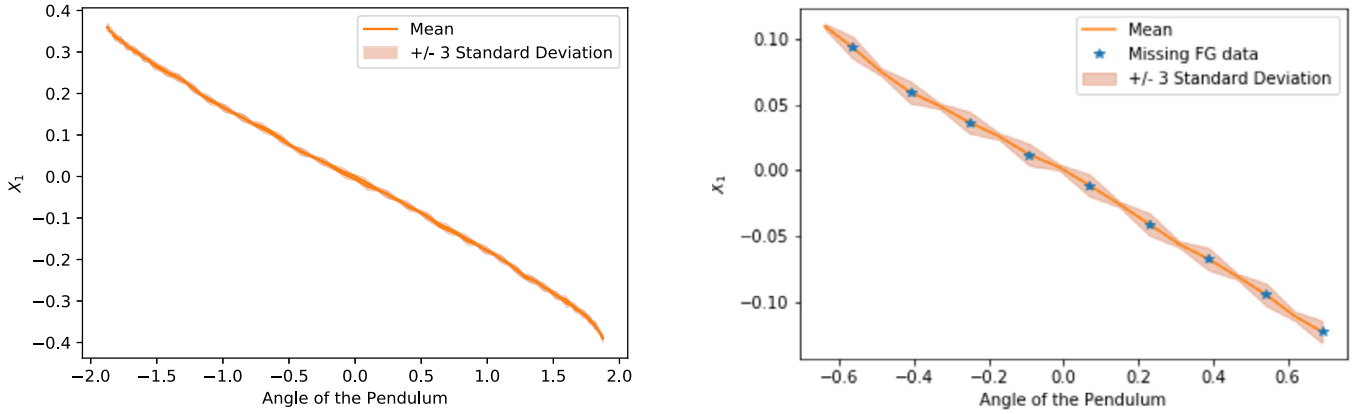


Fig. 32. Effect of missing data on the CG variables. The figure on the right is zoomed-in to show the higher uncertainty associated with CG states with missing data.

knowledge is a priori available and its inclusion can, not only reduce the amount of training data, but endow the CG model learned with the necessary features that would allow it to provide accurate predictions in out-of-distribution settings. Our approach learns simultaneously a coarse-to-fine mapping and an evolution law for the coarse-grained dynamics by employing probabilistic inference tools for the latent variables and model parameters. The use of deep neural nets for the former component can endow great expressiveness and flexibility. The concept of sparsity, which is invoked in learning CG dynamics from a large vocabulary of right-hand-side terms, is readily incorporated using sparsity-inducing Bayesian priors without any hyperparameter tuning. Furthermore, appropriate priors can promote the discovery of slow-varying CG variables which better capture the macroscopic features of the system. As a result of the aforementioned characteristics, the framework can learn from *Small Data* (i.e. shorter and fewer FG time-sequences) which is a crucial advantage in multiscale models where the simulation of the FG dynamics is expensive and slow in exploring the state-space. The model proposed was successfully tested on coarse-graining tasks from different areas. In all three examples, the method performed well under interpolative, and more importantly under extrapolative settings i.e. in cases where initial conditions different from the ones seen during training, are prescribed. Partial or incomplete FG observations can readily be handled due to its generative nature. Moreover,

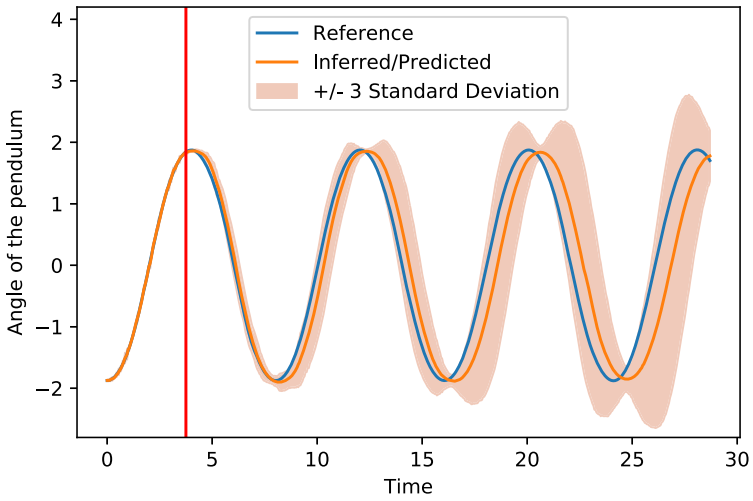


Fig. 33. Inferred/predicted evolution of the center of mass of the pendulum for the missing data case. The vertical line separates the inferred states from the predictions.

as it is able to reconstruct the entire FG state vector at any future time instant, it is capable of producing predictions of any FG observable of interest as well as quantify the associated predictive uncertainty.

There exist various possibilities to extend the proposed framework, both methodologically as well as in terms of applications. In the latter case and apart from using it for predictive purposes, the CG model learned could also be employed in optimization and control applications. On the methodological front an obvious extension would be to account for the virtual observables at future time-instants as well. This would ensure their enforcement by future CG states but would unavoidably complicate their simulation as a probabilistic inference scheme would need to be employed in order to draw samples.

Another important question pertains to the stability of the CG dynamics identified [68]. This is not currently guaranteed in the discretized nor in the continuous version. This could potentially be achieved by an a-priori parametrization of the CG dynamics in a way that guarantees stability which could in turn reduce the expressivity of the model. Finally, we note that, in our opinion, the most difficult question in coarse-graining multiscale systems, is finding the number of CG state variables that are needed. In physics problems, very often one has an idea of which variables would be suitable either based on the analysis-objectives and/or physical insight. Almost never though does one have a guarantee that these variables are sufficient. Assuming they are, the problem then reduces to finding the appropriate closures (i.e. right-hand sides in the CG dynamics) which is the problem we try to address in this paper. The discovery of additional, potentially non-physical CG state variables, would require additional advances for which we believe the ELBO, i.e. the (approximate) model evidence, could serve as the guiding objective.

Declaration of competing interest

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

References

[1] D. Givon, R. Kupferman, A. Stuart, Extracting macroscopic dynamics: model problems and algorithms, Nonlinearity (2004).
[2] W. Bialek, Biophysics: Searching for Principles, Princeton University Press, 2012.
[3] M. Alber, A. Buganza Tepole, W.R. Cannon, S. De, S. Dura-Bernal, K. Garikipati, G. Karniadakis, W.W. Lytton, P. Perdikaris, L. Petzold, E. Kuhl, Integrating machine learning and multiscale modeling-perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences, NPJ Digit. Med. 2 (2019) 115, https://doi.org/10.1038/s41746-019-0193-y.
[4] Z. Ghahramani, Probabilistic machine learning and artificial intelligence, Nature 521 (2015) 452–459, https://doi.org/10.1038/nature14541, http://www.nature.com/nature/journal/v521/n7553/full/nature14541.html.
[5] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[6] P.-S. Koutsourelakis, N. Zabaras, M. Girolami, Big data and predictive computational modeling, J. Comput. Phys. 321 (2016) 1252–1254.
[7] G. Marcus, E. Davis, Rebooting AI: Building Artificial Intelligence We Can Trust, Pantheon, 2019.
[8] P. Stinis, T. Hagge, A.M. Tartakovsky, E. Yeung, Enforcing constraints for interpolation and extrapolation in generative adversarial networks, J. Comput. Phys. 397 (2019) 108844.
[9] P.-S. Koutsourelakis, E. Bilionis, Scalable Bayesian reduced-order models for simulating high-dimensional multiscale dynamical systems, Multiscale Model. Simul. 9 (2011) 449–485, https://doi.org/10.1137/100783790.
[10] I. Kevrekidis, C. Gear, J. Hyman, P. Kevrekidis, O. Runborg, K. Theodoropoulos, Equation-free multiscale computation: enabling microscopic simulators to perform system-level tasks, Commun. Math. Sci. 1 (2003) 715–762.
[11] P.J. Schmid, Dynamic mode decomposition of numerical and experimental data, J. Fluid Mech. 656 (2010) 5–28, https://doi.org/10.1017/S0022112010001217, https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/dynamic-mode-decomposition-of-numerical-and-experimental-data/AA4C763B525515AD4521A6CC5E10DBD4.

[12] M.O. Williams, I.G. Kevrekidis, C.W. Rowley, A data–driven approximation of the Koopman operator: extending dynamic mode decomposition, J. Nonlinear Sci. 25 (2015) 1307–1346, https://doi.org/10.1007/s00332-015-9258-5.
[13] H. Wu, F. Noé, Variational approach for learning Markov processes from time series data, 2017, arXiv:1707.04659 [math, stat].
[14] G. Froyland, G.A. Gottwald, A. Hammerlindl, A computational method to extract macroscopic variables and their dynamics in multiscale systems, SIAM J. Appl. Dyn. Syst. 13 (2014) 1816–1846, https://doi.org/10.1137/130943637, https://epubs.siam.org/doi/abs/10.1137/130943637.
[15] L. Felsberger, P. Koutsourelakis, Physics-constrained, data-driven discovery of coarse-grained dynamics, Commun. Comput. Phys. 25 (2019) 1259–1301, https://doi.org/10.4208/cicp.OA-2018-0174.
[16] M. Schöberl, N. Zabaras, P.-S. Koutsourelakis, Predictive coarse-graining, J. Comput. Phys. 333 (2017) 49–77.
[17] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations, arXiv preprint, arXiv:1711.10561, 2017.
[18] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[19] Y. Yang, P. Perdikaris, Conditional deep surrogate models for stochastic, high-dimensional, and multi-fidelity systems, Comput. Mech. (2019) 1–18.
[20] A. Mardt, L. Pasquali, H. Wu, F. Noé, VAMPnets for deep learning of molecular kinetics, Nat. Commun. 9 (2018), https://doi.org/10.1038/s41467-017-02388-1, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5750224/.
[21] H. Wu, A. Mardt, L. Pasquali, F. Noe, Deep generative Markov state models, in: Advances in Neural Information Processing Systems, 2018, pp. 3975–3984.
[22] L. Duncker, G. Bohner, J. Boussard, M. Sahani, Learning interpretable continuous-time models of latent stochastic dynamical systems, arXiv preprint, arXiv:1902.04420, 2019.
[23] C. Grigo, P.-S. Koutsourelakis, A physics-aware, probabilistic machine learning framework for coarse-graining high-dimensional systems in the small data regime, arXiv preprint, arXiv:1902.03968, 2019.
[24] Y. Pantazis, I. Tsamardinos, A unified approach for sparse dynamical system inference from temporal measurements, Bioinformatics 35 (2019) 3387–3396, https://doi.org/10.1093/bioinformatics/btz065, https://academic.oup.com/bioinformatics/article/35/18/3387/5305020.
[25] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (2016) 3932–3937.
[26] E. Kaiser, J.N. Kutz, S.L. Brunton, Sparse identification of nonlinear dynamics for model predictive control in the low-data limit, Proc. R. Soc. A 474 (2018) 20180335.
[27] K. Champion, B. Lusch, J.N. Kutz, S.L. Brunton, Data-driven discovery of coordinates and governing equations, arXiv preprint, arXiv:1904.02107, 2019.
[28] J. Ohkubo, Nonparametric model reconstruction for stochastic differential equations from discretely observed time-series data, Phys. Rev. E 84 (2011) 066702, https://doi.org/10.1103/PhysRevE.84.066702, https://link.aps.org/doi/10.1103/PhysRevE.84.066702.
[29] S. Klus, F. Nüske, P. Koltai, H. Wu, I. Kevrekidis, C. Schütte, F. Noé, Data-driven model reduction and transfer operator approximation, J. Nonlinear Sci. 28 (2018) 985–1010, https://doi.org/10.1007/s00332-017-9437-7.
[30] B.O. Koopman, Hamiltonian systems and transformations in Hilbert space, Proc. Natl. Acad. Sci. USA 17 (1931) 315–318, https://www.jstor.org/stable/86114.
[31] I. Mezić, Spectral properties of dynamical systems, model reduction and decompositions, Nonlinear Dyn. 41 (2005) 309–325.
[32] S.L. Brunton, B.W. Brunton, J.L. Proctor, J.N. Kutz, Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control, PLoS ONE 11 (2016) e0150171.
[33] M.A. Katsoulakis, P. Plecháč, Information-theoretic tools for parametrized coarse-graining of non-equilibrium extended systems, J. Chem. Phys. 139 (2013) 074115.
[34] V. Harmandaris, E. Kalligiannaki, M. Katsoulakis, P. Plecháč, Path-space variational inference for non-equilibrium coarse-grained systems, J. Comput. Phys. 314 (2016) 355–383, https://doi.org/10.1016/j.jcp.2016.03.021, http://www.sciencedirect.com/science/article/pii/S002199911600173X.
[35] M.A. Katsoulakis, P. Vilanova, Data-driven, variational model reduction of high-dimensional reaction networks, J. Comput. Phys. (2019) 108997.
[36] H. Mori, Transport, collective motion, and Brownian motion, Prog. Theor. Phys. 33 (1965) 423–455.
[37] R. Zwanzig, Nonlinear generalized Langevin equations, J. Stat. Phys. 9 (1973) 215–220.
[38] A. Chorin, P. Stinis, Problem reduction, renormalization, and memory, Commun. Appl. Math. Comput. Sci. 1 (2007) 1–27.
[39] H. Lei, N.A. Baker, X. Li, Data-driven parameterization of the generalized Langevin equation, Proc. Natl. Acad. Sci. 113 (2016) 14183–14188.
[40] Y. Zhu, J.M. Dominy, D. Venturi, On the estimation of the Mori-Zwanzig memory integral, J. Math. Phys. 59 (2018) 103501, https://doi.org/10.1063/1.5003467, https://aip.scitation.org/doi/10.1063/1.5003467.
[41] M.D. Hoffman, D.M. Blei, C. Wang, J. Paisley, Stochastic variational inference, J. Mach. Learn. Res. 14 (2013) 1303–1347.
[42] C. Archambeau, M. Opper, Approximate Inference for Continuous-Time Markov Processes, Bayesian Time Series Models, 2011, pp. 125–140.
[43] R.G. Krishnan, U. Shalit, D. Sontag, Structured inference networks for nonlinear state space models, in: Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[44] V. Fortuin, G. Rätsch, S. Mandt, Multivariate time series imputation with variational autoencoders, arXiv preprint, arXiv:1907.04155, 2019.
[45] J.C. Butcher, Numerical Methods for Ordinary Differential Equations, John Wiley & Sons, 2016.
[46] R. Coifman, I. Kevrekidis, S. Lafon, M. Maggioni, B. Nadler, Diffusion maps, reduction coordinates and low dimensional representation of stochastic systems, Multiscale Model. Simul. 7 (2008) 842–864.
[47] J. Trashorras, D. Tsagkarogiannis, From mesoscale back to microscale: reconstruction schemes for coarse-grained stochastic lattice systems, SIAM J. Numer. Anal. 48 (2010) 1647–1677, https://doi.org/10.1137/080722382, http://epubs.siam.org/doi/abs/10.1137/080722382.
[48] M.A. Katsoulakis, J. Trashorras, Information loss in coarse-graining of stochastic particle dynamics, J. Stat. Phys. 122 (2006) 115–135, http://link.springer.com/article/10.1007/s10955-005-8063-1.
[49] D. Kondrashov, M.D. Chekroun, M. Ghil, Data-driven non-Markovian closure models, Phys. D: Nonlinear Phenom. 297 (2015) 33–55, https://doi.org/10.1016/j.physd.2014.12.005, http://www.sciencedirect.com/science/article/pii/S0167278914002413.
[50] B.D. Coleman, M.E. Gurtin, Thermodynamics with internal state variables, J. Chem. Phys. 47 (1967) 597–613, https://doi.org/10.1063/1.1711937, http://scitation.aip.org/content/aip/journal/jcp/47/2/10.1063/1.1711937.
[51] O. Cappe, E. Moulines, T. Ryden, Inference in Hidden Markov Models, Springer-Verlag, 2005.
[52] Z. Ghahramani, Unsupervised learning, in: O. Bousquet, G. Raetsch, U. von Luxburg (Eds.), Advanced Lectures on Machine Learning, in: LNAI, vol. 3176, Springer-Verlag, 2004.
[53] D. Durstewitz, A state space approach for piecewise-linear recurrent neural networks for identifying computational dynamics from neural measurements, PLoS Comput. Biol. 13 (2017) e1005542, https://doi.org/10.1371/journal.pcbi.1005542, http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005542.
[54] L. Wiskott, T.J. Sejnowski, Slow feature analysis: unsupervised learning of invariances, Neural Comput. 14 (2002) 715–770, https://doi.org/10.1162/089976602317318938.
[55] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[56] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in: The International Conference on Learning Representations (ICLR), Banff, Alberta, Canada, 2014, http://arxiv.org/abs/1312.6114.

[57] Y. Kim, S. Wiseman, A.C. Miller, D. Sontag, A.M. Rush, Semi-amortized variational autoencoders, arXiv preprint, arXiv:1802.02550, 2018.
[58] C. Grigo, P.-S. Koutsourelakis, Bayesian model and dimension reduction for uncertainty propagation: applications in random media, SIAM/ASA J. Uncertain. Quantificat. 7 (2019) 292–323.
[59] J. Li, P.G. Kevrekidis, C.W. Gear, I.G. Kevrekidis, Deciding the nature of the coarse equation through microscopic simulations: the baby-bathwater scheme, SIAM Rev. 49 (2007) 469–487, https://doi.org/10.1137/070692303.
[60] W.G. Noid, Perspective: coarse-grained models for biomolecular systems, J. Chem. Phys. 139 (2013), https://doi.org/10.1063/1.4818908, http://scitation.aip.org/content/aip/journal/jcp/139/9/10.1063/1.4818908.
[61] D. Mackay, Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks, Netw. Comput. Neural Syst. 6 (1995) 469–505, https://doi.org/10.1088/0954-898X/6/3/011.
[62] C.M. Bishop, M.E. Tipping, Variational relevance vector machines, in: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 2000, pp. 46–53.
[63] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.
[64] G.-H. Cottet, P.D. Koumoutsakos, Vortex Methods: Theory and Practice, 2 edition, Cambridge University Press, Cambridge, New York, 2000.
[65] S. Roberts, Convergence of a random walk method for the Burgers equation, Math. Comput. 52 (1989) 647–673, https://doi.org/10.2307/2008486, http://www.jstor.org/stable/2008486.
[66] A. Chertock, D. Levy, Particle methods for dispersive equations, J. Comput. Phys. 171 (2001) 708–730, https://doi.org/10.1006/jcph.2001.6803, http://www.sciencedirect.com/science/article/pii/S0021999101968032.
[67] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: large-scale machine learning on heterogeneous distributed systems, arXiv preprint, arXiv:1603.04467, 2016.
[68] S. Pan, K. Duraisamy, Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability, SIAM J. Appl. Dyn. Syst. 19 (2020) 480–509.

B. Paper B

[1] https://creativecommons.org/licenses/by-nc-sa/3.0/deed.en_US

14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020
Virtual Congress: 11-–15 January 2021
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

PHYSICS-AWARE, DEEP PROBABILISTIC MODELING OF MULTISCALE DYNAMICS IN THE SMALL DATA REGIME

Sebastian Kaltenbach[1], **Phaedon-Stelios Koutsourelakis**[1]

[1] Professorship of Continuum Mechanics, Technical University of Munich
Boltzmannstr.15, 85748 Garching
{sebastian.kaltenbach; p.s.koutsourelakis}@tum.de, www.mw.tum.de/contmech

Key words: Bayesian machine learning, virtual observables, multiscale modeling, coarse-graining

Abstract: *The data-based discovery of effective, coarse-grained (CG) models of high-dimensional dynamical systems presents a unique challenge in computational physics and particularly in the context of multiscale problems. The present paper offers a probabilistic perspective that simultaneously identifies predictive, lower-dimensional coarse-grained (CG) variables as well as their dynamics. We make use of the expressive ability of deep neural networks in order to represent the right-hand side of the CG evolution law. Furthermore, we demonstrate how domain knowledge that is very often available in the form of physical constraints (e.g. conservation laws) can be incorporated with the novel concept of virtual observables. Such constraints, apart from leading to physically realistic predictions, can significantly reduce the requisite amount of training data which enables reducing the amount of required, computationally expensive multiscale simulations (Small Data regime). The proposed state-space model is trained using probabilistic inference tools and, in contrast to several other techniques, does not require the prescription of a fine-to-coarse (restriction) projection nor time-derivatives of the state variables. The formulation adopted is capable of quantifying the predictive uncertainty as well as of reconstructing the evolution of the full, fine-scale system which allows to select the quantities of interest a posteriori. We demonstrate the efficacy of the proposed framework in a high-dimensional system of moving particles.*

1 INTRODUCTION

The solution of high-dimensional, multiscale system is challenging as the required computational resources usually grow exponentially with the dimension of the state-space as well as with the smallest time-scale that needs to be resolved. As such systems are ubiqitious in applied physics and engineering, reduced/coarse-grained descriptions and models are necessary that are predictive of various observables or the high-dimensional system, but whose discretization time-scales can be much larger than the inherent ones [1].

We adopt a data-based perspective [2, 3] that relies on data generated by simulations of a fine-grained (FG) system in order to learn a coarse-grained (CG) model. We nevertheless note that such coarse-graining tasks exhibit fundamental differences from large-scale machine learn-

ing tasks [4, 5] as the data involved is usually small due to the expensive data acquisition and as information about the underlying physical structure of the problem is available. When this domain knowledge is incorporated into the CG model it can improve its predictive ability [6, 7].

In contrast to other frameworks for reduced-order modeling (e.g. SINDy [8]) where the dynamics of the CG model is learned based on a large vocabulary of feature functions, we employ a deep neural network for the CG dynamics in order to gain great flexibility and be able to not restrict ourselves to an a priori chosen set of feature functions. This approach is similar to the ideas of Neural ODEs [9] and Neural SDEs [10] which also use neural networks to represent the dynamics. Another possibility would be the use of Gaussian Processes [11] which would allow non-parametric, probabilistic modeling.

In this paper, we combine a generative, probabilistic machine learning framework [12] with virtual observables [6] and deep neural networks for the CG dynamics as well as the mapping from the CG states to the FG states. In doing so, we propose a framework that can make use of the flexibility of neural nets, while still obeying physical laws. We carry out the tasks of model estimation and dimensionality reduction simultaneously and identify the CG states variables, their dynamics as well as a probabilistic coarse-to-fine map based only on small amounts of FG simulation data.

2 METHODOLOGY

In general, the subscript f or lower-case letters are used to denote variables associated with the (high-dimensional) fine-grained(FG) model and the subscript c or upper-case letters are used for quantities of the (lower-dimensional) coarse-grained(CG) description. We also use a circumflex $\hat{}$ to denote observed/known variables.

2.1 The FG and CG model

The fine-grained system considered is a high-dimensional system with state variables $\boldsymbol{x}$ ($\boldsymbol{x} \in \mathscr{X}_f \subset \mathbf{R}^{d_f}$), whose dimension d_f is very large ($d_f >> 1$). We describe the dynamics of such a FG system by a system of deterministic or stochastic ODEs i.e.,

$$\dot{\boldsymbol{x}}_t = \boldsymbol{f}(\boldsymbol{x}_t, t), \quad t > 0 \tag{1}$$

The FG system is moreover considered to have the initial condition $\boldsymbol{x}_0$ that might be deterministic or drawn from a specified distribution. In this work, we want to coarse-grain such a system only based on simulated data, i.e. time sequences simulated from Equation (1) with a time-step δt.

Our goal is to simultaneously identify (unknown) CG state variables $\boldsymbol{X}$ with $\boldsymbol{X} \in \mathscr{X}_c \subset \mathbf{R}^{d_c}$ as well as the dynamics of those CG variables. The dimension d_c of these CG state variables is intended to be much smaller than d_f. For the CG dynamics a Markovian dynamic is assumed in the form:

$$\dot{\boldsymbol{X}}_t = \boldsymbol{F}(\boldsymbol{X}_t, t), \quad t > 0 \tag{2}$$

2.2 Emission law

In contrast to approaches based on the Mori-Zwanzig formalism [13, 14], which include a mapping from the FG system to the quantities of interest, we employ a probabilistic, generative coarse-to-fine map [15] from the CG state-variables to the FG description. We indicate the associated (conditional) density by:

$$p_{cf}(\boldsymbol{x}_t|\, \boldsymbol{X}_t;\, \boldsymbol{\theta}_{cf}) \tag{3}$$

where $\boldsymbol{\theta}_{cf}$ denote the (unknown) parameters that we will try to learn from the data. This conditional density p_{cf} can be endowed a priori with domain knowledge by adapting its form to the particulars of the problem or it can parametrized by deep neural networks to allow for maximum flexibility.

Employing a probabilistic coarse-to-fine map instead of a deterministic, restriction operator has many advantages as e.g. the full FG system's reconstruction and probabilistic predictive estimates.

2.3 Transition law

In the following, we consider discretized time with a fixed time-step Δt and time-related subscripts refer to the number of time-steps.

We model the CG dynamics with the help of a deep neural network in order to gain a great flexibility and be able to express nonlinear functions. Therefore, we assume an explicit discretization of Equation (2) and model the right-hand-side by the deep neural network $NN(.)$ parametrized by $\boldsymbol{\theta}_{NN}$:

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t + NN(\boldsymbol{X}_t, \boldsymbol{\theta}_{NN}) + \sigma_r \boldsymbol{\varepsilon}, \qquad \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{4}$$

where the parameter $\sigma_r \geq 0$ is responsible for the stochastic part of the CG dynamics. This leads to the following conditional density:

$$p(\boldsymbol{X}_{t+1}\,|\boldsymbol{X}_t, \boldsymbol{\theta}_{NN}, \sigma_r) = \mathcal{N}(\boldsymbol{X}_{t+1}\,|\,\boldsymbol{X}_t + NN(\boldsymbol{X}_t, \boldsymbol{\theta}_{NN}), \sigma_r^2 \boldsymbol{I}) \tag{5}$$

which effectively represents a discretized version of the neural stochastic ODEs of [10] and is more flexible as compared to approaches in which the right-hand side consists of a restricted amount of first- and second-order interactions of $\boldsymbol{X}_t$ [6].

2.4 Virtual observables

As the CG state-variables $\boldsymbol{X}$ employed in multiscale modeling are usually given physical meaning, we employ the concept of *virtual observables* [6] in order to incorporate general physical principles such as conservation of mass, momentum or energy. Let these be expressed as equalities of the form at each time-step l:

$$\boldsymbol{c}_l(\boldsymbol{X}_l) = \boldsymbol{0}, \quad l = 0, 1, \ldots \tag{6}$$

where $\boldsymbol{c}_l : \mathscr{X}_c \subset \mathbb{R}^{d_c} \to \mathbb{R}^{M_c}$. The only requirement we will impose is that of differentiability of $\boldsymbol{c}_l$ [6]. We define a new variable $\hat{\boldsymbol{c}}_l$ which relates to $\boldsymbol{c}_l$ as follows:

$$\hat{\boldsymbol{c}}_l = \boldsymbol{c}_l(\boldsymbol{X}_l) + \sigma_c \boldsymbol{\varepsilon}_c, \qquad \boldsymbol{\varepsilon}_c \sim \mathscr{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{7}$$

Now, it is assumed that the $\hat{\boldsymbol{c}}_l$ have been *virtually* observed and this set of virtual observations $\hat{\boldsymbol{c}}_l = 0$ leads to to an augmented version of the FG data and therefore virtual likelihoods of the type:

$$p(\hat{\boldsymbol{c}}_l = \boldsymbol{0} \mid \boldsymbol{X}_l, \sigma_R) = \mathscr{N}(\boldsymbol{0} \mid \boldsymbol{c}_l(\boldsymbol{X}_l), \sigma_c^2 \boldsymbol{I}) \tag{8}$$

The "noise" parameter σ_c can be used to account for the intensity of the enforcement of the virtual observations and represents the tolerance parameter with which the constraints would be enforced in a deterministic setting.

We note that the concept of virtual observables is not restricted to physical constraints but could also be applied to residuals of temporal discretization schemes [6] or of PDEs [16]. In both of this cases, it is shown that the incorporation of virtual observables can reduce the amount of training data required and enable training in the Small Data regime.

2.5 Inference and learning

Due to the introduction of virtual observables, we can adopt an enlarged definition of data which we cumulatively denote by $\mathscr{D} = \left\{ \hat{\boldsymbol{x}}_{0:T}^{(1:n)}, \hat{\boldsymbol{c}}_{0:T}^{(1:n)} \right\}$ and which encompasses:

- FG simulation data consisting of n sequences of the FG state-variables. These are denoted by $\hat{\boldsymbol{x}}_{0:T}^{(1:n)}$ as the likelihood model implied by the p_{cf} in Equation (3) involves only the observables at each coarse time-step.
- Virtual observables $\hat{\boldsymbol{c}}_l^{(1:n)}$ relating to the CG states $\boldsymbol{X}_l$ at each time-step l and which relate to the physical constraints as in Equation (7). In the example they pertain to all time-steps from 0 to T and are denoted by $\hat{\boldsymbol{c}}_{0:T}^{(1:n)}$.

We represent the latent (unobserved) variables of the model by the CG state-variables $\boldsymbol{X}_{0:T}^{(1:n)}$ and relate them to the FG data through the p_{cf} (in Equation (3)) and to the virtual observables through Equation (8). The parameters of the model are denoted cumulatively by $\boldsymbol{\theta}$ and consist of[1]:

- $\boldsymbol{\theta}_{NN}$ which parametrize the neural network for the right-hand-side of the CG evolution law (see section 2.3),
- $\boldsymbol{\theta}_{cf}$ which parametrize the probabilistic coarse-to-fine map (Equation (3)),
- σ_r involved in the stochasticity of the transition law Equation (4) and
- σ_c involved in the enforcement of virtual observables in Equation (7)

[1] If any of these parameters are prescribed, then they are omitted from $\boldsymbol{\theta}$.

We follow a fully-Bayesian formulation and express the posterior of the unknowns (i.e. latent variables and parameters) as follows:

$$p(\boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta} \mid \mathscr{D}) = \frac{p(\mathscr{D} \mid \boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{\theta})\ p(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{\theta})}{p(\mathscr{D})} \tag{9}$$

where $p(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{\theta})$ denotes the prior on the latent variables and parameters. The likelihood term $p(\mathscr{D}|\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{\theta})$ involved can be decomposed into the product of two (conditionally) independent terms, one for the FG data and one for the virtual observables, i.e.:

$$p(\mathscr{D} \mid \boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{\theta}) = p(\hat{\boldsymbol{x}}_{0:T}^{(1:n)} \mid \boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{\theta})\ p(\hat{\boldsymbol{c}}_{0:T}^{(1:n)} \mid \boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{\theta}) \tag{10}$$

We further note that (from Equation (3)):

$$p(\hat{\boldsymbol{x}}_{0:T}^{(1:n)} \mid \boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{\theta}) = \prod_{i=1}^{n}\prod_{t=0}^{T} p_{cf}(\boldsymbol{x}_t^{(i)} \mid \boldsymbol{X}_t^{(i)}, \boldsymbol{\theta}_{cf}) \tag{11}$$

and (from Equation (8)):

$$\begin{aligned} p(\hat{\boldsymbol{c}}_{0:T}^{(1:n)}|\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{\theta}) &= \textstyle\prod_{i=1}^{n}\prod_{l=0}^{T} \mathscr{N}(\boldsymbol{0}|\boldsymbol{c}_l(\boldsymbol{X}_l^{(i)}), \sigma_c^2 \boldsymbol{I}) \\ &\propto \textstyle\prod_{i=1}^{n}\prod_{l=0}^{T} \frac{1}{\sigma_c^{dim(\boldsymbol{c})}} \exp\left\{ -\frac{1}{2\sigma_c^2} \left| \boldsymbol{c}_l(\boldsymbol{X}_l^{(i)}) \right|^2 \right\} \end{aligned} \tag{12}$$

The prior $p(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{\theta})$ can be decomposed into the transition density of Equation (5) and a prior for $\boldsymbol{X}_0$ as well as the parameters $\boldsymbol{\theta}$:

$$p(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(\boldsymbol{X}_0^{(i)}) \prod_{t=0}^{T-1} p(\boldsymbol{X}_{t+1}^{(i)} \mid \boldsymbol{X}_t^{(i)}, \boldsymbol{\theta}_{NN}, \sigma_r)\ \ p(\boldsymbol{\theta}) \tag{13}$$

We advocate the use of Stochastic Variational Inference [17] for computing an approximate posterior. We select a parameterized family of densities, $q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta})$ and attempt to find the one that best approximates the posterior by minimizing their Kullback-Leibler divergence. It can be shown [18], that this optimal $q_{\boldsymbol{\phi}}$ maximizes the Evidence Lower Bound (ELBO) $\mathscr{F}(q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta}))$:

$$\begin{aligned} \log p(\mathscr{D}) &= \log \textstyle\int p(\mathscr{D},\ \boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta})\ d\boldsymbol{X}_{0:T}^{(1:n)}\ d\boldsymbol{\theta} \\ &= \log \textstyle\int \frac{p(\mathscr{D}|\ \boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta}) p(\boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta})} q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta})\ d\boldsymbol{X}_{0:T}^{(1:n)}\ d\boldsymbol{\theta} \\ &\geq \textstyle\int \log \frac{p(\mathscr{D}|\ \boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta}) p(\boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta})} q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta})\ d\boldsymbol{X}_{0:T}^{(1:n)}\ d\boldsymbol{\theta} \\ &= \mathscr{F}(q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta})) \end{aligned} \tag{14}$$

In the following illustrations, we postulate a *mean-field* decomposition:

$$q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)},\ \boldsymbol{\theta}) = q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)})\ p_{\boldsymbol{\phi}}(\boldsymbol{\theta}) = \left[\prod_{i=1}^{n} q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(i)})\right]\ \delta_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \tag{15}$$

where we make use of the (conditional) independence of the time sequences in the likelihood. We further note that we employed Dirac $\delta_{\boldsymbol{\phi}}$ functions for the $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ and therefore obtain MAP estimates $\boldsymbol{\theta}_{MAP}$ (i.e. $\boldsymbol{\phi}$ includes $\boldsymbol{\theta}_{MAP}$) for the unknown parameters.

Gradients of the ELBO with respect to the parameters $\boldsymbol{\phi}$ involve expectations with respect to $q_{\boldsymbol{\phi}}$. These were approximated with Monte Carlo estimates which employ the reparametrization trick [19] and stochastic optimization was carried out with the ADAM algorithm [20].

2.6 Predictions

The proposed framework can produce probabilistic predictive estimates for a sequence which was observed up to time-step T i.e. $\hat{\boldsymbol{x}}_{0:T}^{(i)}$. This predictive uncertainty reflects not only the information-loss due to the coarse-graining process but also the epistemic uncertainty arising from finite (and small) datasets.

In particular, if $q_{\boldsymbol{\phi}}(\boldsymbol{X}_T^{(i)})$ is the (marginal) posterior of the last, hidden CG state and $\boldsymbol{\theta}_{MAP}$ the MAP estimate of the model parameters, then we follow the steps described in Algorithm 1. This procedure generates samples of the full FG state evolution but does not necessarily guarantees the enforcement of the constraints for the CG states.

We note that if we would also like to enforce the constraints $\boldsymbol{c}_l$ for future predictions, then these would need to be included in the posterior density defined in Equation (9). Consequently, future (FG or CG) states would need to be inferred from this augmented posterior and an enlarged inference process is required for predictions.

Algorithm 1: Prediction - Algorithm

Result: Sample of $\boldsymbol{x}_{(T+P)}^{(i)}$
Data: $q_{\boldsymbol{\phi}}(\boldsymbol{X}_T), \boldsymbol{\theta}_{MAP}$

```
Sample from q_φ(X_T^(i));
while Time-step (T+P) not reached do
    Sample from the CG evolution law in Equation (4);
end
Sample from p_cf(x_(T+P) | X_(T+P), θ_MAP)
```

3 NUMERICAL ILLUSTRATIONS

We demonstrate the capabilities of the proposed framework by applying it to a high-dimensional system of stochastically moving particles.

3.1 FG model

For the simulations presented in this section, we used $d_f = 250 \times 10^3$ particles, which, at each microscopic time step $\delta t = 2.5 \times 10^{-3}$ performed random, non-interacting, jumps of size $\delta s = \frac{1}{640}$, either to the left with probability $p_{left} = 0.1875$ or to the right with probability $p_{right} = 0.2125$. The positions were restricted to a domain of $[-1,1]$ with periodic boundary conditions. It is well-known [21] that in the limit (i.e. $d_f \to \infty$) the particle density $\rho(s,t)$ can be described with an advection-diffusion PDE with diffusion constant $D = (p_{left} + p_{right})\frac{\delta s^2}{2\delta t}$ and velocity $v = (p_{right} - p_{left})\frac{\delta s}{\delta t}$:

$$\frac{\partial \rho}{\partial t} + v\frac{\partial \rho}{\partial s} = D\frac{\partial^2 \rho}{\partial s^2}, \qquad s \in (-1,1).. \tag{16}$$

3.2 CG model specifications

The CG model relates to a discretization of the particle density into $d_c = 25$ equally-sized bins at each coarse time step . The nature of the CG variables $\boldsymbol{X}_t$ gives rise to a multinomial for the coarse-to-fine density p_{cf} (section 2.2) i.e.:

$$p_{cf}(\boldsymbol{x}_t|\boldsymbol{X}_t) = \frac{d_f!}{m_1(\boldsymbol{x}_t)!\, m_2(\boldsymbol{x}_t)! \ldots m_{d_c}(\boldsymbol{x}_t)!} \prod_{j=1}^{d_c} X_{t,j}^{m_j(\boldsymbol{x}_t)}, \tag{17}$$

where $m_j(\boldsymbol{x}_t)$ is the number of particles in bin j. We assume that, given the CG state $\boldsymbol{X}_t$, the coordinates of the particles $\boldsymbol{x}_t$ are conditionally independent. This does not imply that they move independently nor that they cannot exhibit coherent behavior [22]. The consequence of Equation (17) is that for this example no parameters need to be learned for p_{cf}.

For the transition law (section 2.3), we assume a coarse time step of $\Delta t = 4$ and employed a two-layered fully connected neural network $NN(.)$ with ReLU activation functions. Each layers consisted of 25 neurons.We enforce conservation of mass, using the following constraint at each time step l:

$$c_l(\boldsymbol{X}_l) = 1 - \sum_{j=1}^{d_c} X_{l,j} = 0, \quad l = 0,1,\ldots \tag{18}$$

These are complemented by the virtual observables presented earlier and with $\sigma_c^2 = 10^{-9}$ (Equation (7)).

For the family of variational distributions $q_{\boldsymbol{\phi}}(\boldsymbol{X}^{(i)}_{(0:T)})$ and since $X^{(i)}_{t,j} > 0, \forall j,t$, we employed multivariate lognormals with a diagonal covariance matrices i.e. we assume $X^{(i)}_{t,j}$ are a posteriori independent. The mean and covariance matrix of the underlying Gaussians for each sequence

i become part of the parameters $\boldsymbol{\phi}$ with respect to which the ELBO is maximized (see Section 2.5). We note that it would also be possible to use an amortized formulation and explicitly account for the dependence on the data values by employing a neural network for both mean and covariance with the time sequence as an input.

3.3 Results

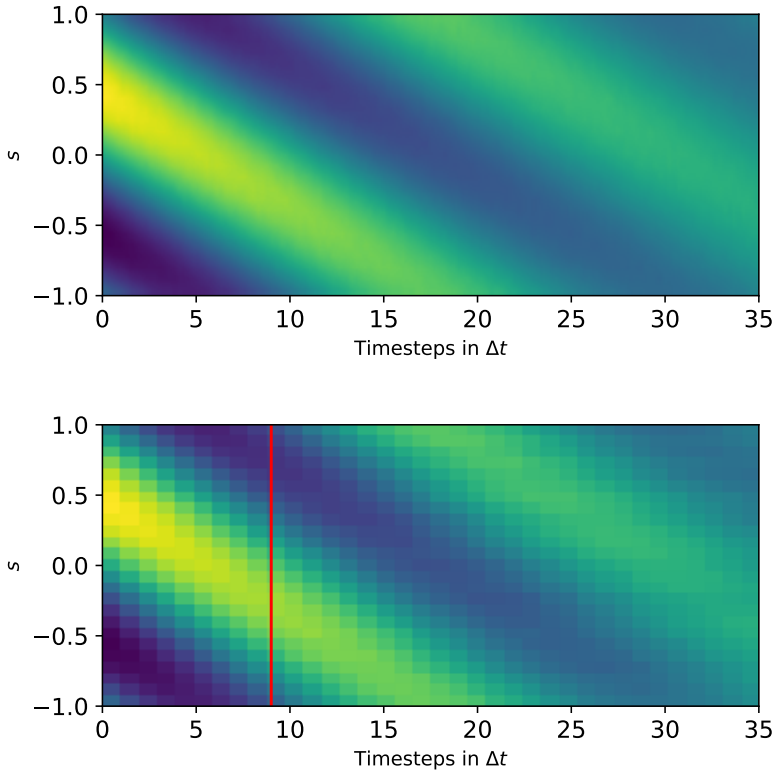


Figure 1: Particle density: Inferred and predicted posterior mean (bottom) in comparison with the ground truth (top). The red line divides inferred quantities from predicted ones.

We employed $n = 64$ time sequences with $T = 9$ for training and applied our framework in order to infer the unobserved CG states but more importantly the model parameters in right-hand side of the CG dynamics.

In Figure 1 we compare the true particle density with the one predicted by the trained CG model for one illustrative time sequence. We note that the latter is computed by reconstructing the x_t futures. The trained model is able to accurately track first-order statistics well into the future for many more time steps than those contained in the training data.

A more detailed view of the predictive estimates with snapshots of the particle density at selected time instances is presented in Figure 2 and 3 where the predictive posterior mean but also the associated uncertainty is displayed. Inferred as well as predicted particle densities

match accurately the ground-truth and reasonable uncertainty bounds are computed.

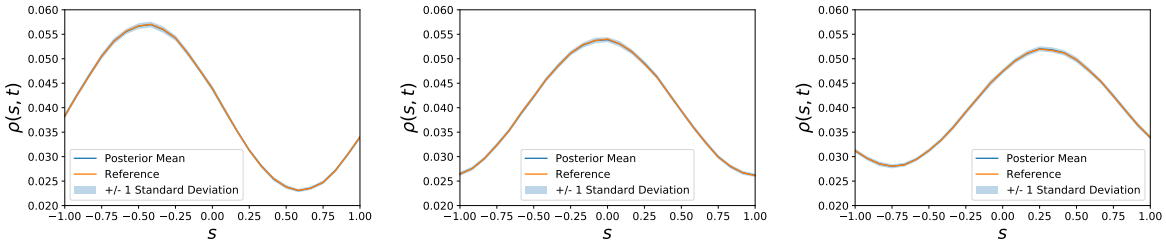


Figure 2: Inferred particle density profiles at $t = 0, 5\Delta t, 9\Delta t$ (from left to right).

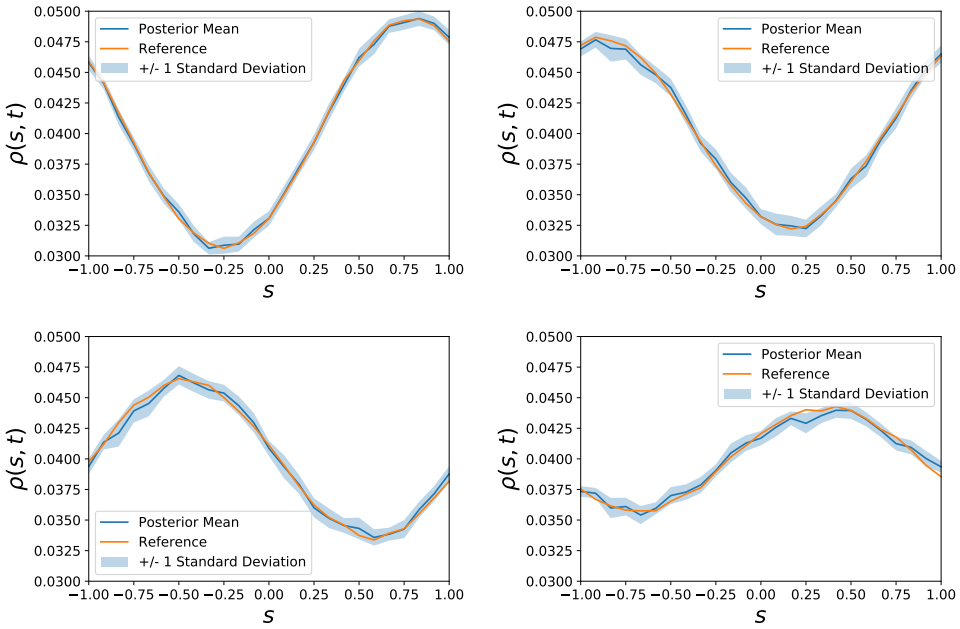


Figure 3: Predicted particle density profiles at $t = 15\Delta t, 20\Delta t, 25\Delta t, 35\Delta t$ (from left to right and top to bottom).

Finally, in Figure 4, the mass constraint is depicted for inferred as well as predicted particle densities and good agreement with the target value ($= 1$) is observed. This result is particularly important as it demonstrates that the virtual observables were able to find *CG* state variables that agree with an a priori given physical constraint and additionally a transition law has been learned that is able to automatically satisfy the constraint in the future.

4 CONCLUSIONS

We combined a probabilistic generative model with physical constraints and deep neural networks in order to obtain a framework for the automated discovery of coarse-grained variables

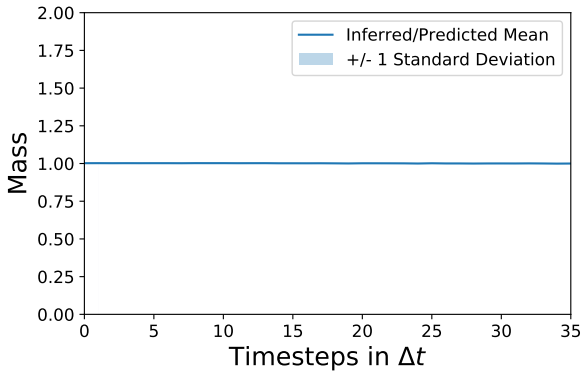


Figure 4: Mass based on inferred and predicted particle densities.

and dynamics based on fine-grained simulation data. The FG simulation data are augmented in a fully Bayesian fashion by virtual observables that enable the incorporation of physical constraints at the CG level. These could be for instance conservation laws that are available when CG variables have physical meaning. Deviations from such conservation laws would invalidate predictions. As a result of augmenting the training data with domain knowledge, the model proposed can learn from Small Data (i.e. shorter and fewer FG time-sequences) which is a crucial advantage in multiscale settings where the simulation of the FG dynamics is computationally very expensive.

Our approach learns simultaneously a coarse-to-fine mapping and a transition law for the coarse-grained dynamics by employing probabilistic inference tools for the latent variables and model parameters. Deep neural networks can be used in both of these components in order to endow great expressiveness and flexibility.

The model proposed was successfully tested on a coarse-graining task which involved stochastic particle dynamics. In the example presented, the method was able to accurately predict particle densities at time steps not contained in the training data. Moreover, as it is able to reconstruct the entire FG state vector at any future time instant, it is capable of producing predictions of any FG observable of interest as well as quantify the associated predictive uncertainty.

A shortcoming of presented framework is that the CG dynamics are not fully interpretable and long-term stability is not guaranteed. These limitations have been addressed in [23] where an additional layer of latent variables was employed that ensured the discovery of stable CG dynamics but also promoted the identification of slow-varying processes that are most predictive of the system's long-term evolution.

REFERENCES

[1] D.Givon, R.Kupferman and A.Stuart, Extracting Macroscopic Dynamics: Model Problems and Algorithms, Nonlinearity, 2004.

[2] Z. Ghahramani, Probabilistic machine learning and artificial intelligence, Nature, 2015

[3] Y. LeCun, Y. Bengio and G. Hinton, Deep Learning, Nature, 2015

[4] P.-S.Koutsourelakis, N.Zabaras and M. Girolami, Big data and predictive computational modeling, Journal of Computational Physics, 2016.

[5] M.Alber, A.Tepole, W.Cannon, S.De, S.Dura-Bernal, K.Garikipati, G. Karniadakis, W.Lytton, P.Perdikaris, L.Petzold and E.Kuhl, Integrating machine learning and multiscale modeling - perspectives, challenges,and opportunities in the biological, biomedical, and behavioral sciences, NPJ digital medicine, 2019

[6] S. Kaltenbach and P.-S. Koutsourelakis, Incorporating physical constraints in a deep probabilistic machine learning framework for coarse-graining dynamical systems, Journal of Computational Physics, 2020

[7] P.Stinis, T.Hagge, A.M.Tartakovsky and E.Yeung: Enforcing constraints for interpolation and extrapolation in generative adversarial networks, Journal of Computational Physics, 2019

[8] S. L. Brunton, J. L. Proctor and J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the National Academy of Sciences 113 (15) (2016) 3932–3937.

[9] R.Chen, Y.Rubanova, J.Bettencourt and D.Duvenaud, Neural ordinary differential equations, Advances in neural information porcessing systems, 2018

[10] X. Li, T.-K. Wong, R. TQ Chen and D. Duvenaud, Scalable gradients for stochastic differential equations, arXiv preprint 2001.01328, 2020

[11] M. Raissi and G.E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equation, Journal of Computational Physics 357 (2018) 125–142.

[12] P.-S. Koutsourelakis and I. Bilionis, Scalable Bayesian Reduced-Order Models for Simulating High-Dimensional Multiscale Dynamical Systems, Multiscale Modeling and Simulation, 2011

[13] H.Mori, Transport, collective motion, and brownian motion, Progress of theoretical physics, 33(3):423–455, 1965.

[14] R.Zwanzig, Nonlinear generalized langevin equations, Journal of Statistical Physics, 9(3):215–220, 1973.

[15] M. Schöberl, N. Zabaras and P.-S. Koutsourelakis, Predictive coarse-graining, Journal of Computational Physics 333 (2017) 49–77.

[16] M. Rixner and P.-S. Koutsourelakis, A probabilistic generative model for semi-supervised training of coarse-grained surrogates and enforcing physical constraints through virtual observables, Arxiv Preprint 2006.01789, 2020

[17] M.Hoffman, D.Blei, C. Wang and J.Paisley, Stochastic variational inference, The Journal of Machine Learning Research, 14(1):1303–1347, 2013.

[18] C. Bishop, Pattern Recognition and Machine Learning. Springer, 2006.

[19] D.Kingma and M.Welling, Auto-Encoding Variational Bayes, International Conference on

Learning Representations (ICLR), 2014.

[20] D.Kingma and J.Ba, Adam: A method for stochastic optimization, arXiv preprint 1412.6980, 2014

[21] G.Cottet and P.Koumoutsakos, Vortex Methods: Theory and Practice, Cambridge University Press, March 2000. ISBN 978-0-521-62186-1.

[22] L. Felsberger and P.-S. Koutsourelakis, Physics-constrained, data-driven discovery of coarse-grained dynamics, Communications in Computational Physics, 2019

[23] S. Kaltenbach and P.-S. Koutsourelakis, Physics-aware, probabilistic model order reduction with guaranteed stability, International Conference on Learning Representations (ICLR), 2021

C. Paper C

Reprinted from Kaltenbach and Koutsourelakis (2021b) according to the openreview guidelines[1].

[1] https://openreview.net/about

PHYSICS-AWARE, PROBABILISTIC MODEL ORDER REDUCTION WITH GUARANTEED STABILITY

Sebastian Kaltenbach, Phaedon-Stelios Koutsourelakis
Professorship of Continuum Mechanics
Technical University of Munich
{sebastian.kaltenbach,p.s.koutsourelakis}@tum.de

ABSTRACT

Given (small amounts of) time-series' data from a high-dimensional, fine-grained, multiscale dynamical system, we propose a generative framework for learning an effective, lower-dimensional, coarse-grained dynamical model that is predictive of the fine-grained system's long-term evolution but also of its behavior under different initial conditions. We target fine-grained models as they arise in physical applications (e.g. molecular dynamics, agent-based models), the dynamics of which are strongly non-stationary but their transition to equilibrium is governed by unknown slow processes which are largely inaccessible by brute-force simulations. Approaches based on domain knowledge heavily rely on physical insight in identifying temporally slow features and fail to enforce the long-term stability of the learned dynamics. On the other hand, purely statistical frameworks lack interpretability and rely on large amounts of expensive simulation data (long and multiple trajectories) as they cannot infuse domain knowledge. The generative framework proposed achieves the aforementioned desiderata by employing a flexible prior on the complex plane for the latent, slow processes, and an intermediate layer of physics-motivated latent variables that reduces reliance on data and imbues inductive bias. In contrast to existing schemes, it does not require the a priori definition of projection operators or encoders and addresses simultaneously the tasks of dimensionality reduction and model estimation. We demonstrate its efficacy and accuracy in multiscale physical systems of particle dynamics where probabilistic, long-term predictions of phenomena not contained in the training data are produced.

1 INTRODUCTION

High-dimensional, nonlinear systems are ubiquitous in engineering and computational physics. Their nature is in general multi-scale[1]. E.g. in materials, defects and cracks occur on scales of millimeters to centimeters whereas the atomic processes responsible for such defects take place at much finer scales (Belytschko & Song, 2010). Local oscillations due to bonded interactions of atoms (Smit, 1996) take place at time scales of femtoseconds ($10^{-15}s$), whereas protein folding processes which can be relevant for e.g. drug discovery happen at time scales larger than milliseconds ($10^{-3}s$). In Fluid Mechanics, turbulence phenomena are characterized by fine-scale spatiotemporal fluctuations which affect the coarse-scale response (Laizet & Vassilicos, 2009). In all of these cases, macroscopic observables are the result of microscopic phenomena and a better understanding of the interactions between the different scales would be highly beneficial for predicting the system's evolution (Givon et al., 2004). The identification of the different scales, their dynamics and connections however is a non-trivial task and is challenging from the perspective of statistical as well as physical modeling.

[1]With the term multiscale we refer to systems whose behavior arises from the synergy of two or more processes occurring at different (spatio)temporal scales. Very often these processes involve different physical descriptions and models (i.e. they are also multi-physics). We refer to the description/model at the finer scale as fine-grained and to the description/model at the coarser scale as coarse-grained.

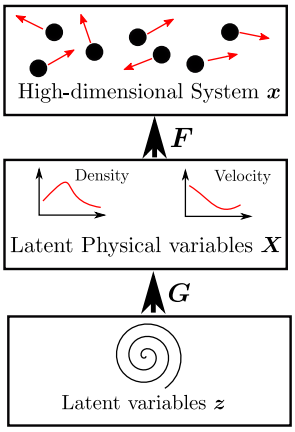


Figure 1: Visual summary of proposed framework. The low-dimensional variables $\boldsymbol{z}$ act via a probabilistic map $\boldsymbol{G}$ as generators of an intermediate layer of latent, physically-motivated variables $\boldsymbol{X}$ that are able to reconstruct the high-dimensional system $\boldsymbol{x}$ with another probabilistic map $\boldsymbol{F}$.

In this paper we propose a novel physics-aware, probabilistic model order reduction framework with guaranteed stability that combines recent advances in statistical learning with a hierarchical architecture that promotes the discovery of interpretable, low-dimensional representations. We employ a generative state-space model with two layers of latent variables. The first describes the latent dynamics using a novel prior on the complex plane that guarantees stability and yields a clear distinction between fast and slow processes, the latter being responsible for the system's long-term evolution. The second layer involves physically-motivated latent variables which infuse inductive bias, enable connections with the very high-dimensional observables and reduce the data requirements for training. The probabilistic formulation adopted enables the quantification of a crucial, and often neglected, component in any model compression process, i.e. the predictive uncertainty due to information loss. We finally want to emphasize that the problems of interest are *Small Data* ones due to the computational expense of the physical simulators. Hence the number of time-steps as well as the number of time-series used for training is small as compared to the dimension of the system and to the time-horizon over which predictions are sought.

2 Physics-Aware, Probabilistic Model Order Reduction

Our data consists of N times-series $\{\boldsymbol{x}_{0:T}^{(i)}\}_{i=1}^{N}$ over T time-steps generated by a computational physics simulator. This can represent positions and velocities of each particle in a fluid or those of atoms in molecular dynamics. Their dimension is generally very high i.e. $\boldsymbol{x}_t \in \mathcal{M} \subset \mathbb{R}^f$ ($f >> 1$). In the context of state-space models, the goal is to find a lower-dimensional set of collective variables or latent generators $\boldsymbol{z}_t$ and their associated dynamics. Given the difficulties associated with these tasks and the solutions that have been proposed in statistics and computational physics literature, we advocate the use of an intermediate layer of physically-motivated, lower-dimensional variables $\boldsymbol{X}_t$ (e.g. density or velocity fields), the meaning of which will become precise in the next sections. These variables provide a coarse-grained description of the high-dimensional observables and imbue interpretability in the learned dynamics. Using $\boldsymbol{X}_t$ alone (without $\boldsymbol{z}_t$) would make it extremely difficult to enforce long-term stability (see Appendix H.2) while ensuring sufficient complexity in the learned dynamics (Felsberger & Koutsourelakis, 2019; Champion et al., 2019). Furthermore and even if the dynamics of $\boldsymbol{x}_t$ are first-order Markovian, this is not necessarily the case for $\boldsymbol{X}_t$ (Chorin & Stinis, 2007). The latent variables $\boldsymbol{z}_t$ therefore effectively correspond to a nonlinear coordinate transformation that yields not only Markovian but also stable dynamics (Gin et al., 2019). The general framework is summarized in Figure 1 and we provide details in the next section.

2.1 MODEL STRUCTURE

Our model consists of three levels. At the first level, we have the latent variables $\boldsymbol{z}_t$ which are connected with $\boldsymbol{X}_t$ in the second layer through a probabilistic map $\boldsymbol{G}$. The physical variables $\boldsymbol{X}_t$ are finally connected to the high-dimensional observables through another probabilistic map $\boldsymbol{F}$. We parametrize $\boldsymbol{F}, \boldsymbol{G}$ with deep neural networks and denote by $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ the corresponding parameters (see Appendix D). In particular, we postulate the following relations:

$$z_{t,j} = z_{t-1,j} \exp(\lambda_j) + \sigma_j \epsilon_{t,j} \quad \lambda_j \in \mathbb{C}, \quad \epsilon_{t,j} \sim \mathcal{CN}(0,1),\ j = 1, 2, \ldots, h \tag{1}$$

$$\boldsymbol{X}_t = \boldsymbol{G}(\boldsymbol{z}_t, \boldsymbol{\theta}_1) \tag{2}$$

$$\boldsymbol{x}_t = \boldsymbol{F}(\boldsymbol{X}_t, \boldsymbol{\theta}_2) \tag{3}$$

We assume that the latent variables $\boldsymbol{z}_t$ are complex-valued and a priori independent. Complex variables were chosen as their evolution includes a harmonic components which are observed in many physical systems. In Appendix H.1 we present results with a real-valued latent variables $z_{t,j}$ and illustrate their limitations. We model their dynamics with a discretized Ornstein-Uhlenbeck process on the complex plane with initial conditions $z_{0,j} \sim \mathcal{CN}(0, \sigma_{0,j}^2)$[2]. The parameters associated with this level are denoted summarily by $\boldsymbol{\theta}_0 = \{\sigma_{0,j}^2, \sigma_j^2, \lambda_j\}_{j=1}^h$. These, along with $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ mentioned earlier, and the state variables $\boldsymbol{X}_t$ and $\boldsymbol{z}_t$ have to be inferred from the data $\boldsymbol{x}_t$. We explain each of the aforementioned components in the sequel.

2.1.1 STABLE LOW-DIMENSIONAL DYNAMICS

While the physical systems (e.g. particle dynamics) of interest are highly non-stationary, they generally converge to equilibrium in the long-term. We enforce long-term stability here by ensuring that the real-part of the λ_j's in Equation (1) is negative, i.e.:

$$\lambda_j = \Re(\lambda_j) + i\, \Im(\lambda_j) \text{ with } \Re(\lambda_j) < 0 \tag{4}$$

which guarantees first and second-order stability i.e. the mean as well as the variance are bounded at all time steps.

The transition density each process $\boldsymbol{z}_{t,j}$ is given by:

$$p\left(z_{t,j} \mid z_{t-1,j}\right) = \mathcal{N}\left(\begin{bmatrix}\Re(z_{t,j})\\ \Im(z_{t,j})\end{bmatrix} \mid s_j\, \boldsymbol{R}_j \begin{bmatrix}\Re(z_{t-1,j})\\ \Im(z_{t-1,j})\end{bmatrix},\ \boldsymbol{I}\frac{\sigma_j^2}{2}\right) \tag{5}$$

where the orthogonal matrix $\boldsymbol{R}_j$ depends on the imaginary part of λ_j:

$$\boldsymbol{R}_j = \begin{bmatrix}\cos(\Im(\lambda_j)) & -\sin(\Im(\lambda_j))\\ \sin(\Im(\lambda_j)) & \cos(\Im(\lambda_j))\end{bmatrix} \tag{6}$$

and the decay rate s_j depends on the real part of λ_j:

$$s_j = \exp(\Re(\lambda_j)) \tag{7}$$

i.e. the closer to zero the latter is, the "slower" the evolution of the corresponding process is. As in probabilistic Slow Feature Analysis (SFA) (Turner & Sahani, 2007; Zafeiriou et al., 2015), we set $\sigma_j^2 = 1 - \exp(2\,\Re(\lambda_j)) = 1 - s_j^2$ and $\sigma_{0,j}^2 = 1$. As a consequence, a priori, the latent dynamics are stationary[3] and an ordering of the processes $z_{t,j}$ is possible on the basis of $\Re(\lambda_j)$. Hence the only independent parameters are the λ_j, the imaginary part of which can account for periodic effects in the latent dynamics (see Appendix B).

The joint density of $\boldsymbol{z}_t$ can finally be expressed as:

$$p(\boldsymbol{z}_{0:T}) = \prod_{j=1}^{h}\left(\prod_{t=1}^{T} p(z_{t,j} \mid z_{t-1,j}, \boldsymbol{\theta}_0) p(z_{0,j} | \boldsymbol{\theta}_0)\right) \tag{8}$$

The transition density between states at non-neighbouring time-instants is also available analytically and is useful for training on longer trajectories or in cases of missing data. Details can be found in Appendix B.

[2] A short review of complex normal distributions, denoted by $\mathcal{CN}$, can be found in Appendix A.

[3] More details can be found in Appendix B.

2.1.2 PROBABILISTIC GENERATIVE MAPPING

We employ fully probabilistic maps between the different layers which involve two conditional densities based on Equations (2) and (3), i.e.:

$$p(\boldsymbol{x}_t \mid \boldsymbol{X}_t, \boldsymbol{\theta}_2) \quad \text{and} \quad p(\boldsymbol{X}_t \mid \boldsymbol{z}_t, \boldsymbol{\theta}_1) \tag{9}$$

In contrast to the majority of physics-motivated papers (Chorin & Stinis, 2007; Champion et al., 2019) as well as those based on transfer-operators Klus et al. (2018), we note that the generative structure adopted does not require the prescription of a restriction operator (or encoder) and the reduced variables need not be selected a priori but rather are adapted to best reconstruct the observables.

The splitting of the generative mapping into two parts through the introduction of the intermediate variables $\boldsymbol{X}_t$ has several advantages. Firstly, known physical dependencies between the data $\boldsymbol{x}$ and the physical variables $\boldsymbol{X}$ can be taken into account, which reduces the complexity of the associated maps and the total number of parameters. For instance, in the case of particle simulations where $\boldsymbol{X}$ represents a density or velocity field, i.e. it provides a coarsened or averaged description of the fine-scale observables, it can be used to (probabilistically) reconstruct the positions or velocities of the particles. This physical information can be used to compensate for the lack of data when only few training sequences are available (Small data) and can seen as a strong prior to the model order reduction framework. Due to the lower dimension of associated variables, the generative map between $\boldsymbol{z}_t$ and $\boldsymbol{X}_t$ can be more easily learned even with few training samples. Lastly, the inferred physical variables $\boldsymbol{X}$ can provide insight and interpretability to the analysis of the physical system.

2.2 INFERENCE AND LEARNING

Given the probabilistic relations above, our goal is to infer the state variables $\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)}$ as well as all model parameters $\boldsymbol{\theta}$. We follow a hybrid Bayesian approach in which the posterior of the state variables is approximated using structured Stochastic Variational Inference (Hoffman et al., 2013) and MAP point estimates for $\boldsymbol{\theta} = \{\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$ are computed.

The application of Bayes' rule leads to the following posterior:

$$p(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)}, \boldsymbol{\theta} | \boldsymbol{x}_{0:T}^{(1:n)}) = \frac{p(\boldsymbol{x}_{0:T}^{(1:n)} | \boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)}, \boldsymbol{\theta})\, p(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)}, \boldsymbol{\theta})}{p(\boldsymbol{x}_{0:T}^{(1:n)})} \tag{10}$$

$$= \frac{p(\boldsymbol{x}_{0:T}^{(1:n)} | \boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{\theta})\, p(\boldsymbol{X}_{0:T}^{(1:n)} \mid \boldsymbol{z}_{0:T}^{(1:n)}, \boldsymbol{\theta})\, p(\boldsymbol{z}_{0:T}^{(1:n)} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\boldsymbol{x}_{0:T}^{(1:n)})} \tag{11}$$

where $p(\boldsymbol{\theta})$ denotes the prior on the model parameters. In the context of variational inference, we use the following factorization of the approximate posterior[4]:

$$q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)}) = \prod_{i=1}^{n} \left(\prod_{j=0}^{h} q_{\boldsymbol{\phi}}(\boldsymbol{z}_{0:T,j}^{(i)} \mid \boldsymbol{X}_{0:T}^{(i)}) \right) q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(i)}) \tag{12}$$

We approximate the conditional posterior of $\boldsymbol{z}$ given $\boldsymbol{X}$ with a complex multivariate normal which is parameterized using a tridiagonal precision matrix as proposed in Archer et al. (2015); Bamler & Mandt (2017). This retains dependencies between temporally neighbouring $\boldsymbol{z}$, but the number of parameters grows linearly with the dimension of $\boldsymbol{z}$ which leads to a highly scalable algorithm. For the variational posterior of $\boldsymbol{X}$ we employ a Gaussian with a diagonal covariance, i.e.:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}_{0:T,j}^{(i)} \mid \boldsymbol{X}_{0:T}^{(i)}) = \mathcal{CN}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(i)}), \left[\boldsymbol{B}_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(i)}) \boldsymbol{B}_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(i)})^T \right]^{-1}) \qquad q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(i)}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\phi}}^{(i)}, \boldsymbol{\Sigma}_{\boldsymbol{\phi}}^{(i)}) \tag{13}$$

We denote summarily with $\boldsymbol{\phi}$ the parameters involved and note that deep neural networks are used for the mean $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(i)})$ as well as the upper bidiagonal matrix $\boldsymbol{B}_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(i)})$. Details on the neural net architectures employed are provided in Section 4 and in Appendix D.

[4]We note that this factorization does not introduce any error due to the conditional independence of $\boldsymbol{x}, \boldsymbol{z}$ given $\boldsymbol{X}$.

It can be readily shown that the optimal parameter values are found by maximizing the Evidence Lower Bound (ELBO) $\mathcal{F}(q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)}), \boldsymbol{\theta})$ which is derived in Appendix C. We compute Monte Carlo estimates of the gradient of the ELBO with respect to ϕ and $\boldsymbol{\theta}$ with the help of the reparametrization trick (Kingma & Welling, 2013) and carry out stochastic optimization with the ADAM algorithm (Kingma & Ba, 2014).

2.3 PREDICTIONS

Once state variables have been inferred and MAP estimates $\boldsymbol{\theta}^{MAP}$ for the model parameters have been obtained, the reduced model can be used for probabilistic future predictions. In order to do so for a time sequence used in training, we employ the following Monte Carlo scheme to generate a sample $\boldsymbol{x}_{T+P}$, i.e. P time-steps into the future:

1. Sample $\boldsymbol{X}_T$ and $\boldsymbol{z}_T$ from the inferred posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}_{0:T} \mid \boldsymbol{X}_{0:T}) q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T})$.
2. Propagate $\boldsymbol{z}_T$ for P time steps forward by using the conditional density in Equation (5).
3. Sample $\boldsymbol{X}_{T+P}$ and $\boldsymbol{x}_{T+P}$ from $p(\boldsymbol{X}_{T+P} \mid \boldsymbol{z}_{T+P}, \boldsymbol{\theta}_1^{MAP})$ and $p(\boldsymbol{x}_{T+P} \mid \boldsymbol{X}_{T+P}, \boldsymbol{\theta}_2^{MAP})$ respectively.

More importantly perhaps, the trained model can be used for predictions under new initial conditions, e.g. $\boldsymbol{x}_0$. To achieve this, first the posterior $p(\boldsymbol{z}_0|\boldsymbol{x}_0) \propto \int p(\boldsymbol{x}_0|\boldsymbol{X}_0, \boldsymbol{\theta}_2^{MAP})\ p(\boldsymbol{X}_0 \mid \boldsymbol{z}_0, \boldsymbol{\theta}_1^{MAP})\ d\boldsymbol{X}_0$ must be found before the Monte Carlo steps above can be employed starting at $T = 0$.

3 RELATED WORK

The main theme of our work is the learning of low-dimensional dynamical representations that are stable, interpretable and make use of physical knowledge.

Linear latent dynamics: In this context, the line of work that most closely resembles ours pertains to the use of Koopman-operator theory (Koopman, 1931) which attempts to identify appropriate transformations of the original coordinates that yield linear dynamics (Klus et al., 2018). We note that these approaches (Lusch et al., 2018; Champion et al., 2019; Gin et al., 2019; Lee & Carlberg, 2020) require additionally the specification of an encoder i.e. a map from the original description to the reduced coordinates which we avoid in the generative formulation adopted. Furthermore only a small fraction are probabilistic and can quantify predictive uncertainties but very often employ restrictive parametrizations for the Koopman matrix in order to ensure long-term stability (Pan & Duraisamy, 2020). To the best of our knowledge, none of the works along these lines employ additional, physically-motivated variables and as a result have demonstrated their applicability only in lower-dimensional problems and require very large amounts of training data or some ad hoc pre-processing. We provide comparative results with Koopman-based deterministic and probabilistic models in Appendix H.3.

Data-driven discovery of nonlinear dynamics: The data-driven discovery of governing dynamics has received tremendous attention in recent years. Efforts based on the Mori-Zwanzig formalism can accurately identify dynamics of pre-defined variables, which also account for memory effects, but cannot reconstruct the full fine-grained picture or make predictions about other quantities of interest (Chorin & Stinis, 2007; Kondrashov et al., 2015; Ma et al., 2019). Similar restrictions apply when neural-network-based models are employed as e.g. (Chen et al., 2018; Li et al., 2020). Efforts based on the popular SINDy algorithm (Brunton et al., 2016) require additionally data of the time-derivatives of the variables of interest which when estimated with finite-differences introduce errors and reduce robustness. Sparse Bayesian learning tools in combination with physically-motivated variables and generative models have been employed by (Felsberger & Koutsourelakis, 2019) but cannot guarantee the long-term stability of the learned dynamics as we also show in Appendix H.2 and in the context of the systems investigated in section 4.

Infusing domain knowledge from physics: Several efforts have been directed in endowing neural networks with invariances or equivariances arising from physical principles. Usually those pertain to translation or rotation invariance and are domain-specific as in Schütt et al. (2017). More general formulations such as Hamiltonian (Greydanus et al., 2019; Toth et al., 2019) and Lagrangian

Dynamics (Lutter et al., 2019) are currently restricted in terms of the dimension of the dynamical system. Physical knowledge has been exploited in conjunction with Gaussian Processes in (Camps-Valls et al., 2018) as well as in the context of PDEs for constructing reduced-order models as in (Grigo & Koutsourelakis, 2019) or for learning modulated derivatives using Graph Neural Networks as in (Seo et al., 2020). Another approach involves using physical laws as regularization terms or for augmenting the loss function as in (Raissi et al., 2019; Lusch et al., 2018; Zhu et al., 2019; Kaltenbach & Koutsourelakis, 2020). In the context of molecular dynamics multiple schemes for coarse-graining which also guarantee long-term stability have been proposed by Noé (2018) and Wu et al. (2017; 2018). In our formulation, physically-motivated latent variables are used to facilitate generative maps to very high-dimensional data and serve as the requisite information bottleneck in order to reduce the amount of training data needed.

Slowness and interpretability: Finally, in contrast to general state-space models for analyzing time-series data such as Karl et al. (2016); Rangapuram et al. (2018); Li et al. (2019), the prior proposed on the complex-valued $\boldsymbol{z}_t$ enable the discovery of slow features which are crucial in predicting the evolution of multiscale systems and in combination with the variables $\boldsymbol{X}_t$ can provide interpretability and insight into the underlying physical processes.

4 EXPERIMENTS

The high-dimensional, fine-grained model considered consists of f identical particles which can move in the bounded one-dimensional domain $s \in [-1,\ 1]$ (under periodic boundary conditions). The variables $\boldsymbol{x}_t$ consist therefore of the coordinates of the particles at each time instant t and the dimension of the system f is equal to the number of particles. We consider two types of stochastic particle dynamics that correspond to an advection-diffusion-type (section 4.1) and a viscous-Burgers'-type (section 4.2) behavior. In all experiments, the physically-motivated variables $\boldsymbol{X}_t$ relate to a discretization of the particle density into $d = 25$ equally-sized bins for advection-diffusion-type dynamics and into $d = 64$ equally-sized bins for viscous-Burgers'-type dynamics. In order to automatically enforce the conservation of mass at each time instant, we make use of the softmax function i.e. the particle density at a bin k is expressed as $\frac{\exp(X_{t,k})}{\sum_{l=0}^{d}\exp(X_{t,l})}$. Given this, the probabilistic map ($\boldsymbol{F}$ in Equation (3)) corresponds to a multinomial density, i.e.:

$$p(\boldsymbol{x}_t|\boldsymbol{X}_t) = \frac{f!}{m_1(\boldsymbol{x}_t)!\ m_2(\boldsymbol{x}_t)!\ldots m_k(\boldsymbol{x}_t)!}\prod_{k=1}^{d}\left(\frac{\exp(X_{t,k})}{\sum_{l=0}^{d}\exp(X_{t,l})}\right)^{m_k(\boldsymbol{x}_t)} \tag{14}$$

where $m_k(\boldsymbol{x}_t)$ is the number of particles in bin k. The underlying assumption is that, given $\boldsymbol{X}_t$, the coordinates of the particles $\boldsymbol{x}_t$ are *conditionally* independent. This does *not* imply that they move independently nor that they cannot exhibit coherent behavior (Felsberger & Koutsourelakis, 2019). Furthermore, the aforementioned model automatically satisfies permutation-invariance which a central feature of the fine-grained dynamics. This is another advantage of the physically motivated intermediate variables $\boldsymbol{X}$ as enforcing such symmetries/invariances is not a trivial task even when highly expressive models (e.g. neural networks) are used (Rezende et al., 2019). The practical consequence of Equation (14) is that no parameters $\boldsymbol{\theta}_2$ need to be inferred.

The second map pertains to $p(\boldsymbol{X}_t \mid \boldsymbol{z}_t, \boldsymbol{\theta}_1)$ which we represent with a multivariate normal distribution with a mean and a diagonal covariance matrix modeled by a neural network with parameters $\boldsymbol{\theta}_1$. Details of the parameterization can be found in Appendix D.

We assess the performance of the method by computing first- and second-order statistics as illustrated in the sequel as well as in Appendix F. We provide comparative results on the same examples in Appendix H where we report on the performance of various alternatives, such as a formulation without the latent variables $\boldsymbol{z}_t$ as well as deterministic and probabilistic Koopman-based models. Moreover a short study on the effect of the amount of training data is included in Appendix G.

4.1 PARTICLE DYNAMICS: ADVECTION-DIFFUSION

We train the model on $N = 64$ time-series of the positions of $f = 250 \times 10^3$ particles over $T = 40$ time-steps which were simulated as described in Appendix D.1. Furthermore, we made

use of $h = 5$ complex, latent processes $z_{t,j}$. Details regarding the variational posteriors and neural network architectures involved can be found in the Appendix D.1.

In Figure 2, the estimates of the complex-valued parameters λ_j are plotted as well as the inferred and predicted time-evolution of 2 associated processes $z_{t,j}$ on the complex plane. We note the clear separation of time-scales in the first plot with two slow processes, one intermediate and two fast ones. This is also evident in the indicative trajectories on the complex plane. A detailed discussion of the map $\boldsymbol{G}$ learned (Equation (2)) between $\boldsymbol{z}_t$ and $\boldsymbol{X}_t$ can be found in the Appendix E.

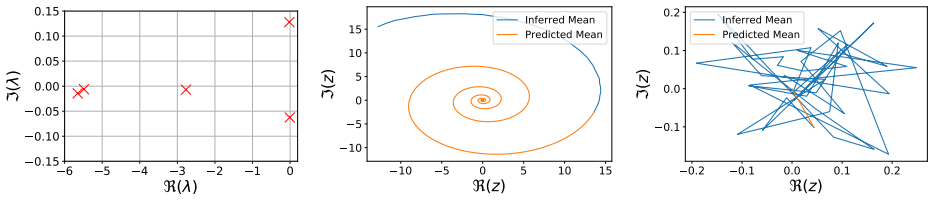


Figure 2: Estimated λ_j (left) and the time evolution of two $z_{t,j}$ processes where one is slow (middle) and the other fast (right).

In Figure 3 we compare the true particle density with the one predicted by the trained reduced model. We note that the latter is computed by reconstructing the $\boldsymbol{x}_t$ futures. We observe that the model is able to accurately track first-order statistics well into the future.

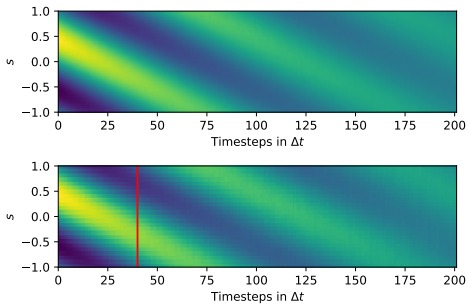


Figure 3: Particle density: Inferred and predicted posterior mean (bottom) in comparison with the ground truth (top). The red line divides inferred quantities from predicted ones. The horizontal axis corresponds to time-steps and the vertical to the one-dimensional spatial domain of the problem $s \in [-1, 1]$.

A more detailed view of the predictive estimates with snapshots of the particle density at selected time instances is presented in Figure 4. Here, not only the posterior mean but also the associated uncertainty is displayed. We want to emphasize the last Figure at $t = 1000$ when the steady state has been reached which clearly shows that our model is capable of converging to stable equilibrium.

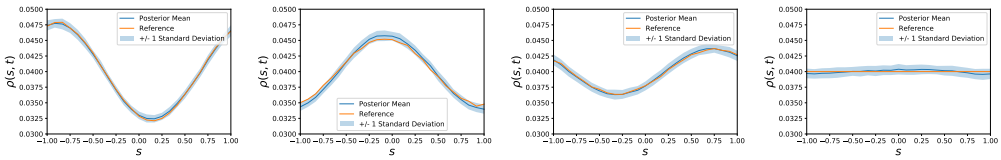


Figure 4: Predicted particle density profiles at $t = 80, 120, 160, 1000$ (from left to right).

Since the proposed model is capable of probabilistically reconstructing the whole fine-grained picture, i.e. $\boldsymbol{x}_t$, predictions with regards to any observable can be obtained. In Appendix F.1 we assess the accuracy of predictions in terms of second-order statistics, and in particular for the probability of finding simultaneously a pair of particles at two specified positions.

Finally, we demonstrate the accuracy of the trained in model in producing predictions under *unseen* initial conditions as described in section 2.3. Figure 5 depicts a new initial condition in terms of the particle density according to which particle positions $\boldsymbol{x}_0$ were drawn. The posterior on the corresponding $\boldsymbol{z}_0$ (see section 2.3) can be used to reconstruct the density as shown also on the same Figure.

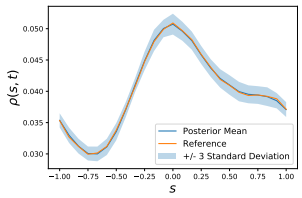


Figure 5: New initial condition (not used in training) in terms of the particle density. Reference shows the actual initial condition, whereas the posterior mean and uncertainty bounds correspond to the reconstruction of the initial condition based on the inferred latent variables $\boldsymbol{z}_0$.

Figure 6 shows predictions of the particle density (i.e. first-order statistics) at various timesteps. We want to emphasise the frame on the right, for $t = 500$ which shows the steady state of the system. We note that even though the initial condition was not contained in the training data, our framework is able to correctly track the system's evolution and to predict the correct steady state.

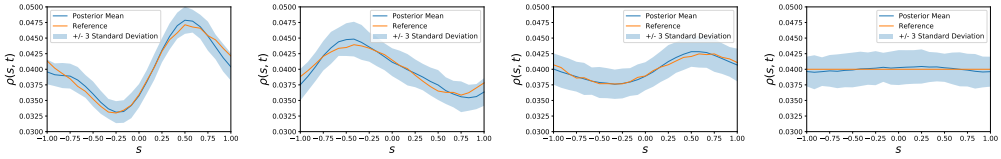


Figure 6: Predictions of the particle density at $t = 25, 75, 125, 500$ (left to right) for on the new initial condition in Figure 5.

4.2 Particle Dynamics: Viscous Burgers' Equation

In this example, we made use of $N = 64$ sequences of $f = 500 \times 10^3$ particles over $T = 40$ time-steps. Details regarding the physical simulator, the stochastic interactions between particles as well as the associated network architectures are contained in Appendix D.2. As in the previous example, we employed the particle density with the softmax transformation for $\boldsymbol{X}_t$ and $h = 5$ complex-valued processes $\boldsymbol{z}_t$ at the lowest model level.

In Figure 7 the estimated values for λ_j are shown where the clear separation of time-scales with three slow and two fast processes can be observed.

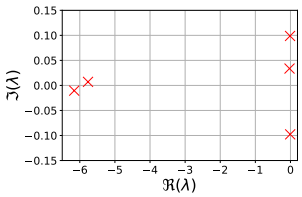


Figure 7: Estimated λ_j for the viscous Burgers' system.

In Figure 8 we compare the evolution of the true particle density with the (posterior mean of) the model-predicted one. We point out the sharp front at the lower left corner which is characteristic of the Burgers' equation and which eventually dissipates due to the viscosity. This is captured in the inferred as well as in the predicted solution.

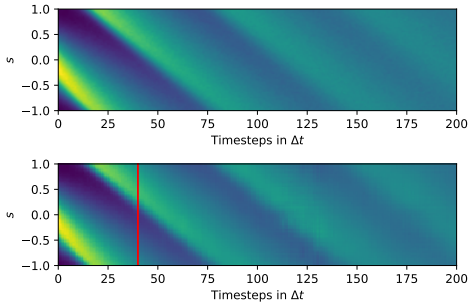


Figure 8: Particle density: Inferred and predicted posterior mean (bottom) in comparison with the ground truth (top). The red line divides inferred quantities from predicted ones. The horizontal axis corresponds to time-steps and the vertical to the one-dimensional spatial domain of the problem $s \in [-1, 1]$

A more detailed view on the predictive results with snapshots of the particle density at selected time instances is presented in Figure 9. We emphasize again the stable convergence of the learned dynamics to the steady state as well as the accuracy in capturing, propagating and dissipating the shock front.

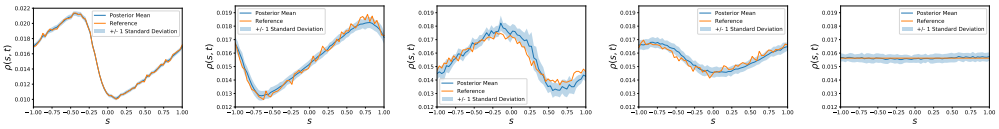


Figure 9: Predicted particle density profiles at $t = 40, 80, 120, 160, 1000$ (from left to right).

We compare the accuracy of the predictions for second-order statistics of the fine-grained system in terms of the two-particle probability in Appendix F.2 where excellent agreement with the ground truth, i.e. the one computed by simulating the fine-grained system, is observed.

5 Conclusions

We presented a framework for efficiently learning a lower-dimensional, dynamical representation of a high-dimensional, fine-grained system that is predictive of its long-term evolution and whose stability is guaranteed. We infuse domain knowledge with the help of an additional layer of latent variables. The latent variables at the lowest level provide an interpretabable separation of the time-scales and ensure the long-term stability of the learned dynamics. We employed scalable variational inference techniques and applied the proposed model in data generated from very large systems of interacting particles. In all cases accurate probabilistic predictions were obtained both in terms of first- and second-order statistics over a very long time range into the future. More importantly, the ability of the trained model to produce predictions under new, unseen initial conditions was demonstrated. An obvious limitation for the applicability of the proposed method to general dynamical systems pertains to the physically motivated variables $\boldsymbol{X}$. While such variables are available for several classes of physical systems, they need to be re-defined when moving to new problems and expert elicitation might be necessary. Furthermore, if an incomplete list of such variables is available from physical insight, this would need to be complemented by additional variables discovered from data. To that end, one could envision that some of the abstract latent variables $\boldsymbol{z}_t$ or functions thereof, e.g. $\boldsymbol{f}(\boldsymbol{z}_t)$, could be employed in a generative map of the form $\boldsymbol{x}_t = \boldsymbol{F}(\boldsymbol{X}_t, \boldsymbol{f}(\boldsymbol{z}_t))$ instead of Equation (3). A final deficiency of the proposed model is the lack of an automated procedure for determining the appropriate number of $\boldsymbol{z}$. We believe that the ELBO, which provides a lower bound on the model evidence, could be used for this purpose.

REFERENCES

H. H. Andersen, M. Højbjerre, D. Sørensen, and P. S. Eriksen. *The Multivariate Complex Normal Distribution*, pp. 15–37. Springer New York, New York, NY, 1995. ISBN 978-1-4612-4240-6. doi: 10.1007/978-1-4612-4240-6_2. URL https://doi.org/10.1007/978-1-4612-4240-6_2.

Evan Archer, Il Memming Park, Lars Buesing, John Cunningham, and Liam Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

Robert Bamler and Stephan Mandt. Structured black box variational inference for latent time series models. *arXiv preprint arXiv:1707.01069*, 2017.

Ted Belytschko and Jeong-Hoon Song. Coarse-graining of multiscale crack propagation. *International journal for numerical methods in engineering*, 81(5):537–563, 2010.

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

Gustau Camps-Valls, Luca Martino, Daniel H Svendsen, Manuel Campos-Taberner, Jordi Muñoz-Marí, Valero Laparra, David Luengo, and Francisco Javier García-Haro. Physics-aware gaussian processes in remote sensing. *Applied Soft Computing*, 68:69–82, 2018.

Kathleen P Champion, Steven L Brunton, and J Nathan Kutz. Discovery of nonlinear multiscale systems: Sampling strategies and embeddings. *SIAM Journal on Applied Dynamical Systems*, 18 (1):312–333, 2019.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583, 2018.

Alina Chertock and Doron Levy. Particle Methods for Dispersive Equations. *Journal of Computational Physics*, 171(2):708–730, August 2001. ISSN 0021-9991. doi: 10.1006/jcph.2001.6803. URL http://www.sciencedirect.com/science/article/pii/S0021999101968032.

Alexandre Chorin and Panagiotis Stinis. Problem reduction, renormalization, and memory. *Communications in Applied Mathematics and Computational Science*, 1(1):1–27, 2007.

Georges-Henri Cottet and Petros D. Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge University Press, Cambridge ; New York, 2 edition edition, March 2000. ISBN 978-0-521-62186-1.

L Felsberger and PS Koutsourelakis. Physics-constrained, data-driven discovery of coarse-grained dynamics. *Communications in Computational Physics*, 25(5):1259–1301, 2019. doi: 10.4208/cicp.OA-2018-0174.

Craig Gin, Bethany Lusch, Steven L Brunton, and J Nathan Kutz. Deep learning models for global coordinate transformations that linearize pdes. *arXiv preprint arXiv:1911.02710*, 2019.

D. Givon, R. Kupferman, and A. Stuart. Extracting Macroscopic Dynamics: Model Problems and Algorithms. *Nonlinearity*, 2004.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, pp. 15379–15389, 2019.

Constantin Grigo and Phaedon-Stelios Koutsourelakis. A physics-aware, probabilistic machine learning framework for coarse-graining high-dimensional systems in the small data regime. *Journal of Computational Physics*, 397:108842, 2019.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Sebastian Kaltenbach and Phaedon-Stelios Koutsourelakis. Incorporating physical constraints in a deep probabilistic machine learning framework for coarse-graining dynamical systems. *Journal of Computational Physics*, 419:109673, 2020.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick Van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Stefan Klus, Feliks Nüske, Péter Koltai, Hao Wu, Ioannis Kevrekidis, Christof Schütte, and Frank Noé. Data-Driven Model Reduction and Transfer Operator Approximation. *Journal of Nonlinear Science*, 28(3):985–1010, June 2018. ISSN 1432-1467. doi: 10.1007/s00332-017-9437-7. URL https://doi.org/10.1007/s00332-017-9437-7.

Dmitri Kondrashov, Mickaël D Chekroun, and Michael Ghil. Data-driven non-markovian closure models. *Physica D: Nonlinear Phenomena*, 297:33–55, 2015.

B. O. Koopman. Hamiltonian Systems and Transformations in Hilbert Space. *Proceedings of the National Academy of Sciences of the United States of America*, 17(5):315–318, 1931. ISSN 0027-8424. URL https://www.jstor.org/stable/86114.

S Laizet and JC Vassilicos. Multiscale generation of turbulence. *Journal of Multiscale Modelling*, 1 (01):177–196, 2009.

Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

Ju Li, Panayotis G. Kevrekidis, C. William Gear, and Ioannis G. Kevrekidis. Deciding the Nature of the Coarse Equation Through Microscopic Simulations: The Baby-Bathwater Scheme. *SIAM Rev.*, 49(3):469–487, July 2007. ISSN 0036-1445. doi: 10.1137/070692303. URL http://dx.doi.org/10.1137/070692303.

Longyuan Li, Junchi Yan, Xiaokang Yang, and Yaohui Jin. Learning interpretable deep state space model for probabilistic time series forecasting. In *IJCAI*, pp. 2901–2908, 2019.

Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. *arXiv preprint arXiv:2001.01328*, 2020.

Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490*, 2019.

Chao Ma, Jianchun Wang, and Weinan E. Model Reduction with Memory and the Machine Learning of Dynamical Systems. *Communications in Computational Physics*, 25(4):947–962, April 2019. ISSN 1815-2406. doi: 10.4208/cicp.OA-2018-0269. Place: Wanchai Publisher: Global Science Press WOS:000455963100001.

Frank Noé. Machine learning for molecular dynamics on long timescales. *arXiv preprint arXiv:1812.07669*, 2018.

Shaowu Pan and Karthik Duraisamy. Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability. *SIAM Journal on Applied Dynamical Systems*, 19(1):480–509, 2020.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. 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, 2019.

Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Advances in neural information processing systems*, pp. 7785–7794, 2018.

Danilo Jimenez Rezende, Sébastien Racanière, Irina Higgins, and Peter Toth. Equivariant Hamiltonian Flows. *arXiv:1909.13739 [cs, stat]*, September 2019. URL http://arxiv.org/abs/1909.13739. arXiv: 1909.13739.

Stephen Roberts. Convergence of a Random Walk Method for the Burgers Equation. *Mathematics of Computation*, 52(186):647–673, 1989. ISSN 0025-5718. doi: 10.2307/2008486. URL http://www.jstor.org/stable/2008486.

Kristof T Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8(1):1–8, 2017.

Sungyong Seo, Chuizheng Meng, and Yan Liu. Physics-aware difference graph networks for sparsely-observed dynamics. In *International Conference on Learning Representations*, 2020.

Berend Smit. *Understanding molecular simulation: from algorithms to applications*. Academic Press, 1996.

Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. *arXiv preprint arXiv:1909.13789*, 2019.

Richard Turner and Maneesh Sahani. A Maximum-Likelihood Interpretation for Slow Feature Analysis. *Neural Computation*, 19(4):1022–1038, April 2007. ISSN 0899-7667. doi: 10.1162/neco.2007.19.4.1022. URL https://doi.org/10.1162/neco.2007.19.4.1022.

Hao Wu, Feliks Nüske, Fabian Paul, Stefan Klus, Péter Koltai, and Frank Noé. Variational koopman models: slow collective variables and molecular kinetics from short off-equilibrium simulations. *The Journal of Chemical Physics*, 146(15):154104, 2017.

Hao Wu, Andreas Mardt, Luca Pasquali, and Frank Noe. Deep generative markov state models. In *Advances in Neural Information Processing Systems*, pp. 3975–3984, 2018.

Lazaros Zafeiriou, Mihalis A Nicolaou, Stefanos Zafeiriou, Symeon Nikitidis, and Maja Pantic. Probabilistic slow features for behavior analysis. *IEEE transactions on neural networks and learning systems*, 27(5):1034–1048, 2015.

Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

A COMPLEX NORMAL DISTRIBUTION

In this Appendix, the complex random normal distribution is reviewed. The mathematical definitions introduced follow Andersen et al. (1995):

A p-variate complex normal random variable $\boldsymbol{Y} \in \mathbb{C}^p$ with $\boldsymbol{Y} \sim \mathcal{CN}(\boldsymbol{\mu}_{\mathbb{C}}, \boldsymbol{\Sigma}_{\mathbb{C}})$ is defined by a complex mean vector $\boldsymbol{\mu}_{\mathbb{C}} \in \mathbb{C}^p$ and a complex Covariance Matrix $\boldsymbol{\Sigma}_{\mathbb{C}} \in \mathbb{C}_{+}^{p \times p}$. The density with respect to Lesbegue measures on $\mathbb{C}^p$ can be stated as:

$$f_{\boldsymbol{Y}}(\boldsymbol{y}) = \pi^{-p} \det(\boldsymbol{\Sigma}_{\mathbb{C}})^{-1} \exp\left(-(\boldsymbol{y} - \boldsymbol{\mu}_{\mathbb{C}})^{*} \boldsymbol{\Sigma}_{\mathbb{C}}^{-1} (\boldsymbol{y} - \boldsymbol{\mu}_{\mathbb{C}})\right) \tag{15}$$

where $*$ indicates the conjugate transpose of a matrix.

This complex normal random variable has similar properties to the well-known, real-valued counterpart. For instance, linear transformations of complex random normal variables are again complex random normal variables.

These properties directly follow from the fact, that for a complex random normal variable there exists an isomorphic transformation to a real valued $2p$-variate normal random variable $W \in \mathbb{R}^{2p}$. This random normal variable is defined with mean

$$\boldsymbol{\mu}_{\mathbb{R}} = \begin{bmatrix} \Re(\boldsymbol{\mu}_{\mathbb{C}}) \\ \Im(\boldsymbol{\mu}_{\mathbb{C}}) \end{bmatrix} \tag{16}$$

and covariance

$$\boldsymbol{\Sigma}_{\mathbb{R}} = \frac{1}{2} \begin{bmatrix} \Re(\boldsymbol{\Sigma}_{\mathbb{C}}) & -\Im(\boldsymbol{\Sigma}_{\mathbb{C}}) \\ \Im(\boldsymbol{\Sigma}_{\mathbb{C}}) & \Re(\boldsymbol{\Sigma}_{\mathbb{C}}) \end{bmatrix} \tag{17}$$

Therefore: $W \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbb{R}}, \boldsymbol{\Sigma}_{\mathbb{R}})$

As an example the real valued isomorphic counterpart of the standard complex random normal distribution $\mathcal{CN}(0, 1)$ is the bivariate normal distribution $\mathcal{N}(\mathbf{0}, \frac{1}{2}\mathbf{I})$.

B CHOICE OF VARIANCE FOR A-PRIORI STEADY STATE

We derive transient and stationary properties of the complex-valued latent processes $z_{t,j}$ and justify our choices for the model parameters. Based on Equation (1) and for each process j, the dynamics can be written in terms of the real and imaginary parts in two dimensions as follows:

$$\begin{bmatrix}\Re(z_{t,j})\\ \Im(z_{t,j})\end{bmatrix} = \boldsymbol{A}_j \begin{bmatrix}\Re(z_{t,j-1})\\ \Im(z_{t,j-1})\end{bmatrix} + \sigma_j \begin{bmatrix}\Re(\epsilon_{t,j})\\ \Im(\epsilon_{t,j})\end{bmatrix}, \tag{18}$$

or:

$$\begin{bmatrix}\Re(z_{t,j})\\ \Im(z_{t,j})\end{bmatrix} = \boldsymbol{A}_j^t \begin{bmatrix}\Re(z_{t,0})\\ \Im(z_{t,0})\end{bmatrix} + \sigma_j \sum_{k=0}^{t-1} \boldsymbol{A}_j^k \begin{bmatrix}\Re(\epsilon_{t-k,j})\\ \Im(\epsilon_{t-k,j})\end{bmatrix} \tag{19}$$

where $\Re(\epsilon_{t,j}), \Im(\epsilon_{t,j}) \sim \mathcal{N}(0, 1/2)$. The matrix $\boldsymbol{A}_j$ is given by (see also Equations (6), (7)):

$$\boldsymbol{A}_j = s_j \boldsymbol{R}_j = e^{\Re(\lambda_j)} \begin{bmatrix}\cos(\Im(\lambda_j)) & -\sin(\Im(\lambda_j))\\ \sin(\Im(\lambda_j)) & \cos(\Im(\lambda_j))\end{bmatrix} \tag{20}$$

and can be diagonalized as $\boldsymbol{A}_j = \boldsymbol{V}\boldsymbol{P}_j\,\boldsymbol{V}^*$ where:

$$\boldsymbol{V} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ -i & i\end{bmatrix}, \qquad \boldsymbol{P}_j = \begin{bmatrix}e^{\lambda_j} & 0\\ 0 & e^{\bar{\lambda}_j}\end{bmatrix}, \tag{21}$$

$\boldsymbol{V}^*$ is the conjugate transpose of $\boldsymbol{V}$ and $\bar{\lambda}_j$ denotes the complex conjugate of λ_j. Due to the linearity of the model and the Gaussian initial conditions, i.e. $\Re(z_{0,j}), \Im(z_{0,j}) \sim \mathcal{N}(0, \sigma_{0,j}^2/2)$, the marginal will remain Gaussian at all times t. A direct consequence of the above is that the mean of real and imaginary parts is always zero, i.e.:

$$\boldsymbol{\mu}_t^{(j)} = \begin{bmatrix}\mathbb{E}[\Re(z_{t,j})]\\ \mathbb{E}[\Im(z_{t,j})]\end{bmatrix} = \begin{bmatrix}0\\ 0\end{bmatrix}. \tag{22}$$

Furthermore, the covariance $\boldsymbol{C}_t^{(j)}$ is given by:

$$\boldsymbol{C}_t^{(j)} = \mathbb{E}\left[\begin{bmatrix}\Re(z_{t,j})\\ \Im(z_{t,j})\end{bmatrix}[\Re(z_{t,j})\ \Im(z_{t,j})]\right] = \frac{\sigma_{0,j}^2}{2}\boldsymbol{A}_j^t(\boldsymbol{A}_j^T)^t + \sigma_k^2 \sum_{k=0}^{t-1} \boldsymbol{A}_j^k(\boldsymbol{A}_j^T)^k \tag{23}$$

which upon use of the diagonalized form of $\boldsymbol{A}_j$ yields:

$$\boldsymbol{C}_t^{(j)} = \frac{\sigma_{0,j}^2}{2}\boldsymbol{V}\begin{bmatrix}0 & e^{2t\Re(\lambda_j)}\\ e^{2t\Re(\lambda_j)} & 0\end{bmatrix}\boldsymbol{V}^T + \frac{\sigma_j^2}{2}\boldsymbol{V}\begin{bmatrix}0 & \frac{1-e^{2t\Re(\lambda_j)}}{1-e^{2\Re(\lambda_j)}}\\ \frac{1-e^{2t\Re(\lambda_j)}}{1-e^{2\Re(\lambda_j)}} & 0\end{bmatrix}\boldsymbol{V}^T \tag{24}$$

As mentioned earlier, a necessary condition for the long-term stability of the processes is that $\Re(\lambda_j) < 0$, in which case and as $t \to \infty$ it leads to:

$$\boldsymbol{C}_t^{(j)} \to \boldsymbol{C}_\infty^{(j)} = \frac{\sigma_j^2}{2}\boldsymbol{V}\begin{bmatrix}0 & \frac{1}{1-e^{2\Re(\lambda_j)}}\\ \frac{1}{1-e^{2\Re(\lambda_j)}} & 0\end{bmatrix}\boldsymbol{V}^T = \frac{\sigma_j^2}{2(1-e^{2\Re(\lambda_j)})}\boldsymbol{I} \tag{25}$$

This implies that real and imaginary parts are asymptotically uncorrelated with a variance $\frac{\sigma_j^2}{2(1-e^{2\Re(\lambda_j)})}$. By setting $\sigma_j^2 = 1 - e^{2\Re(\lambda_j)}$ we enable direct comparisons between $z_{t,j}$ solely on the basis of $Re(\lambda_j)$ i.e. the degree of slowness (Turner & Sahani, 2007; Zafeiriou et al., 2015)). In this case the asymptotic variance of real and imaginary parts becomes $1/2$. Finally by setting $\sigma_{0,j}^2 = 1$ we ensure that, a priori, the processes $z_{t,j}$ are *stationary*, with $\boldsymbol{C}_t^{(j)} = \boldsymbol{C}_\infty^{(j)} = \frac{1}{2}\boldsymbol{I}$, i.e. no a-priori bias is introduced with regards to their transient characteristics.

We finally note that the autocovariance $\boldsymbol{D}_\tau^{(j)}$ of each of these stationary processes j is given by:

$$\boldsymbol{D}_\tau^{(j)} = \mathbb{E}\left[\begin{bmatrix}\Re(z_{t+\tau,j})\\ \Im(z_{t+\tau,j})\end{bmatrix}[\Re(z_{t,j})\ \Im(z_{t,j})]\right] = \boldsymbol{A}_j^\tau \mathbb{E}\left[\begin{bmatrix}\Re(z_{t,j})\\ \Im(z_{t,j})\end{bmatrix}[\Re(z_{t,j})\ \Im(z_{t,j})]\right] = \boldsymbol{A}_j^\tau\,\boldsymbol{C}_t^{(j)} \tag{26}$$

where $\boldsymbol{A}_j$ and the covariance $\boldsymbol{C}_t^{(j)}$ are given above. By exploiting the diagonalization of $\boldsymbol{A}_j$ and that $\boldsymbol{C}_t^{(j)} = \boldsymbol{C}_\infty^{(j)} = \frac{1}{2}\boldsymbol{I}$ (for the parameter values discussed earlier), we obtain that:

$$\boldsymbol{D}_\tau^{(j)} = e^{\tau\Re(\lambda_j)}\begin{bmatrix}\cos(\tau\Im(\lambda_j)) & -\sin(\tau\Im(\lambda_j))\\ \sin(\tau\Im(\lambda_j)) & \cos(\tau\Im(\lambda_j))\end{bmatrix} \tag{27}$$

One can clearly observe harmonic (cross-)correlation terms which depend on the imaginary part of the λ_j and can capture persistent periodic effects of the dynamical system in the long-time range.

C DERIVATION OF THE ELBO

This section contains details of the derivation of the Evidence-Lower-Bound (ELBO) which serves as the objective function for the determination of the parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ during training. In particular:

$$
\begin{aligned}
&\log p(\boldsymbol{x}_{0:T}^{(1:n)}|\boldsymbol{\theta}) \\
&= \log \int p(\boldsymbol{x}_{0:T}^{(1:n)}, \boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)}, \boldsymbol{\theta})\, d\boldsymbol{X}_{0:T}^{(1:n)}\, \boldsymbol{z}_{0:T}^{(1:n)} \\
&= \log \int \frac{p(\boldsymbol{x}_{0:T}^{(1:n)}|\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)}, \boldsymbol{\theta}) p(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)}, \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)})} q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)})\, d\boldsymbol{X}_{0:T}^{(1:n)}\, d\boldsymbol{z}_{0:T}^{(1:n)} \\
&\geq \int \log \frac{p(\boldsymbol{x}_{0:T}^{(1:n)}|\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)}, \boldsymbol{\theta}) p(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)}, \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)})} q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)})\, d\boldsymbol{X}_{0:T}^{(1:n)}\, d\boldsymbol{z}_{0:T}^{(1:n)} \\
&= \mathcal{F}(q_{\boldsymbol{\phi}}(\boldsymbol{X}_{0:T}^{(1:n)}, \boldsymbol{z}_{0:T}^{(1:n)}), \boldsymbol{\theta})
\end{aligned}
\tag{28}
$$

D DETAILS FOR EXPERIMENTS

This appendix contains details for our experiments involving moving particles that have stochastic interactions corresponding to either an Advection-Diffusion behaviour or viscous Burgers' type behaviour.

D.1 PARTICLE DYNAMICS: ADVECTION-DIFFUSION

For the simulations presented $f = 250 \times 10^3$ particles were used, which, at each microscopic time step $\delta t = 2.5 \times 10^{-3}$ performed random, non-interacting, jumps of size $\delta s = \frac{1}{640}$, either to the left with probability $p_{left} = 0.1875$ or to the right with probability $p_{right} = 0.2125$. The positions were restricted in $[-1, 1]$ with periodic boundary conditions. It is well-known (Cottet & Koumoutsakos, 2000) that in the limit (i.e. $f \to \infty$) the particle density $\rho(s, t)$ can be modeled with an advection-diffusion PDE with diffusion constant $D = (p_{left} + p_{right})\frac{\delta s^2}{2\delta t}$ and velocity $v = (p_{right} - p_{left})\frac{\delta s}{\delta t}$:

$$\frac{\partial \rho}{\partial t} + v\frac{\partial \rho}{\partial s} = D\frac{\partial^2 \rho}{\partial s^2}, \qquad s \in (-1, 1).. \tag{29}$$

From this simulation every 800th microscopic time step the particle positions were extracted and used as training data for our system. Sample initial conditions are shown in Figure 10:

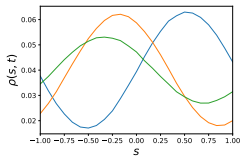


Figure 10: Sample initial conditions for the advection-diffusion type dynamics.

The architecture of the neural networks for the generative mappings described above as well as for the variational posteriors introduced in Section 2.2 can be seen in Figure 11. The neural network used for the generative mapping between the low-dimensional states $\boldsymbol{z}_t$ and the mean and covariance for $\boldsymbol{X}_t$ consists of only one dense layer, whereas the variational posterior on $\boldsymbol{z}_{0:T}$ is parameterized using a dense Layer with ReLu activation followed by another dense layer.

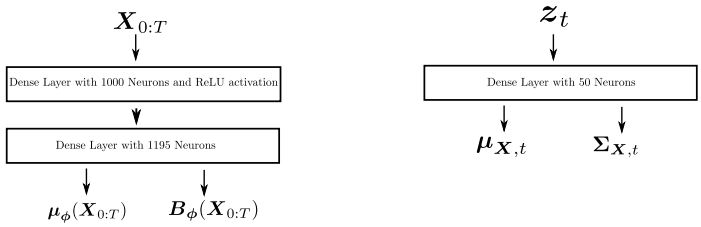


Figure 11: Neural Net architecture used for the particle dynamics corresponding to an advection-diffusion equation.

D.2 PARTICLE DYNAMICS: VISCOUS BURGERS' EQUATION

The second test-case involved a fine-grained system of $f = 500 \times 10^3$ particles which perform *interactive* random walks i.e. the jump performed at each fine-scale time-step $\delta t = 2.5 \times 10^{-3}$ depends on the positions of the other walkers. In particular we adopted interactions as described in Roberts (1989); Chertock & Levy (2001); Li et al. (2007) so as, in the limit (i.e. when $f \to \infty, \delta t \to 0, \delta s \to 0$), the particle density $\rho(s, t)$ follows a viscous Burgers' equation with $\nu = 0.0005$:

$$\frac{\partial \rho}{\partial t} + \frac{1}{2}\frac{\partial \rho^2}{\partial s} = \nu\frac{\partial^2 \rho}{\partial t^2}, \qquad s \in (-1, 1). \tag{30}$$

From this simulation every 800th microscopic time step the particle positions were extracted and used as training data for our system. Sample initial conditions are shown in Figure 12.

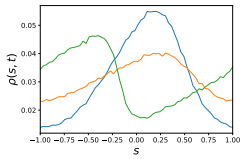


Figure 12: Sample initial conditions for the Burger's type dynamics.

The architecture of the neural networks for the generative mappings described above as well as for the variational posteriors introduced in Section 2.2 can be seen in Figure 13. The neural network used for the generative mapping between the low-dimensional states $\boldsymbol{z}_t$ and the mean and covariance for $\boldsymbol{X}_t$ consists of several dense layers with ReLu activation and Dropout layers to avoid overfitting, whereas the variational posterior on $\boldsymbol{z}_{0:T}$ is parameterized using two dense layers with ReLu activation followed by another dense layer.

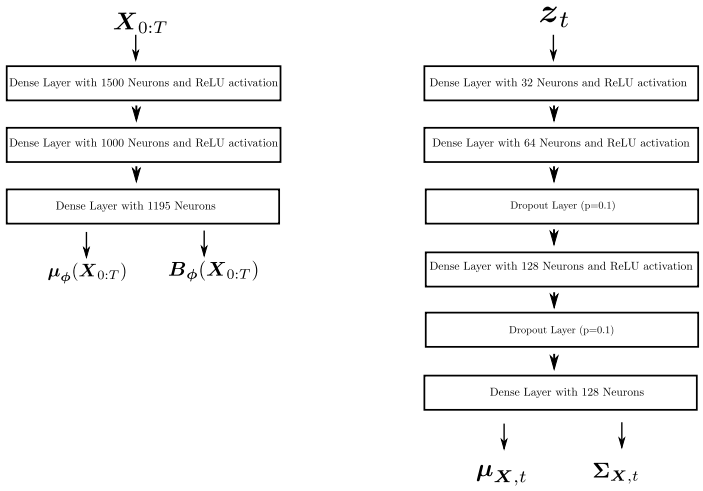


Figure 13: Neural Net architecture used for the particle dynamics corresponding to a viscous Burgers' equation.

E DETAILED ANALYSIS OF THE GENERATIVE MAPPING AND THE SLOW LATENT VARIABLES

In this Appendix we take a closer look at the generative mapping and the (slow) latent variables $\boldsymbol{z}$ learned . For the Advection-Diffusion example, we discovered two slow processes (Section 4.1), $\boldsymbol{z}_1$ and the marginally faster process $\boldsymbol{z}_2$. The rest of the processes were very fast in comparison and took values close to the zero point of the complex plane during the inference as well as during the prediction phase.

In order to visualize the influence through the generative mapping of these two slow processes, we set the value of all other processes to zero and then reconstructed the fine-grained state based on different absolute values of $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. The result are shown in Figure 14 in terms of the reconstructed particle density. It is clearly visible that those two processes are responsible for a variety of density profiles. In accordance with their slowness, $\boldsymbol{z}_1$ (the slightly slower process) is responsible for the most striking changes, whereas the other slow process generates some smaller scale fluctuations.

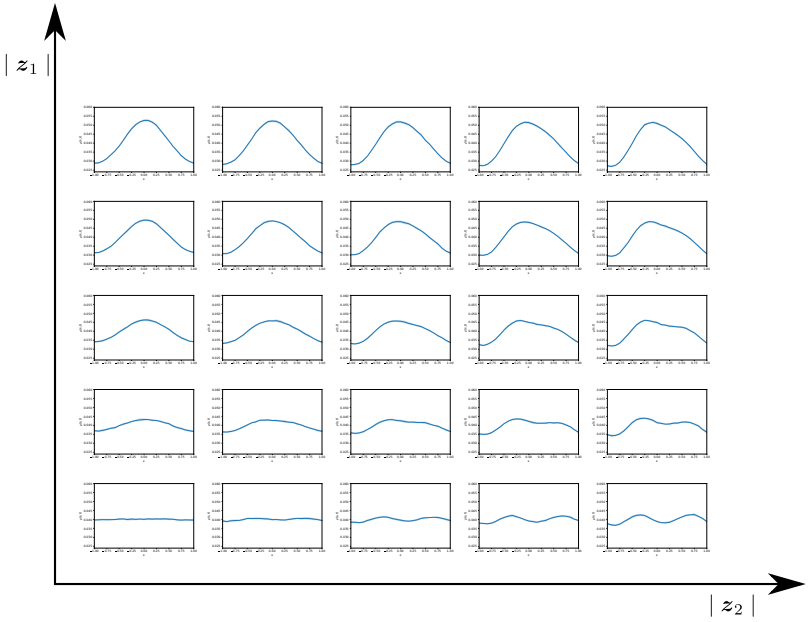


Figure 14: Reconstruction of the particle density profiles for various values of the two slowest processes $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ identified. All other latent variables $z_{t,j}$ were set to zero.

F TWO-POINT PROBABILITY

This appendix contains the predictive estimates for the two-point probability, i.e. the probability of finding two particles simultaneously in two bins (b_1, b_2). This two-point probability can be computed based on the reconstructed fine-grained system and corresponds to a second order statistic.

F.1 PARTICLE DYNAMICS: ADVECTION-DIFFUSION

The estimated two-point probability as well as the comparison to test data is shown for two indicative time-instants in Figure 15 and 16. We note a very good agreement with the ground truth.

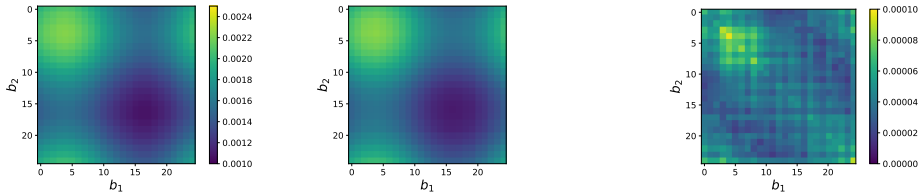


Figure 15: Two-point probability at time step 90: On the left the two-point probability of the data is shown as reference, the figure in the middle contains the predictive posterior mean whereas the figure on the right contains the standard deviation. The figure on the left and the figure in the middle share the same colorbar.

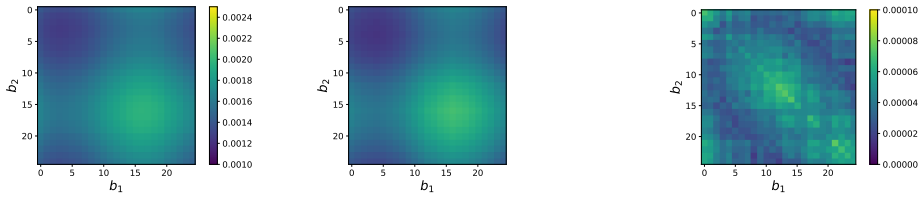


Figure 16: Two-point probability at time step 140: On the left the two-point probability of the data is shown as reference, the figure in the middle contains the predictive posterior mean whereas the figure on the right contains the standard deviation. The figure on the left and the figure in the middle share the same colorbar.

F.2 PARTICLE DYNAMICS: VISCOUS BURGERS' EQUATION

The estimated two-point probability as well as the comparison to test data is shown for two indicative time-instants in Figure 17 and 18. We note a very good agreement with the ground truth.

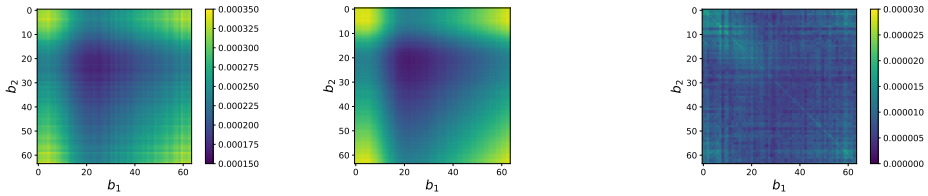


Figure 17: Two-point probability at time step 90: On the left the two-point probability of the data is shown as reference, the figure in the middle contains the predictive posterior mean whereas the figure on the right contains the standard deviation. The figure on the left and the figure in the middle share the same colorbar.

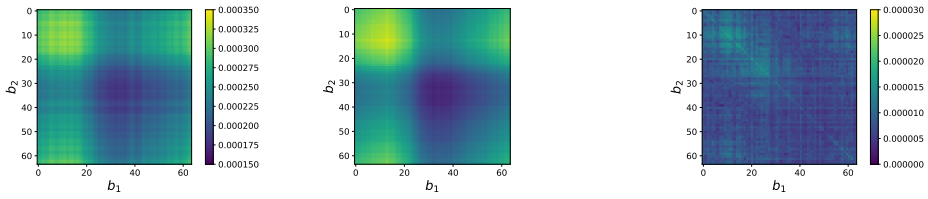


Figure 18: Two-point probability at time step 140: On the left the two-point probability of the data is shown as reference, the figure in the middle contains the predictive posterior mean whereas the figure on the right contains the standard deviation. The figure on the left and the figure in the middle share the same colorbar.

G EFFECT OF THE AMOUNT OF TRAINING DATA

This section contains a study on the influence of the amount of training data. We illustrate this in the context of the Advection-Diffusion example (section 4.1) by using 16 time sequences instead of the 64 employed earlier. In the figures below we note that the proposed model is capable capturing the main features of the system's dynamics as well as the correct steady state even with fewer training data. We believe that this is due to the intermediate layer of physically-motivated variables $\boldsymbol{X}_t$ which introduces an information bottleneck. Finally, and as one would expect, fewer training data leads to increased predictive uncertainty.

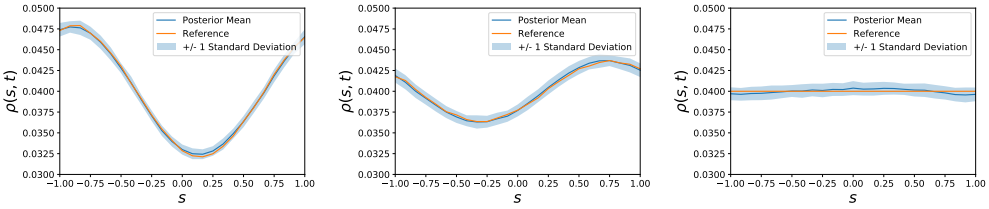


Figure 19: Predicted particle density profiles at $t = 80, 160, 1000$ (from left to right) with 64 samples.

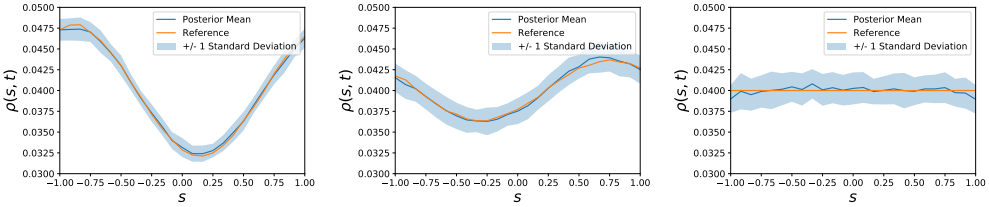


Figure 20: Predicted particle density profiles at $t = 80, 160, 1000$ (from left to right) with 16 samples.

H COMPARISON WITH OTHER APPROACHES

This appendix contains results obtained by other methods for the test cases discussed in the main text. The simulation data is identical to the one used for the proposed method and details of the specific algorithms are described in the following.

H.1 REAL-VALUED LATENT SPACE

To demonstrate the utility of a complex-valued latent space, the two examples were also solved with a real-valued $z_{t,j}$. The only difference here is the restriction of the latent variables $z_{t,j}$ and the λ_j to real values.

A model with real-valued latent space is also capable of ensuring the stability but it is not capable of capturing periodic components of the dynamics. As most physical systems (including the two examples) contain such components, the algorithm is not able to accurately model the dynamics and fails in generating reliable predictions. This can be readily observed in the extrapolative predictions of Figures 21 and 22 where, apart for the biased results, one can also note increased predictive uncertainty.

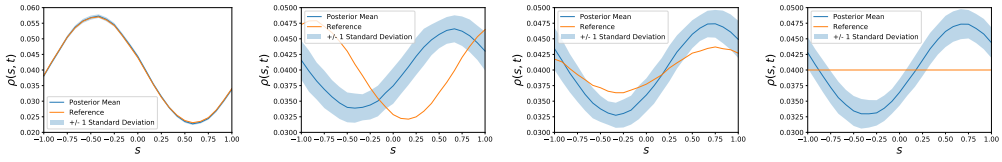


Figure 21: Advection-Diffusion system: Predictions at $t = 0, 80, 160$ and 1000 obtained with real-valued latent variables $z_{t,j}$.

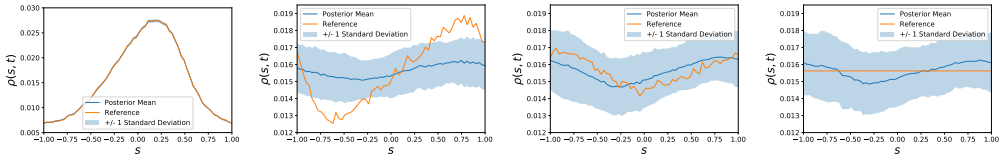


Figure 22: Burgers' system: Predictions at $t = 0, 80, 160$ and 1000 obtained with real-valued latent variables $z_{t,j}$.

H.2 NO STABLE LATENT SPACE

Another possibility is to employ only the physically motivated latent variables $\boldsymbol{X}_t$ and remove completely the latent variables $\boldsymbol{z}_t$ in the first layer. This approach is similar to the one investigated in Felsberger & Koutsourelakis (2019) and Kaltenbach & Koutsourelakis (2020) as well as to the idea of neural ODEs (Chen et al., 2018). In this case, one must learn directly the dynamics of $\boldsymbol{X}_t$ and for this purpose we employed a three-layer, fully-connected neural network NN as follows:

$$\boldsymbol{X}_{t+1} = NN(\boldsymbol{X}_t) + \sigma\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{31}$$

The learned dynamics are in general non-linear and stability is not guaranteed. As it can be observed in Figures 23 and 24, the trained model is capable of producing accurate predictions for some time-steps but eventually in both cases predictions become unstable. This could also be problematic when the trained model is used to make predictions with new initial conditions as the chaotic nature of the nonlinear dynamics can lead to significant errors even for shorter time horizons.

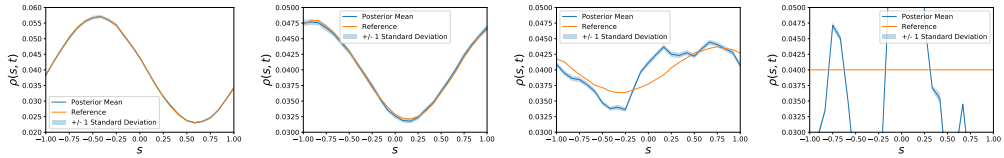


Figure 23: Advection-Diffusion system: Predictions at $t = 0, 80, 160$ and 1000 obtained without $\boldsymbol{z}_t$ and with the model of Equation (31).

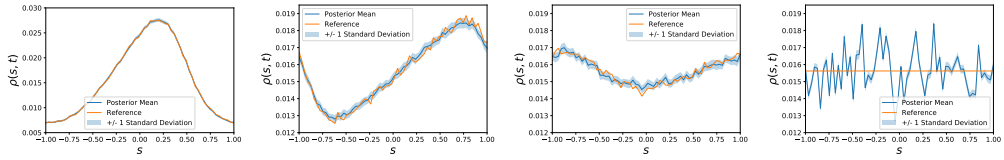


Figure 24: Burgers' system: Predictions at $t = 0, 80, 160$ and 1000 obtained without $\boldsymbol{z}_t$ and with the model of Equation (31).

H.3 KOOPMAN-BASED MODELS

The final alternative explored involved probabilistic and deterministic Koopman-based models for the latent dynamics. We kept the generative framework of our model and did not use an encoder as for instance in Gin et al. (2019) in order to remove the effect of the associated model choice. For the same reason, we retained the intermediate variables $\boldsymbol{X}_t$ even though these do not appear in any known Koopman-operator implementations. We replaced our complex-valued dynamics of the latent processes $z_{t,j}$ with the models described in the sequel.

H.3.1 PROBABILISTIC KOOPMAN-BASED MODEL

We used real-valued latent variables $\boldsymbol{z}_t$ which are not a-priori independent and whose dynamics are parameterized with a Koopman matrix $\boldsymbol{K}$ and a diagonal noise matrix $\boldsymbol{W}$:

$$\boldsymbol{z}_{t+1} = \boldsymbol{K}\boldsymbol{z}_t + \boldsymbol{W}\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{32}$$

The learned matrix $\boldsymbol{K}$ is not guaranteed to be stable in the absence of additional constraints but in both cases examined the eigenvalues of the learned $\boldsymbol{K}$ were real smaller than one. Long-term predictions were stable and for the Burgers' case in Figure 26 we were also able to reach the true steady state. For the (simpler) Advection-Diffusion example in Figure 25 an incorrect steady state was reached and the predictive quality started to deteriorate after some time-steps. In comparison to our framework, the probabilistic Koopman-based model does not provide a direct separation between slow and fast processes and therefore its interpretability is reduced.

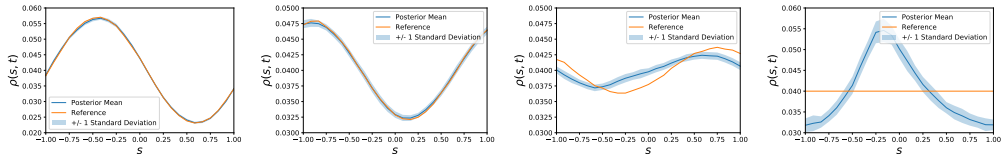


Figure 25: Advection-Diffusion system: Predictions at $t = 0, 80, 160$ and 1000 obtained with the probabilistic Koopman-based model of Equation (32)).

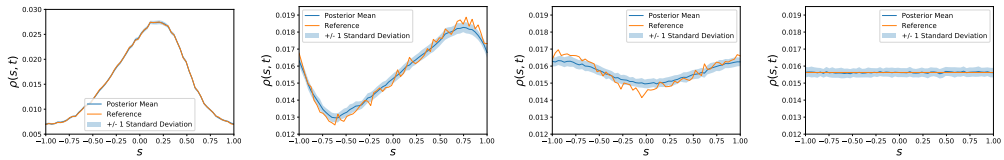


Figure 26: Burgers' system: Predictions at $t = 0, 80, 160$ and 1000 obtained with the probabilistic Koopman-based model of Equation (32)).

H.3.2 NON-PROBABILISTIC KOOPMAN LEARNING

We also used real-valued latent variables $\boldsymbol{z}_t$ with deterministic dynamics which were parameterized as follows:

$$\boldsymbol{z}_{t+1} = \boldsymbol{K}\boldsymbol{z}_t \tag{33}$$

The absence of noise in comparison to Equation (32), led in both cases to an estimate for the Koopman matrix $\boldsymbol{K}$ that did not yield stable predictions (each of the learned $\boldsymbol{K}$ matrices had at least one eigenvalue which was larger than 1). We speculate that the lack of stochasticity made the model less capable of dealing with the information loss. We also note in Figures 27 and 28 that the predictions obtained are, with an exception of a few time-steps, highly inaccurate.

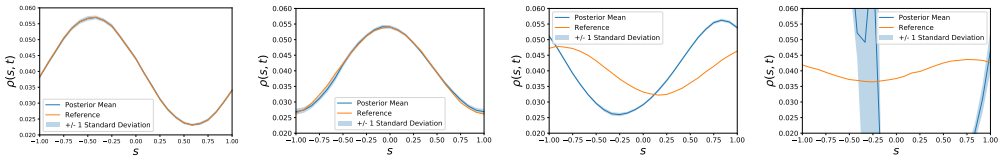


Figure 27: Advection-Diffusion system: Predictions at $t = 0, 20, 80$ and 160 obtained with the deterministic Koopman-based model of Equation (33)).

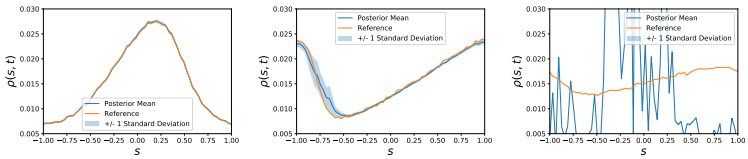


Figure 28: Burgers' system: Predictions at $t = 0, 20$ and 80 obtained with the deterministic Koopman-based model of Equation (33)).

D. Paper D

Semi-supervised Invertible Neural Operators for Bayesian Inverse Problems

Sebastian Kaltenbach[1], Paris Perdikaris[2] and Phaedon-Stelios Koutsourelakis[1,3*]

[1]Professorship of Data-driven Materials Modeling, School of Engineering and Design, Technical University of Munich , Boltzmannstr. 15, Garching, 85748, Germany.
[2]Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, 19104, USA.
[3] Munich Data Science Institute (MDSI - Core member), www.mdsi.tum.de.

*Corresponding author(s). E-mail(s): p.s.koutsourelakis@tum.de;
Contributing authors: sebastian.kaltenbach@tum.de; pgp@seas.upenn.edu;

Abstract

Neural Operators offer a powerful, data-driven tool for solving parametric PDEs as they can represent maps between infinite-dimensional function spaces. In this work, we employ physics-informed Neural Operators in the context of high-dimensional, Bayesian inverse problems. Traditional solution strategies necessitate an enormous, and frequently infeasible, number of forward model solves, as well as the computation of parametric derivatives. In order to enable efficient solutions, we extend Deep Operator Networks (DeepONets) by employing a RealNVP architecture which yields an invertible and differentiable map between the parametric input and the branch-net output. This allows us to construct accurate approximations of the full posterior, irrespective of the number of observations and the magnitude of the observation noise, without any need for additional forward solves nor for cumbersome, iterative sampling procedures. We demonstrate the efficacy and accuracy of the proposed methodology in the context of inverse problems for three benchmarks: an anti-derivative equation, reaction-diffusion dynamics and flow through porous media.

Keywords: Data-driven Surrogates, Invertible Neural Networks, Bayesian Inverse Problems, Semi-supervised Learning

1 Introduction

Nonlinear Partial Differential Equations (PDEs) depending on high- or even infinite-dimensional parametric inputs are ubiquitous in applied physics and engineering and appear in the context of several problems such as model calibration and validation or model-based design/optimization/control. In all these cases, they must be solved repeatedly for different values of the input parameters which poses an often insurmountable obstacle as each of these simulations can imply a significant computational cost. An obvious way to overcome these difficulties is to develop less-expensive but accurate surrogates which can be used on their own or in combination with a reduced number of runs of the high-fidelity, expensive, reference solver. The construction of such surrogates has been based on physical/mathematical considerations or data i.e. input-output

pairs (and sometimes derivatives). Our contribution belongs to the latter category of data-driven surrogates which has attracted a lot of attention in recent years due to the significant progress in the fields of statistical or machine learning (Koutsourelakis et al, 2016; Karniadakis et al, 2021). We emphasize however that unlike typical supervised learning problems in data sciences, in the context of computational physics there are several distinguishing features. Firstly, surrogate construction is by definition a Small (or smallest possible) Data problem. The reason we want to have a surrogate in the first place is to avoid using the reference solver which is the one that generates the training data. Secondly, pertinent problems are rich in domain knowledge which should be incorporated as much as possible, not only in order to reduce the requisite training data but also to achieve higher predictive accuracy particularly in out-of-distribution settings. In the context of Bayesian inverse problems which we investigate in this paper, one does not know a priori where the posterior might be concentrated in the parametric space and cannot guarantee that all such regions will be sufficiently represented in the training dataset. Nevertheless the surrogate learned must be accurate enough in these regions in order to resolve the sought posterior.

Data-driven surrogates which are trained in an offline phase and are subsequently used for various downstream tasks have attracted a lot of attention in recent years (Bhattacharya et al, 2020). Most of these surrogates are constructed by learning a non-linear operator, e.g. a mapping between function spaces and thus between the inputs and the outputs of the PDE, which may depend on additional input parameters. A notable such strategy based on Deep Learning are the Physics-informed Neural Networks (PINNs) (Lagaris et al, 1998; Raissi et al, 2019). An alternative is offered by Deep Operator Networks (DeepONets, (Lu et al, 2021; Wang et al, 2021)), which in contrast to PINNs, not only take the spatial and temporal location as an input but can also account for the dependence of the PDE solution on input parameters such as the viscosity in the Navier-Stokes equation. Furthermore, Fourier Neural Networks (Li et al, 2020) have shown promising results by parametrizing the integral kernel directly in Fourier Space and thus restricting the operator to a convolution. Finally, the Learning Operators with Coupled Attention (LOCA) framework (Kissas et al, 2022) builds upon the well-known attention mechanism that has already shown promising results in natural language processing.

We note that all of the Deep Learning frameworks mentioned fulfill the universal approximation theorem and, under certain conditions, can approximate the non-linear operator to arbitrary accuracy. Another option, is offered by the Optimizing a Discrete Loss (ODIL, Karnakov et al (2022)) framework. It does not rely on Deep Learning and was shown to be faster than PINNs due to the reduced number of tunable parameters but can only approximate the solution on a discrete grid.

Apart from the aforementioned techniques and for time-dependent PDEs in particular, the solution can be approximated by methods based on Koopman-operator theory (Koopman, 1931) which identifies a transformation of the original system that gives rise to linear dynamics (Klus et al, 2018). Nevertheless, these methods (Lee and Carlberg, 2020; Gin et al, 2019; Champion et al, 2019) usually require a large set of reduced-order coordinates or an effective encoder/decoder structure. Especially for physical systems, the restricted dynamics can be endowed with stability and physical, inductive bias (Kaltenbach and Koutsourelakis, 2021; Kalia et al, 2021; Kaltenbach and Koutsourelakis, 2020).

A common limitation of the aforementioned architectures is that they usually learn only the forward operator whereas for the solution of an inverse problem, its inverse would be more useful. In this work, we extend the DeepONet framework by replacing parts of the previously proposed neural-network architecture with an invertible one. To the authors' best knowledge, we are thus presenting the first invertible Neural Operator framework. This allows one to perform both forward and inverse passes with the same neural network and the forward and inverse operators can be learned simultaneously. In particular, we make use of the RealNVP architecture (Dinh et al, 2016) which has an analytical inverse.

Furthermore we make use of both labeled and unlabeled (i.e. only inputs and residuals) training data in a physics-aware, semi-supervised

approach. While the use of labeled training data is straight-forward, unlabeled training data are incorporated by using the governing equations and minimizing the associated residuals, similarly to the physics-informed DeepONet (Wang et al, 2021). Since it is easier and less-expensive to procure unlabeled data in comparison to labeled ones, this leads to significant efficiency gains. Even though our algorithm can produce accurate predictions without any labeled training data and by using only a physics-informed loss, we observe empirically that the addition of labeled training data generally improves the results.
Finally, we show that the proposed invertible DeepONet can be used to very efficiently solve Bayesian inverse problems, i.e. to approximate the whole posterior distribution, without any need for multiple likelihood evaluations and cumbersome iterations as required by alternative inference schemes such as Markov Chain Monte Carlo (MCMC, Beskos et al (2017)) or Sequential Monte Carlo (SMC, Koutsourelakis (2009)) or Stochastic Variational Inference (SVI, Detommaso et al (2018)). In particular, we propose a novel approximation that employs a mixture of Gaussians, the parameters of which are computed semi-analytically. When the proposed Neural Operator framework is trained solely on unlabeled data, this means that we can obtain the solution to the (forward and) inverse problem without ever solving the underlying PDE. While Deep Learning has been successfully applied to inverse problems before (Adler and Öktem, 2017; Ardizzone et al, 2018; Mo et al, 2019), our work differs by making use of a fully invertible, operator-learning architecture which leads to highly efficient approximation of the whole posterior.

The rest of the paper is structured as follows. In section 2 we review the basic elements of invertible neural networks (NNs) and DeepoNets and subsequently illustrate how these can be combined and trained with labeled and unlabeled data. Furthermore we present how the resulting invertible DeepONet can be employed in order to approximate the posterior of a model-based, Bayesian inverse problem at minimal additional cost. We illustrate several features of the proposed methodology and assess its performance in section 3 where it is applied to a reaction-diffusion PDE and a Darcy-diffusion problem. The cost and accuracy of the posterior approximation in the context of pertinent Bayesian inverse problems are demonstrated in section 3.4. Finally, we conclude in section 4 with a summary of the main findings and a discussion on the (dis)advantages of the proposed architecture and potential avenues for improvements.

2 Methodology

We first review some basic concepts of invertible neural networks and DeepONets. We subsequently present our novel contributions which consist of an invertible DeepONet architecture and its use for solving efficiently Bayesian inverse problems.

2.1 Invertible Neural Networks

Neural Networks are in general not invertible which restricts their application in problems requiring inverse operations. Invertibility can be achieved by adding a momentum term (Sander et al, 2021), restricting the Lipschitz-constant of each layer to be smaller than one (Behrmann et al, 2019) or using special building blocks (Dinh et al, 2016). These formulations have primarily been developed for flow-based architectures but we will apply them to operator learning within this work. In particular, we make use of the RealNVP (Dinh et al, 2016) as this architecture enables an analytical inverse which ensures efficient computations. Each RealNVP building block consists of the transformation below which includes two neural networks denoted by $k(.)$ and $r(.)$. Given a D dimensional input $\boldsymbol{x} = \{x_i\}_{i=1}^{D}$ of an invertible layer, the output $\boldsymbol{y} = \{y_i\}_{i=1}^{D}$ is obtained as follows:

$$y_{1:d} = x_{1:d} \tag{1}$$

$$y_{d+1:D} = x_{d+1:D} \circ exp(k(x_{1:d})) + r(x_{1:d}), \tag{2}$$

where $d < D$. Here, $\circ$ is the Hadamard or element-wise product and d is usually chosen to be half of the dimension of the input vector i.e. $d = D/2$.
As only d of the components are updated, the input entries after each building block are permuted, e.g. by reversing the vector, to ensure that after a second building block all of them are modified. Therefore, for $d = D/2$, at least two building blocks are needed in order to modify all entries. We note, that the dimension of the input cannot change and it needs to be identical to the dimension of the output. The two neural networks involved can consist of arbitrary layers as long as

their output and input dimensions are consistent with Equation (2).
The maps defined can be easily inverted which leads to the following equations:

$$x_{1:d} = y_{1:d} \tag{3}$$

$$x_{d+1:D} = (y_{d+1:D} - r(x_{1:d})) \circ exp(-k(x_{1:d})) \tag{4}$$

We note that due to this structure, the Jacobian is lower-triangular and its determinant can be obtained by multiplying the diagonal entries only.

2.2 DeepONets

Before presenting our novel architecture for invertible DeepONets, we briefly review the original DeepONet formulation by Lu et al (2021). DeepONets have been developed to solve parametric PDEs and significantly extend the Physics-Informed Neural Network (PINNs, Raissi et al (2019)) framework as no additional training phase is required if the input parameters of the PDE are changed. We consider a, potentially nonlinear and time-dependent, PDE with an input function $u \in \mathcal{U}$ and solution function $s \in \mathcal{S}$ where $\mathcal{U}, \mathcal{S}$ are appropriate Banach spaces. The former can represent e.g. source terms, boundary or initial conditions, material properties. Let:

$$\mathcal{N}(u,\ s)(\boldsymbol{\xi}) = 0 \tag{5}$$

denote the governing PDE where $\mathcal{N} : \mathcal{U} \times \mathcal{S} \to \mathcal{V}$ is an appropriate differential operator and $\boldsymbol{\xi}$ the spatio-temporal coordinates. Furthermore, let:

$$\mathcal{B}(u,\ s)(\boldsymbol{\xi}) = 0 \tag{6}$$

denote the operator $\mathcal{B} : \mathcal{U} \times \mathcal{S} \to \mathcal{V}$ associated with the boundary or initial conditions. Assuming that the solution s for each $u \in \mathcal{U}$ is unique, we denote with $\mathcal{G} : \mathcal{U} \to \mathcal{S}$ the solution operator that maps from any input u to the corresponding solution s. The goal of DeepONets is to approximate it with an operator $G_{\boldsymbol{\theta}}$ that depends on tunable parameters $\boldsymbol{\theta}$. The latter can yield an approximation to the actual solution at any spatio-temporal point $\boldsymbol{\xi}$ which we denote by $G_{\boldsymbol{\theta}}(\boldsymbol{\xi})$. It is based on a separated representation (Lu et al, 2021)[1]:

$$G_{\boldsymbol{\theta}}(u)(\boldsymbol{\xi}) = \sum_{j=1}^{Q} b_j \left(\underbrace{u(\boldsymbol{\eta}_1), ..., u(\boldsymbol{\eta}_F)}_{\boldsymbol{u}} \right) t_j(\boldsymbol{\xi}) \tag{7}$$

and consists of the so-called *branch network* whose terms b_j depend on the values of the input function u at F fixed spatio-temporal locations[2] $\{\boldsymbol{\eta}_l\}_{l=1}^{F}$ which we summarily denote with the vector $\boldsymbol{u} \in \mathbb{R}^F$, and the so-called *trunk network* whose terms t_j depend on the spatio-temporal coordinates $\boldsymbol{\xi}$ (see Figure 2.2). Both networks have trainable weight and bias parameters which we denote collectively by $\boldsymbol{\theta}$. We emphasize that, once trained, the DeepONet can provide predictions of the solution at *any spatio-temporal location* $\boldsymbol{\xi}$, a feature that is very convenient in the context of inverse problems as the same DeepONet can be used for solving problems with different sets of observations.

We note that in the next section, we will use a vectorized formulation of Equation (7) and process various spatio-temporal coordinate datapoints together as this is needed to ensure invertibility of the DeepONet. Labeled data can be used for training which consist of pairs of u and corresponding solutions $s = \mathcal{G}(u)$ evaluated at certain spatio-temporal locations. Unlabeled training data (i.e. only inputs) can also be employed in a physics-informed approach as introduced by Wang et al (2021), by including the governing PDE in Equation (5) in an additional loss term as discussed section 2.4.

2.3 Invertible DeepONets

The invertible RealNVP introduced in section 2.1 is employed exclusively on the branch network i.e. we assume that:

$$D = F = Q \tag{8}$$

and the input $\boldsymbol{x}$ of section 2.1 is the vector $\boldsymbol{u} \in \mathbb{R}^F$ containing the values of the PDE-input at $D = F$

[1] We omit the NN parameters $\boldsymbol{\theta}$ on the right-hand side in order to simplify the notation.

[2] These points are usually chosen to be uniformly distributed over the entire domain, but it is also possible to increase their density in certain areas, e.g. with high variability.

Fig. 1 (Left) Classical DeepONet (Lu et al, 2019) and (Right) proposed Invertible DeepONet architecture

spatio-temporal locations whereas the output $\boldsymbol{y}$ of section 2.1 is now the $D = Q$ values of the branch net $\boldsymbol{b} = [b_1, \ldots, b_Q]^T \in \mathbb{R}^D$. We note that this restriction regarding the equality of the dimension of the input $\boldsymbol{u}$ and the output of the branch network $\boldsymbol{b}$ is due to the use of an invertible architecture. As a consequence, the dimension of the trunk-network output i.e. $\{t_j(\boldsymbol{\xi})\}_{j=1}^{Q}$ is also the same as the dimension of $\boldsymbol{u}$. This requirement does not reduce the generality of the methodology advocated as Q is a free parameter in the definition of the operator $G_{\boldsymbol{\theta}}$ in Equation (7).

In view of the inverse problems we would like to address, we consider K spatio-temporal locations, $\{\boldsymbol{\xi}_k\}_{k=1}^{K}$ and we denote with $\boldsymbol{s} \in \mathbb{R}^K$ the vector containing the PDE-solution's values at these locations i.e. $\boldsymbol{s} = [s(\boldsymbol{\xi}_1), \ldots, s(\boldsymbol{\xi}_K)]^T$. Finally we denote with $\boldsymbol{Y}$ the $K \times D$ matrix constructed by the values of the trunk network outputs at the aforementioned locations, i.e.:

$$\boldsymbol{Y} = \begin{bmatrix} t_1(\boldsymbol{\xi}_1) & \ldots & t_D(\boldsymbol{\xi}_1) \\ \ldots & & \ldots \\ t_1(\boldsymbol{\xi}_K) & \ldots & t_D(\boldsymbol{\xi}_K) \end{bmatrix}. \tag{9}$$

As a result of Equation (7), we can write that:

$$\boldsymbol{s} = \boldsymbol{Y}\boldsymbol{b} \tag{10}$$

As the matrix $\boldsymbol{Y}$ is in general non-invertible, one can determine $\boldsymbol{b}$ given $\boldsymbol{s}$ by solving a least-squares problem, i.e.:

$$\min_{\boldsymbol{b}} \ \|\boldsymbol{s} - \boldsymbol{Y}\boldsymbol{b}\|_2^2 \tag{11}$$

or a better-behaved, regularized version thereof:

$$\min_{\boldsymbol{b}} \ \|\boldsymbol{s} - \boldsymbol{Y}\boldsymbol{b}\|_2^2 + \epsilon\|\boldsymbol{b}\|_2^2 \tag{12}$$

where a small value is generally sufficient for the regularization parameter $\epsilon << 1$. We note that given $\boldsymbol{s}$ and once $\boldsymbol{b}$ has been determined by solving Equation (11) or Equation (12), we can make use of the invertibility of the branch net in order to obtain the input vector $\boldsymbol{u}$. While other approaches are possible in order to determine $\boldsymbol{b}$, we recommend using the regularized, least-squares formulation, as this led to robust results in our experiments. It is nevertheless important to use the same method during training and when deterministic predictions are sought, since different methods can lead to different $\boldsymbol{b}$'s for the same $\boldsymbol{s}$. We note that in the proposed method for the solution of Bayesian inverse problems (see Section 2.5), no use of Equation (12) is made except for the training of the DeepONet (see Section 2.4).

For the ensuing equations we denote the forward map implied by Equation (10) as:

$$\boldsymbol{s} = \mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{u}, \boldsymbol{Y}) \tag{13}$$

and the inverse obtained by the two steps described above as:

$$\boldsymbol{u} = \mathcal{I}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{Y}) \tag{14}$$

where we explicitly account for the NN parameters $\boldsymbol{\theta}$.

2.4 A Semi-supervised Approach for Invertible DeepONets

As mentioned earlier and in order to train the invertible DeepONet proposed, i.e. to find the optimal values for the parameters $\boldsymbol{\theta}$, we employ both labeled (i.e pairs of PDE-inputs u and PDE-outputs s) and unlabled data (i.e. only PDE-inputs u) in combination with the governing equations. The loss function L employed is

therefore decomposed into two parts as[3]:

$$L = L_{labeled} + L_{unlabeled} \tag{15}$$

The first term $L_{labeled}$ pertains to the labeled data and is further decomposed as:

$$L_{labeled} = L_{l,forward} + L_{l,inverse} \tag{16}$$

Without loss of generality and in order to keep the notation as simple as possible we assume that N_l pairs of labeled data are available, each of which consists of the values of the PDE-input u at D locations which we denote with $\boldsymbol{u}^{(i)} \in \mathbb{R}^D,\ i = 1, \dots N_l$ and the values of the PDE-output at K spatio-temporal locations which we denote with $\boldsymbol{s}^{(i)} \in \mathbb{R}^K,\ i = 1, \dots N_l$. If the $K \times D$ matrix $\boldsymbol{Y}$ is defined as in Equation (9) and in view of the forward (Equation (13)) and inverse (Equation (14)) maps defined earlier, we write:

$$L_{l,forward} = \frac{1}{N_l} \sum_{i=1}^{N_l} \|\boldsymbol{s}^{(i)} - \mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{u}^{(i)}, \boldsymbol{Y})\|_2^2 \tag{17}$$

and:

$$L_{l,inverse} = \frac{1}{N_l} \sum_{i=1}^{N_l} \|\boldsymbol{u}^{(i)} - \mathcal{I}_{\boldsymbol{\theta}}(\boldsymbol{s}^{(i)}, \boldsymbol{Y})\|_2^2. \tag{18}$$

By employing both loss terms, the NN parameters $\boldsymbol{\theta}$ can balance the accuracy of the approximation in both maps.

Furthermore and assuming N_u PDE-inputs are available each of which is evaluated at D spatio-temporal points $\{\boldsymbol{\xi}^{(l)}\}_{l=1}^D$ with $\boldsymbol{u}^{(i)} \in \mathbb{R}^D$ denoting these values, we express the $L_{unlabeled}$ loss term as:

$$L_{unlabeled} = L_{BC} + L_{res} + L_{u,inverse}. \tag{19}$$

The first L_{BC} and second L_{res} terms are physics-informed Wang et al (2021) and account for the residuals in the boundary (and/or initial) conditions and the governing PDE respectively. In the case of L_{BC} we select N_B (uniformly distributed) points along the boundary, say $\boldsymbol{\xi}_B^{(j)}, l = 1, \dots, N_B$.

[3]All loss functions depend on $\boldsymbol{\theta}$ which we omit in order to simplify the notation.

Then, in view of Equation (6), we employ:

$$L_{BC} = \frac{1}{N_u N_B} \sum_{i=1}^{N_u} \sum_{l=1}^{N_B} \|\mathcal{B}(u^{(i)}, G_{\boldsymbol{\theta}}(u^{(i)}))(\boldsymbol{\xi}_B^{(l)})\|_2^2 \tag{20}$$

In the interior of the problem domain and in view of Equation (5), we employ a loss:

$$L_{res} = \frac{1}{N_u\ N_{res}} \sum_{i=1}^{N_u} \sum_{l=1}^{N_{res}} \|\mathcal{N}(u^{(i)}, G_{\boldsymbol{\theta}}(u^{(i)})(\boldsymbol{\xi}^{(l)})\|_2^2 \tag{21}$$

which involves N_{res} collocation points.

The third term $L_{u,inverse}$ pertains to the forward and inverse maps in Equations (13), (14) and can be expressed as:

$$L_{u,inverse} = \frac{1}{N_u} \sum_{i=1}^{N_u} \|\boldsymbol{u}^i - \mathcal{I}_{\boldsymbol{\theta}}(\mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{u}^{(i)}, \boldsymbol{Y}), \boldsymbol{Y}))\|_2^2 \tag{22}$$

where the matrix $\boldsymbol{Y}$ is defined as in Equation (9).

The minimization of the combined loss L, with respect to the NN parameters $\boldsymbol{\theta}$ of the branch and trunk network, is performed with stochastic gradient descent and the ADAM (Kingma and Ba, 2014) scheme in particular. Gradients of the loss were computed using the automatic differentiation tools of the JAX library (Bradbury et al, 2018). We finally note that the D spatioemporal locations need not be the same nor do they need to be equal in number in all data instances as assumed in the equations above. In such cases the vector of the observables and the matrices $\boldsymbol{Y}$ involved would differ which would further complicate the notation but the same DeepONet parameters $\boldsymbol{\theta}$ would appear in all terms.

2.5 Invertible DeepONets for Bayesian inverse problems

In this section we discuss how the invertible DeepONets proposed and trained as previously discussed, can be used to efficiently approximate the solution of a Bayesian inverse problem in the presence of, potentially noisy, observations as well as prior uncertainty about the unknowns. A central role is played by the readily available invertible map which the RealNVP architecture affords. In particular, let $\hat{\boldsymbol{s}} \in \mathbb{R}^K$ denote a vector of noisy observations of the PDE-solution at certain K

spatio-temporal locations. These are assumed to be related to the PDE-solution's values at these locations, denoted summarily by $\boldsymbol{s} \in \mathbb{R}^K$, as follows:

$$\hat{\boldsymbol{s}} = \boldsymbol{s} + \sigma\ \boldsymbol{\eta}, \qquad \boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{23}$$

where σ^2 is the variance of the observational noise. This in turn defines a conditional density (likelihood) $p(\hat{\boldsymbol{s}} \mid \boldsymbol{s})$:

$$p(\hat{\boldsymbol{s}} \mid \boldsymbol{s}) = \mathcal{N}(\hat{\boldsymbol{s}} \mid \boldsymbol{s}, \sigma^2 \boldsymbol{I}). \tag{24}$$

In the context of a Bayesian formulation and given the implicit dependence of the PDE-output $\boldsymbol{s}$ on $\boldsymbol{u}$, the likelihood would be combined with the a prior density $p_u(\boldsymbol{u})$ on the PDE-inputs in order to define the sought posterior:

$$p(\boldsymbol{u} \mid \hat{\boldsymbol{s}}) \propto p(\hat{\boldsymbol{s}} \mid \boldsymbol{s})\ p_u(\boldsymbol{u}).$$

Even if the trained DeepONet were used to infer $p(\boldsymbol{u} \mid \hat{\boldsymbol{s}})$ (e.g. using MCMC) several evaluations would be needed especially if the dimension of $\boldsymbol{u}$ was high. In the sequel we demonstrate how one can take advantage of the invertible NN architecture in order to obtain a semi-analytic approximation of the posterior in the form of a mixture of Gaussians and by avoiding iterative algorithms like MCMC altogether.

We note first that by combining the likelihood with Equation (10), we can write it in terms of the $D-$dimensional, branch-network output vector $\boldsymbol{b}$ as:

$$p(\hat{\boldsymbol{s}} \mid \boldsymbol{b}) = \mathcal{N}(\hat{\boldsymbol{s}} \mid \boldsymbol{Y}\ \boldsymbol{b}, \sigma^2 \boldsymbol{I}). \tag{25}$$

Since $\boldsymbol{u} \in \mathbb{R}^D$ is related to $\boldsymbol{b}$ through the invertible RealNVP $\boldsymbol{b}_{NN} : \mathbb{R}^D \to \mathbb{R}^D$, we can also obtain a prior density $p_b(\boldsymbol{b})$ on $\boldsymbol{b}$ as:

$$p_b(\boldsymbol{b}) = p_u(\boldsymbol{b}_{NN}^{-1}(\boldsymbol{b}))\ J(\boldsymbol{b}) \tag{26}$$

where $\boldsymbol{b}_{NN}^{-1}$ denotes the inverse and $J(\boldsymbol{b}) = |\frac{\partial \boldsymbol{b}_{NN}^{-1}}{\partial \boldsymbol{b}}|$ is the determinant of its Jacobian. The latter, as mentioned in section 2.1, is a triangular matrix and its determinant can be readily computed at a cost $\mathcal{O}(D)$.

We choose not to directly operate with the prior $p_b(\boldsymbol{b})$, but construct an approximation $p_{b,G}(\boldsymbol{b})$ to this in the form of a mixture of $D-$dimensional Gaussians as this allows as to facilitate subsequent steps in finding the posterior. In particular:

$$p_{b,G}(\boldsymbol{b}) = \sum_{m=1}^{M} w_j\ \mathcal{N}(\boldsymbol{b} \mid \boldsymbol{m}_{b,m}, \boldsymbol{S}_{b,m}) \tag{27}$$

where M denotes the number of mixture components and $\boldsymbol{m}_{b,m}$, $\boldsymbol{S}_{b,m}$ the mean vector and covariance matrix of the m^{th} component respectively. Such an approximation can be readily computed, e.g. using Variational Inference (Wainwright and Jordan, 2008) and *without* any forward or inverse model evaluations by exploiting the fact that samples from p_b can be readily drawn using ancestral sampling i.e. by drawing samples of $\boldsymbol{u}$ from p_u and propagating those with $\boldsymbol{b}_{NN}$. We note that finding this representation can become more diffucult in case M is large but the complexity of the algorithms involved in general scales linearly with M (Bishop and Nasrabadi, 2006).

By combining the (approximate prior) $p_{b,G}(\boldsymbol{b})$ above with the Gaussian likelihood $p(\hat{\boldsymbol{s}} \mid \boldsymbol{b})$ of Equation (25) we obtain an expression for the posterior $\tilde{p}(\boldsymbol{b} \mid \hat{\boldsymbol{s}})$ using Bayes' theorem:

$$\tilde{p}(\boldsymbol{b} \mid \hat{\boldsymbol{s}}) \propto p(\hat{\boldsymbol{s}} \mid \boldsymbol{b}) p_{b,G}(\boldsymbol{b}) \tag{28}$$

Due to the conjugacy of prior and likelihood, we can directly conclude that the (approximate) posterior is also a mixture of Gaussians (Bishop and Nasrabadi, 2006). Therefore, using expressions for the aforementioned likelihood/prior pair, we obtain a closed-form posterior $\tilde{p}(\boldsymbol{b} \mid \hat{\boldsymbol{s}})$ on $\boldsymbol{b}$ of the form:

$$\tilde{p}(\boldsymbol{b} \mid \hat{\boldsymbol{s}}) = \sum_{m=1}^{M} \tilde{w}_j\ \mathcal{N}(\boldsymbol{b} \mid \boldsymbol{\mu}_{b,m}, \boldsymbol{C}_{b,m}) \tag{29}$$

where the mean $\boldsymbol{\mu}_{b,m}$ and covariance $\boldsymbol{C}_{b,m}$ of each mixture component can be computed as:

$$\begin{aligned} \boldsymbol{C}_{b,m}^{-1} &= \sigma^{-2} \boldsymbol{Y}^T \boldsymbol{Y} + \boldsymbol{S}_{b,m}^{-1} \\ \boldsymbol{C}_{b,m}^{-1} \boldsymbol{\mu}_{b,m} &= \sigma^{-2} \boldsymbol{Y}^T \hat{\boldsymbol{s}} + \boldsymbol{S}_b^{-1} \boldsymbol{m}_{b,m} \end{aligned} \tag{30}$$

The weights $\tilde{w}_m$ ($\sum_{m=1}^{M} \tilde{w}_m = 1$) would be proportional to:

$$\tilde{w}_m \propto w_m \mid \boldsymbol{D}_m \mid^{-1/2} \exp(-\tfrac{1}{2}(\hat{\boldsymbol{s}} - \boldsymbol{Y}\boldsymbol{m}_{b,m})^T \boldsymbol{D}_m^{-1}(\hat{\boldsymbol{s}} - \boldsymbol{Y}\boldsymbol{m}_{b,m})) \tag{31}$$

where:

$$\boldsymbol{D}_m = \sigma^2 \boldsymbol{I} + \boldsymbol{Y} \boldsymbol{S}_{b,m} \boldsymbol{Y}^T \tag{32}$$

Therefore inference tasks on the sought bsu can be readily carried out by sampling $\boldsymbol{b}$ from the mixture-of-Gaussians posterior above and propagating those samples through the inverse map $\boldsymbol{b}_{NN}^{-1}$ to obtain $\boldsymbol{u}$-samples. We note that by employing a mixture of Gaussians with sufficient components M, one can approximate with arbitrary accuracy any non-Gaussian density as well as capture multimodal posteriors, a task that is extremely cumbersome with standard, Bayesian inference schemes (Franck and Koutsourelakis, 2017).

3 Numerical Illustrations

We applied the proposed framework to three examples, i.e. the antiderivate operator , a reaction-diffusion PDE as well as a Darcy-type elliptic PDE. In each of these cases, we report the relative errors of forward and inverse maps (on test data) when trained with varying amounts of labeled and unlabeled training data. For the reaction-diffusion PDE and the Darcy-type elliptic PDE, we also use the proposed invertible-DeepONet-surrogate to solve pertinent Bayesian inverse problems. The code for the aforementioned numerical illustrations is available here[4]. In Table 1, we summarize the most important dimensions for each of the following examples, namely D: the dimension of the PDE-input, K: the dimension of the observed PDE-output, N_l: number of labeled data (Equations (17), (18)), N_u: the number of unlabeled data (e.g. Equation (22)), N_{res}: the number of interior collocation points (Equation (21)) and N_{BC} the number of boundary collocation points (Equation (20)).

3.1 Anti-derivative Operator

As a first test case we considered the antiderivative operator on the interval $\xi \in [0, 1]$ with:

$$\frac{ds(\xi)}{d\xi} = u(\xi) \text{ with } s(0) = 0 \tag{33}$$

i.e. when the input u corresponds to the right-hand-side of this ODE and the operator $\mathcal{G}(u)$ that we attempt to approximate is simply the integral operator $\mathcal{G}(u)(\xi) = \int_0^\xi u(t)\, dt$. We generated $N_u = 10000$ unlabeled training data by sampling inputs u from a Gaussian process with zero mean and exponential quadratic covariance kernel with a length scale $\ell = 0.2$. Their values at the same $D = 100$ uniformly-distributed locations in $[0, 1]$ were recorded. We subsequently randomly choose $N_{res} = 200$ collocation points to evaluate the residuals (see Equation (21)).

Moreover, we used up to $N_l = 10000$ labeled training data, for which the inputs were generated as for the unlabeled training data, and the outputs were obtained by solving the ODE above and evaluating it at $K = 200$ randomly chosen points. We trained the invertible DeepONet on $N_u = 10000$ unlabeled training data with a batch size of 100. In each batch we added 1, 10 or 100 labeled training data points per batch (i.e. $N_l = 100, 1000, 10000$ respectively in Equations (17), (18)). A minimum of one labeled datapoint is required in order to set the initial condition correctly as we did not enforce this separately in the unlabeled loss part. With regards to the architecture of the networks used, we employed a MLP with four layers and 100 neurons each for the trunk network and 6 RealNVP building blocks for the branch network which were parametrized by a two-layered MLP. Variations around these values in the number of neurons, layers were also explored (in the subsequent examples as well) and did not impact significantly the performance.

Using the ADAM optimizer and an initial learning rate of 10^{-3}, we run the model training for 4×10^4 iterations with an exponential learning rate decay with rate 0.9 every 1000 iterations. As test data, we used 1000 new (i.e. not included in the training data) input-output pairs and compared the predicted forward and inverse solutions with the actual ones. The results obtained in terms of the relative errors are summarized in Table 2.

The error values indicate that both the forward as well as the inverse maps are very well approximated by the proposed invertible DeepONet. The addition of more labeled training data results in even lower errors especially for the inverse map for which the relative error is decreased from almost $\sim 4\%$ to $\sim 2\%$.

In order to visualize the results we plot for four

[4] URL https://github.com/pkmtum/Semi-supervised_Invertible_Neural_Operators

	section 3.1	section 3.2	section 3.3	section 3.4.1	section 3.4.2
D	100	100	64	100	64
K	200	200	3844	$25, 100$	$1922, 3844$
N_l	$10^2, 10^3, 10^4$	$0, 500, 5000$	10^3	500	5000
N_u	10^4	5000	10^3	10^4	5000
N_{res}	200	200	3844	200	200
N_{BC}	-	300	-	-	-

Table 1 Main dimensions for each numerical illustration

labeled data [%]	1	10	100
relative error s (forward map)	0.0152 ± 0.0151	0.00791 ± 0.00799	0.00728 ± 0.00797
relative error u (inverse map)	0.0371 ± 0.0241	0.034 ± 0.024	0.0215 ± 0.0153

Table 2 Relative test errors and their standard deviations depending on the amount of labeled training data for the anti-derivative operator. The percentage of labeled data is the amount of data used in comparison to unlabeled training data, e.g. in the 10% case we used ten times more unlabeled training data whereas in the 100% case the amount of labeled and unlabeled training data was the same.

randomly-chosen test cases the predictions (when trained with 10% labeled data) of both the forward (Figure 2) and inverse (Figure 3) operator. In all cases, the predictions are indistinguishable from the reference functions.

In Appendix A we include additional results for this problem with varying amounts of unlabeled and labeled training data in order to further show their influence.

3.2 Reaction-Diffusion dynamics

The second illustrative example involves the reaction-diffusion equation on the space-time domain $\boldsymbol{\xi} = (x, t) \in [0, 1] \times [0, 1]$:

$$\frac{\partial s}{\partial t} = D_s \frac{\partial^2 s}{\partial x^2} + ks^2 + u(x) \tag{34}$$

Here, $D_s = 0.01$ is the diffusion constant, $k = 0.01$ the reaction rate and the source-term $u(x)$ is chosen to be the PDE-input. We used zero values as initial conditions and boundary conditions. We generated random source terms by sampling from a Gaussian process with zero mean and and exponential quadratic covariance kernel with a length scale $\ell = 0.2$ which were then evaluated at $D = 100$ uniformly distributed points over $[0, 1]$. The PDE was subsequently solved using an implicit Finite-Difference scheme and evaluated at 200 randomly chosen points to generate the labeled training data.

We trained our model with $N_u = 5000$ unlabeled data which were processed in batches of 100 samples and to which varying amounts of labeled data were added. Since for this problem the boundary conditions were enforced separately, the amount of labeled training data used could also be zero. All unlabeled training data points were evaluated at $N_{res} = 200$ randomly selected collocation points. With regards to the network architecture, we employed a MLP with five layers and 100 neurons each for the trunk network and 3 RealNVP building blocks for the branch network which were parametrized by a three-layered MLP. Using the ADAM optimizer and an initial learning rate of 10^{-3}, we run the model training for 12×10^4 iterations with an exponential learning rate decay with rate 0.9 every 2000 iterations. For our test dataset, we generated 1000 new (unseen) source terms u and corresponding solutions s. A summary of the relative errors obtained is contained in Table 3.

We note that again for all three settings we achieve very low error rates, which decrease as the amount of labeled training data increases. In Figure 4 and 5 we show the predictions (trained with 500 i.e. 10% labeled data) of both forward and inverse map for three randomly chosen test cases.

3.3 Flow through porous media

In the final example we considered the Darcfy-flow elliptic PDE in the two-dimensional domain

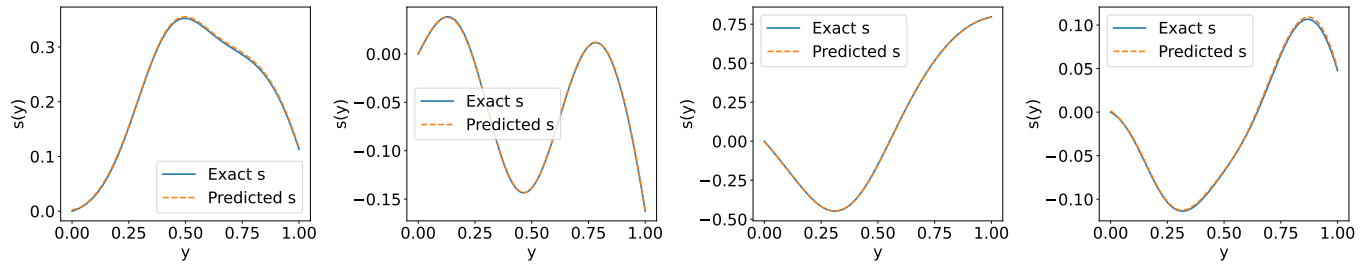


Fig. 2 Forward map - Comparison of the true PDE-output/solution s (given a PDE-input u) with the one predicted by the proposed invertible DeepONet and for the anti-derivative operator

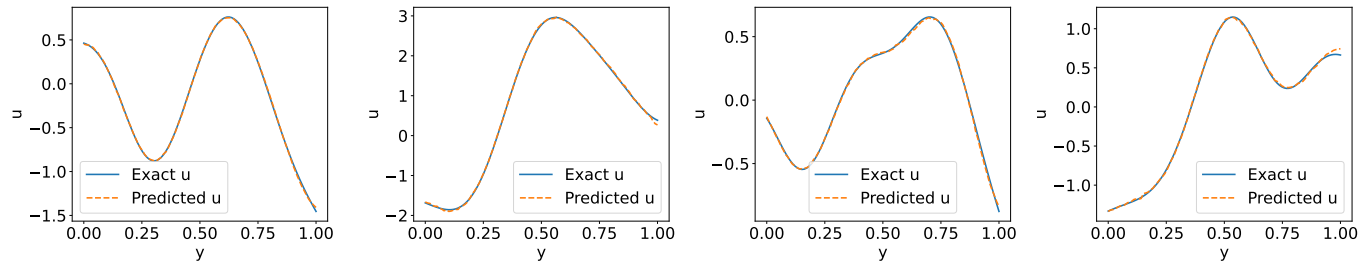


Fig. 3 Inverse map - Comparison of the true PDE-input u (given the PDE-output/solution s) with the one predicted by the proposed invertible DeepONet and for the anti-derivative operator

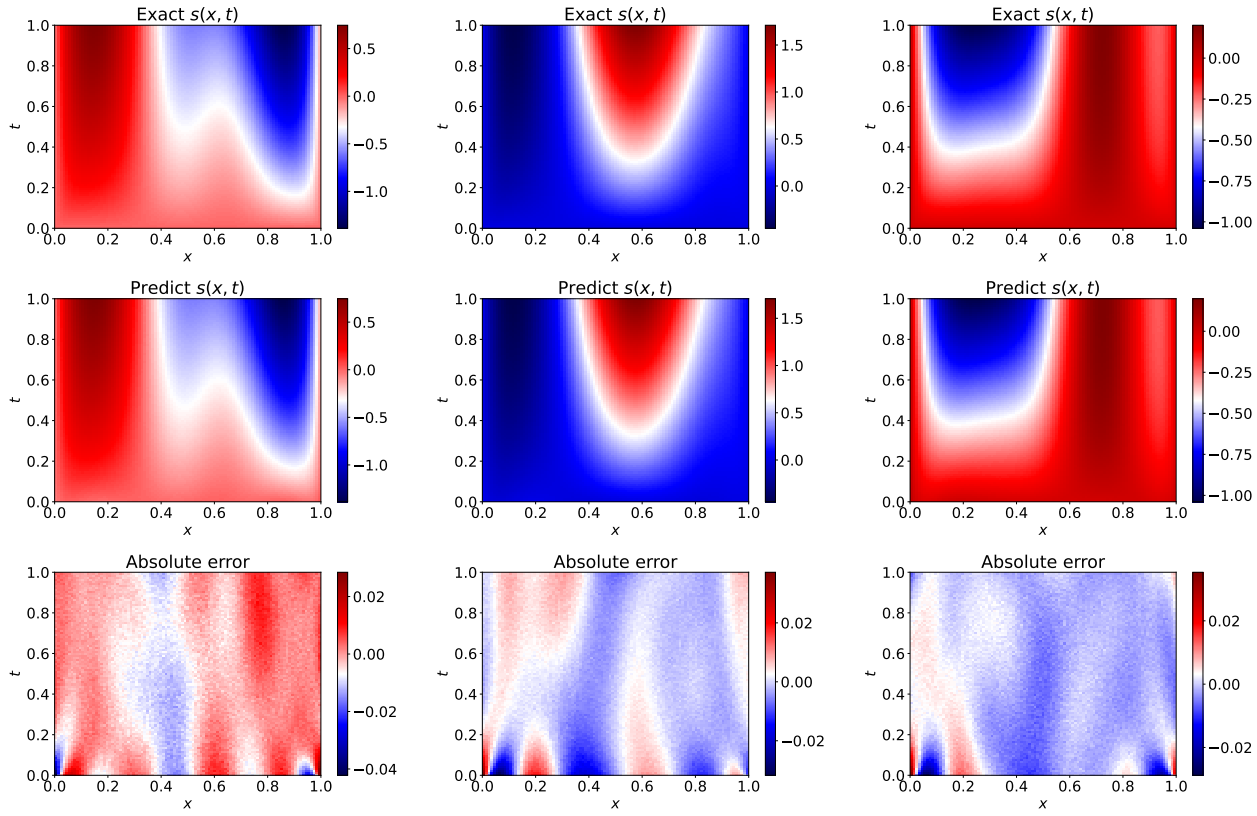


Fig. 4 Forward map - Comparison of the true PDE-output/solution s (given the PDE-input u) with the one predicted by the proposed invertible DeepONet and for Reaction-Diffusion PDE.

labeled data [%]	0	10	100
relative error for s	0.00925 ± 0.00492	0.0105 ± 0.00519	0.00813 ± 0.00445
relative error for u	0.024 ± 0.01021	0.0184 ± 0.00578	0.0162 ± 0.00592

Table 3 Relative errors on test data depending on the amount of labeled training data for the reaction-diffusion case.

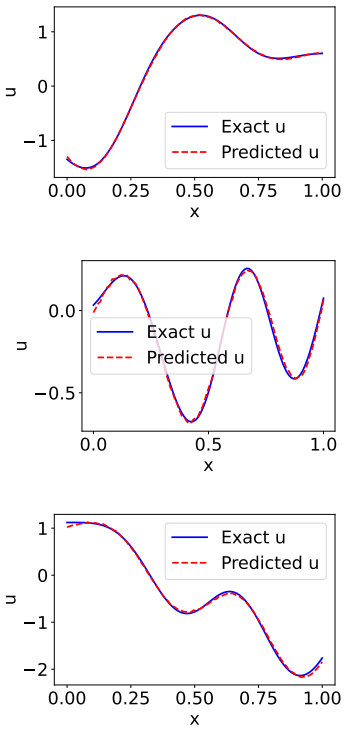


Fig. 5 Inverse map - Comparison of the true PDE-input u (given the PDE-output/solution s) with the one predicted by the proposed invertible DeepONet and for Reaction-Diffusion PDE.

$$\boldsymbol{\xi} = (x_1, x_2) \in [0,1]^2$$

$$\nabla \cdot (u(\boldsymbol{\xi}) \nabla s(\boldsymbol{\xi})) = 10 \tag{35}$$

where the PDE-input u corresponds to the permeability field. We assumed zero values for the solution s along all boundaries which we a-priori incorporated in our operator approximation by multiplying the DeepONet expansion in Equation (7) with the polynomial $x_1(1-x_1)\, x_2(1-x_2)$. We used $N_u = 1000$ unlabeled training data points with $N_{res} = 3844$ collocation points (Equation (21)) during training and added either no labeled training data at all (i.e. $N_l = 0$) or $N_l = 1000$. In order to obtain the latter we solved Equation (35) with the Finite Element library FEniCS (Logg et al, 2012) on a 128×128 mesh with linear elements and evaluated the solution at 3844 regularly distributed points. We represent the PDE-input u as follows[5]:

$$\begin{aligned} \ln(u) = \sum_{f_1=1}^{4} \sum_{f_2=1}^{4} & c_{f_1,f_2,1} \sin(f_1 x_1) \cos(f_2 x_2) \\ & + c_{f_1,f_2,2} \sin(f_1 x_1) \sin(f_2 x_2) \\ & + c_{f_1,f_2,3} \cos(f_1 x_1) \sin(f_2 x_2) \\ & + c_{f_1,f_2,4} \cos(f_1 x_1) \cos(f_2 x_2) \end{aligned} \tag{36}$$

using 64 feature functions and corresponding coefficients c. In order to generate the training data, we sampled each of the aforementioned 64 coefficients from a uniform distribution in $[0,1]$. In this example the 64-dimensional vector of the c's serves as the input in the branch network (i.e. $D = 64$). With the help of the c's and of Equation (36), one can reconstruct the full permeability field.

With regards to the network architecture, we employed a MLP with five layers and 64 Neurons each for the trunk network and 3 RealNVP building blocks for the branch network which were parametrized by a three-layered MLP. Using the ADAM optimizer and an initial learning rate of 10^{-3}, we run the model training for 10^5 iterations with an exponential learning rate decay with rate 0.9 every 2000 iterations. We tested the trained model on 2500 unseen test data and obtained the results in Table 4. As in the previous examples, the inclusion of labeled data significantly improves the predictive accuracy of the trained model. For the case without data the predictive accuracy of the forward map is slightly lower but the accuracy in the inverse map is comparably low. The addition of labeled data improves the predictive accuracy for both maps.

labeled data [%]	0	100
relative error for s	0.0134 ± 0.00509	0.0245 ± 0.0108
relative error for u	0.235 ± 0.137	0.0566 ± 0.0198

Table 4 Relative errors on test data depending on the amount of labeled training data for the Darcy example with feature coefficients as inputs.

[5] We employ this expansion for the logarithm of u in order to ensure that the resulting permeability field is positive

In Figure 6 we compare the reference solution for two illustrative test cases with the the forward map learned with labeled training data. As suggested by the cumulative results in Table 4 the two predictions are very close to the reference and the accuracy is very high. In Figures 7 (without labeled training) and 8 (with labeled training) the results for two illustrative inverse test cases are shown.

While locally the error can be significant, the main characteristics of the PDE-input field u can be captured.

We discuss in the next section the case where the input permeability field u is not represented with respect to some feature functions but rather as a discretized continuous field.

3.3.1 Coarse-grained (CG) input parameters

In this sub-case, we modeled the permeability field u with an exponentiated (to ensure positivity) Gaussian Process with mean zero and exponential quadratic covariance with length scale $\ell = 0.1$. The PDE was then again solved on a 128×128 FE mesh and the values of the solution s were assumed to be observed at 3844 regularly distributed points. We moreover sub-sampled the generated PDE input on a regular 8×8 grid and its $D = 64$ values represented the branch network input $\boldsymbol{u}$. We generated $N_u = 1000$ unlabeled fields u in total and used $N_{res} = 3844$ collocation points (Equation (21)) during training. We also trained the model with $N_l = 1000$ labeled training data.

The results obtained can be found in Table 5. The test data in this table consists of 2500 unseen, discretized, permeability fields and their respective solutions. The error rates are computed with respect to the coarse-grained reference input. As in the previous setting, we observe a significant improvement in the accuracy of the inverse map when labeled data are used in training. In Figure 9 we compare the reference solution for two illustrative test cases with the the forward map learned with labeled training data. As suggested by the cumulative results in Table 5 the two predictions are very close to the reference and the accuracy is very high.

labeled data [%]	0	100
relative error for s	0.0164 ± 0.00712	0.0164 ± 0.00748
relative error for u	0.121 ± 0.041	0.0656 ± 0.0168

Table 5 Relative errors on test data depending on the amount of labeled training data for the Darcy example with coarse-grained input parameters

In Figures 10 (without labeled training) and 11 (with labeled training) the results for two illustrative inverse test cases are shown. We note again that the main features of the PDE-input's spatial variability are captured, despite the presence of localized errors.

3.4 Bayesian Inverse Problems

In this section we demonstrate the utility of the invertible DeepONet proposed in the solution of Bayesian inverse problems and in obtaining accurate approximations of the posterior without any need for additional reference model runs nor for any costly and asymptotically-exact sampling. For each of the examples considered, only one observed output $\hat{\boldsymbol{s}}$ was assumed to be given. The variance of the observational noise σ^2 was assumed to be given although this could readily be inferred, especially if a conjugate inverse-Gamma prior was used for it. In this manner, any deviations from the actual posterior could be attributed to inaccuracies of the DeepONet-based surrogate. Errors due to the approximation of the prior with a mixture of Gaussians as in Equation (27) can be made arbitrarily small by increasing the number of mixture components M.

3.4.1 Reaction-Diffusion dynamics

We employed the trained model of the reaction-diffusion system (with 10% labeled training data), in combination with the formulation detailed in section 2.5 for approximating the posterior. We use a prior $p_u(\boldsymbol{u})$ arising from the discretization of Gaussian Process with zero mean and exponential quadratic covariance kernel with a length scale $\ell = 0.2$. For the Gaussian mixture models involved for the prior and subsequently the posterior on $\boldsymbol{b}$ we used two components i.e. $M = 2$ in Equations (27), (29). The results can be seen in the following Figures. The obtained posterior encapsulates the true parameter input for all three cases.

In Figure 12 we used test cases with 100 observed solution data points for each parameter input and a noise level of $\sigma^2 = 0.001$ (see Equation (23)). In

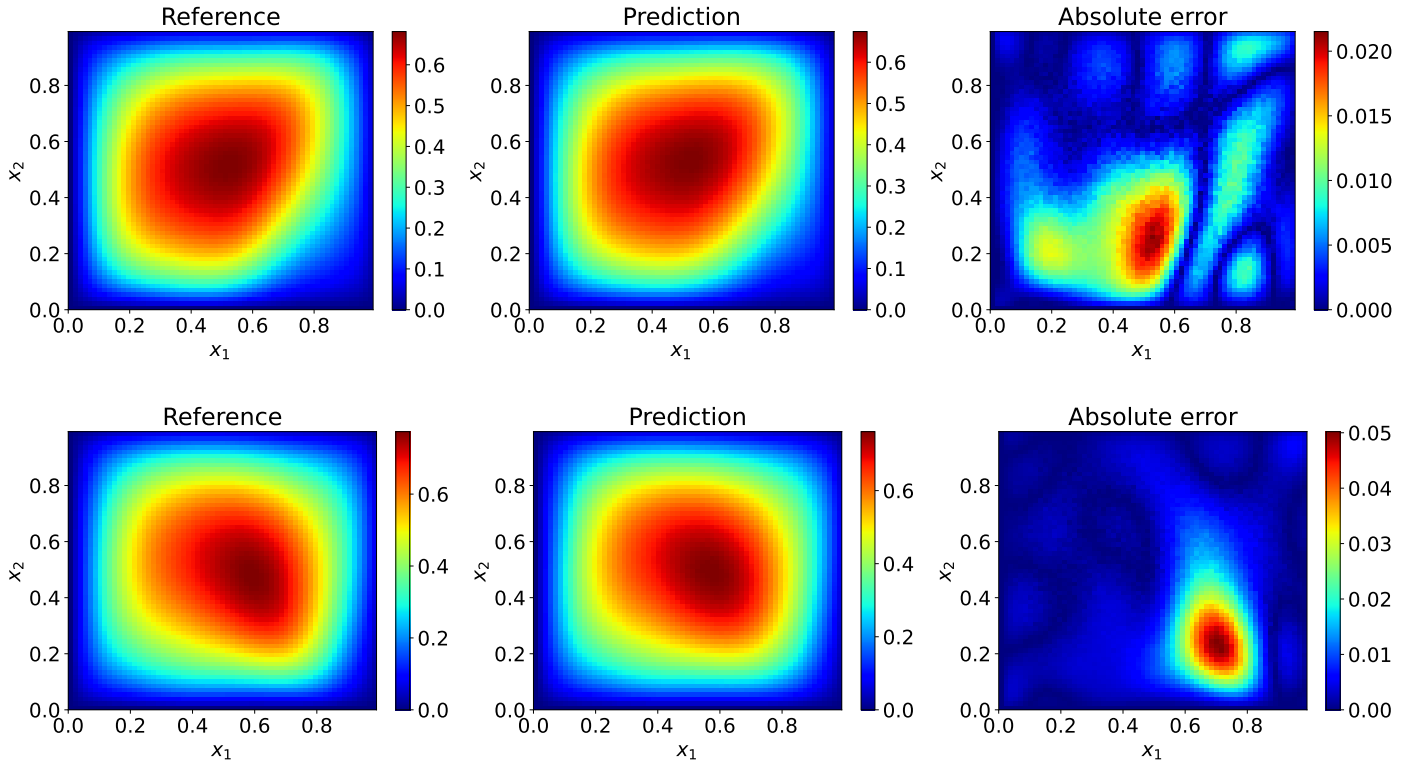


Fig. 6 Forward map - Comparison of the true PDE-output/solution s (given feature coefficients as the PDE-input u) with the one predicted by the proposed invertible DeepONet and for Darcy-type PDE.

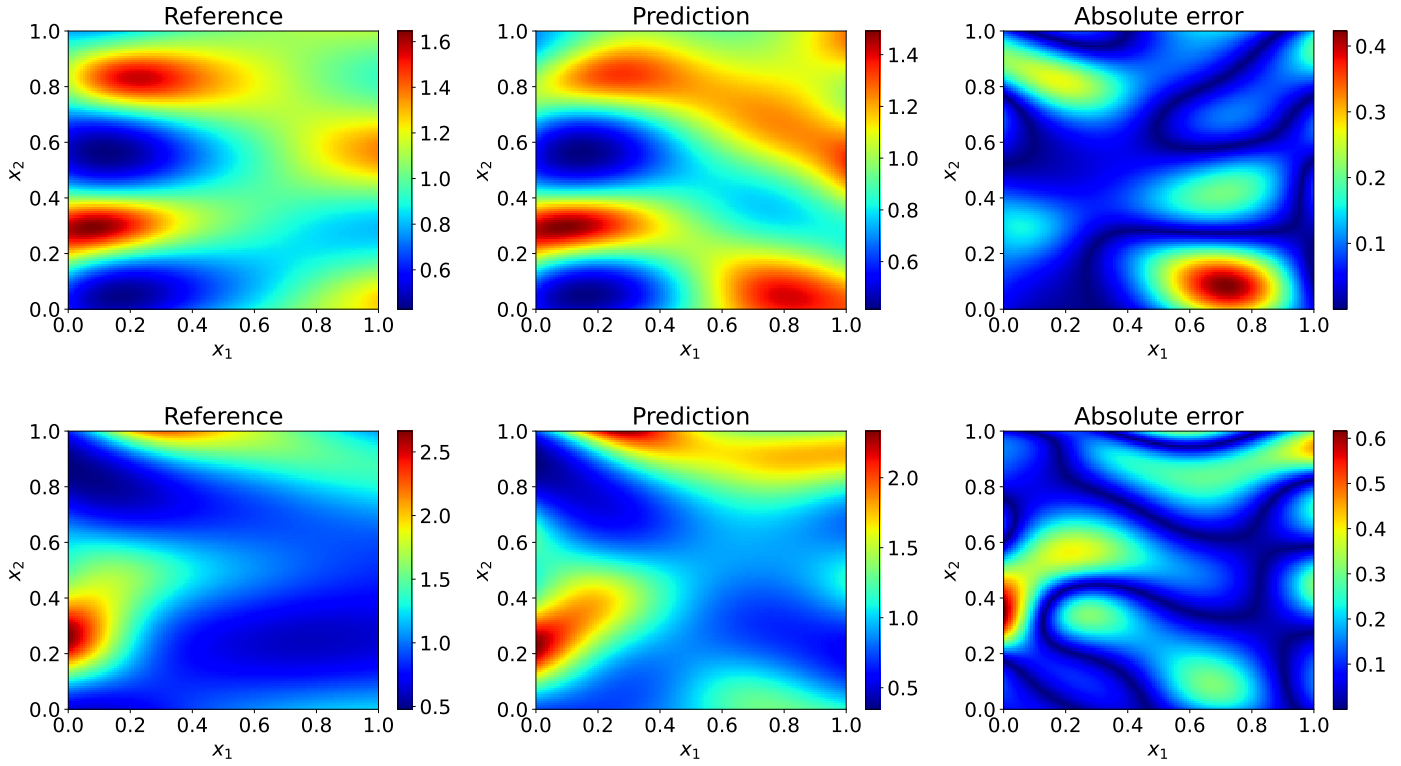


Fig. 7 Inverse map - Comparison of the reconstructed PDE-input (given the PDE-output/solution s) with the one predicted by the proposed invertible DeepONet and for Darcy-type PDE with zero labeled training data.

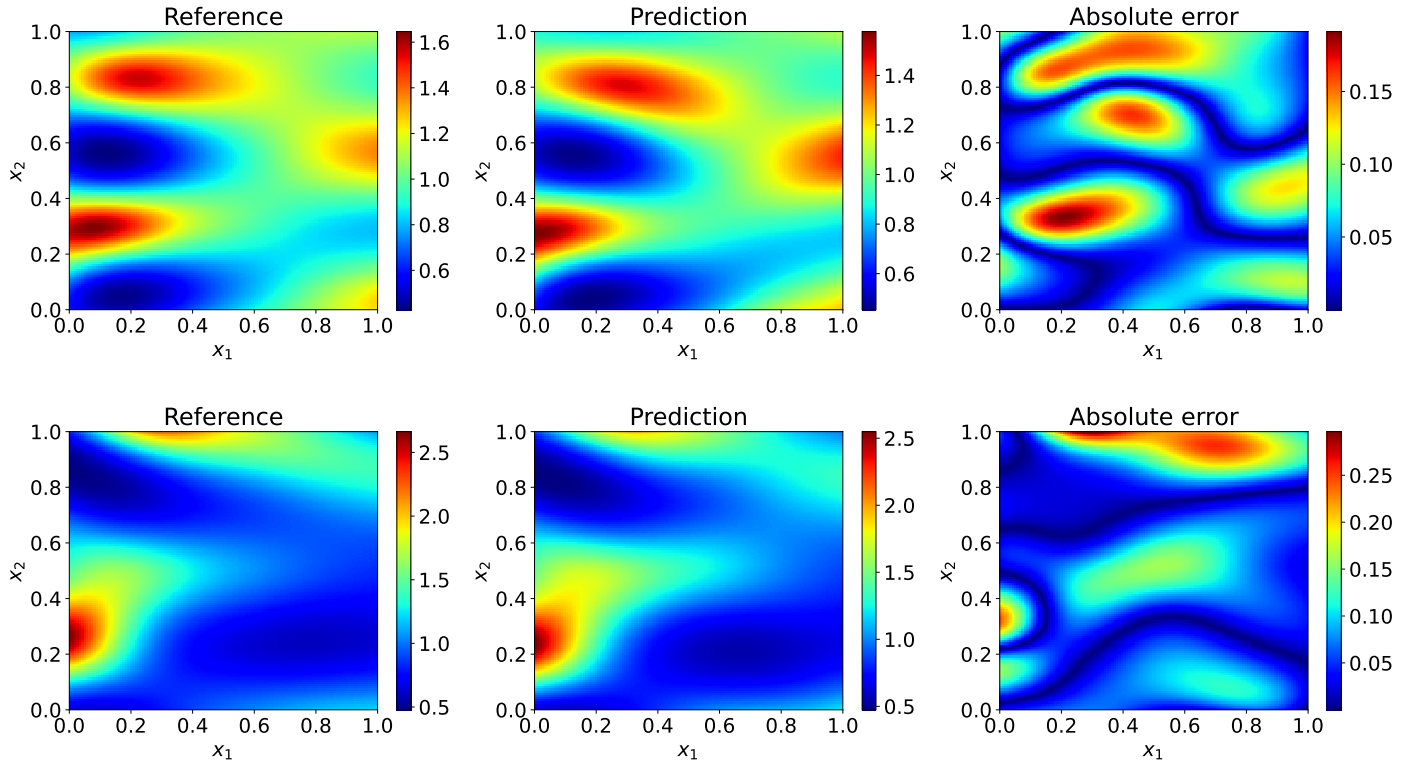


Fig. 8 Inverse map - Comparison of the reconstructed PDE-input (given the PDE-output/solution s) with the one predicted by the proposed invertible DeepONet and for Darcy-type PDE.

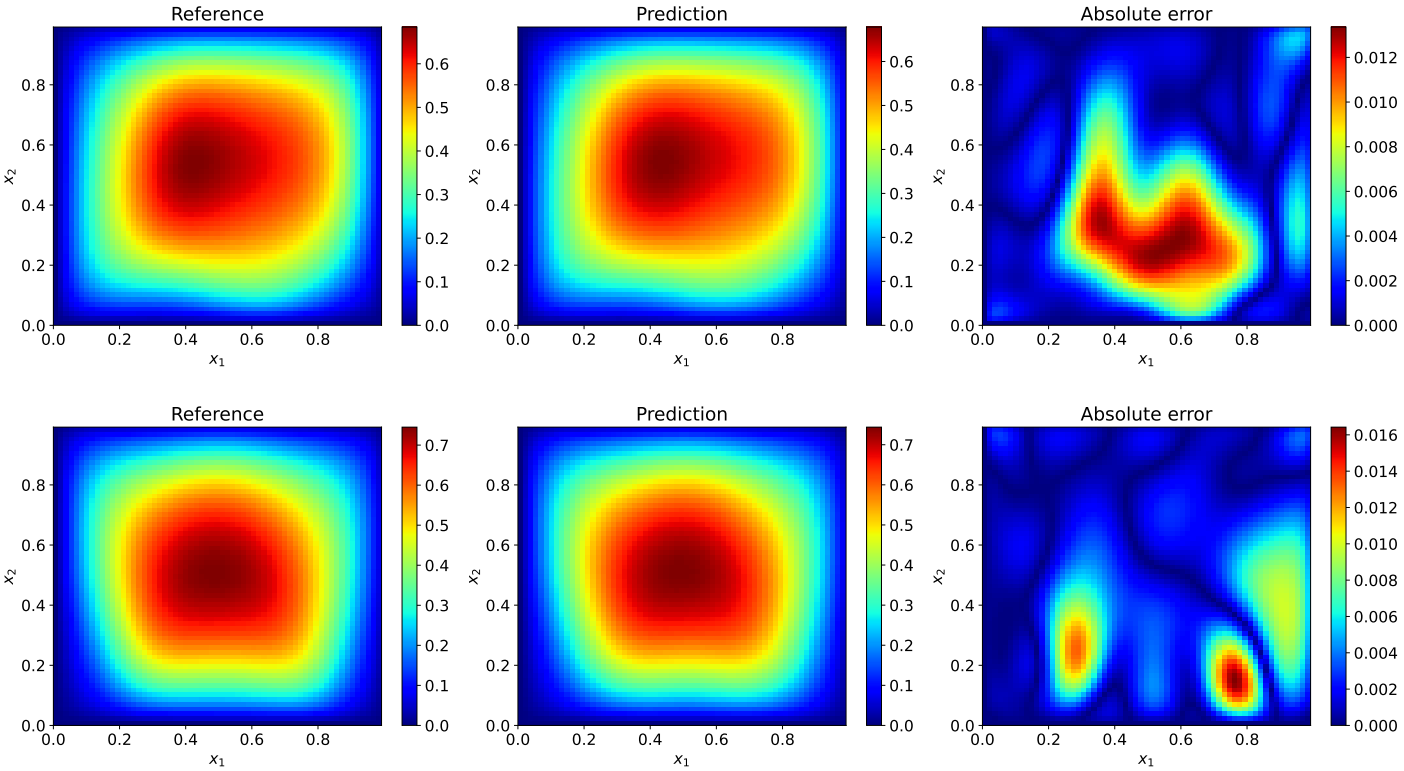


Fig. 9 Forward map - Comparison of the true PDE-output/solution s (given the coarse-grained PDE-input u) with the one predicted by the proposed invertible DeepONet and for Darcy-type PDE.

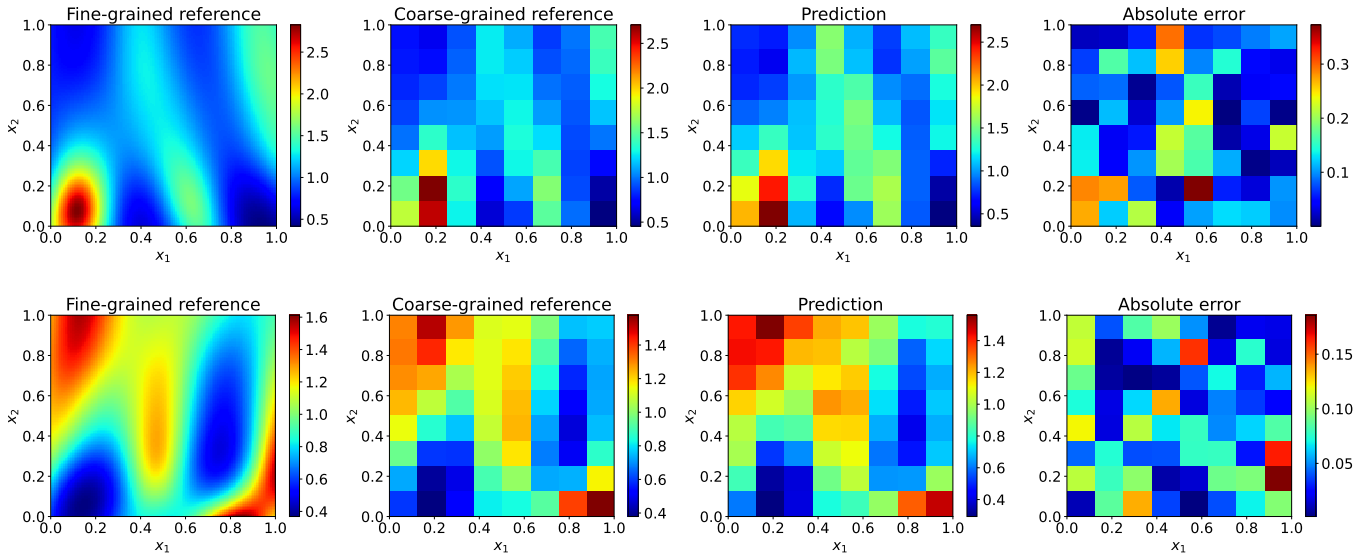


Fig. 10 Inverse map - Comparison of the coarse-grained PDE-input u (given the PDE-output/solution s) with the one predicted by the proposed invertible DeepONet and for Darcy-type PDE with zero labeled training data.

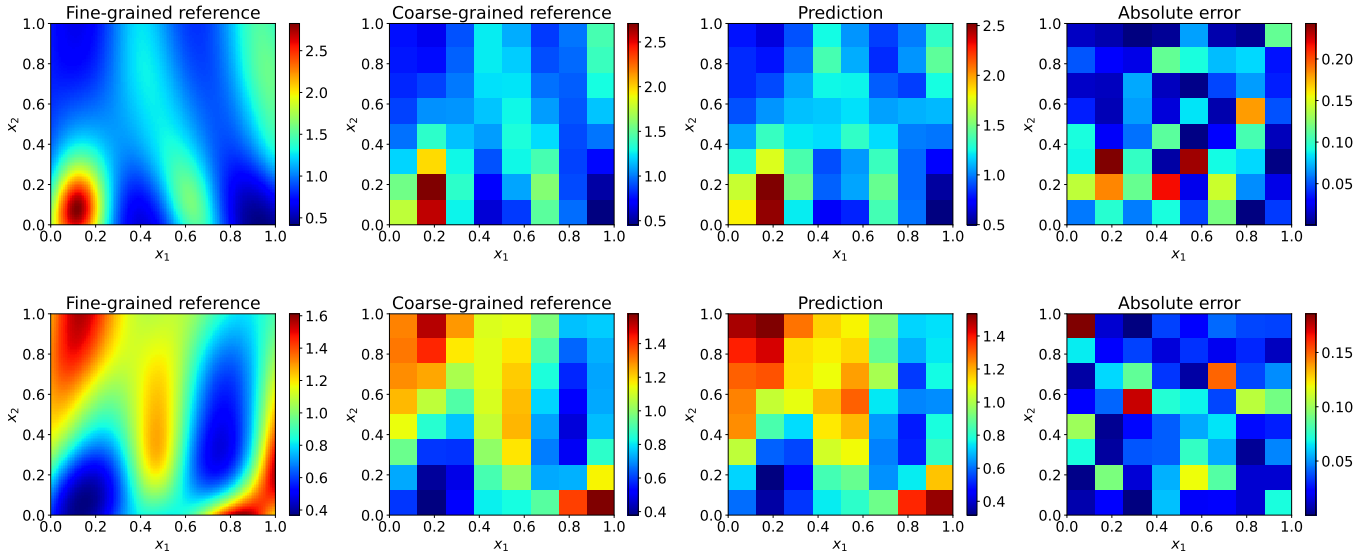


Fig. 11 Inverse map - Comparison of the coarse-grained PDE-input (given the PDE-output/solution s) with the one predicted by the proposed invertible DeepONet and for Darcy-type PDE.

Figure 13 we increased the noise level ten-fold, to $\sigma^2 = 0.01$ and, as expected, so did the posterior uncertainty. In Figure 14 we used $\sigma^2 = 0.001$ but decreased the number of observations of the PDE-solution to 25 points (instead of 100). As expected, this led to an increase in posterior uncertainty.

Our method can therefore be used as a fast approach without any need for optimization and MCMC sampling to generate an approximate posterior. We note that the posterior uncertainty increases if number of observations decreases or if the observation noise σ^2 increases. In Appendix B, we show the excellent agreement of the approximate posterior computed with the actual one as obtained by costly and time-consuming MCMC simulations.

3.4.2 Flow through porous media

We also solved a Bayesian inverse problem in the context of the Darcy-type PDE by using our trained model of section 3.3 with added labeled

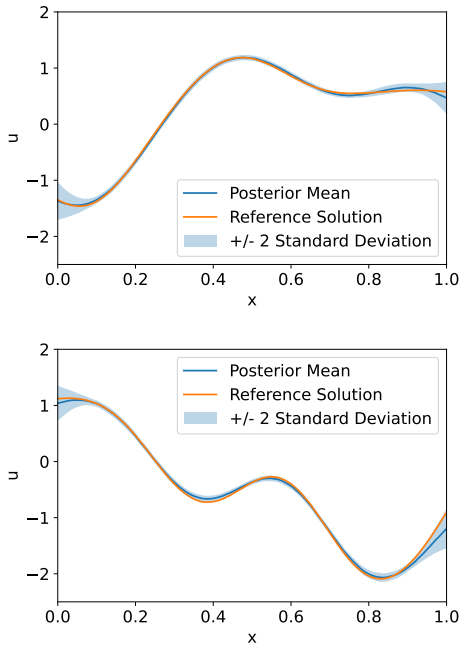


Fig. 12 Bayesian Inverse Problem for Reaction-Diffusion PDE: 100 observed data points with $\sigma^2 = 0.001$ and a 100-dimensional parametric input.

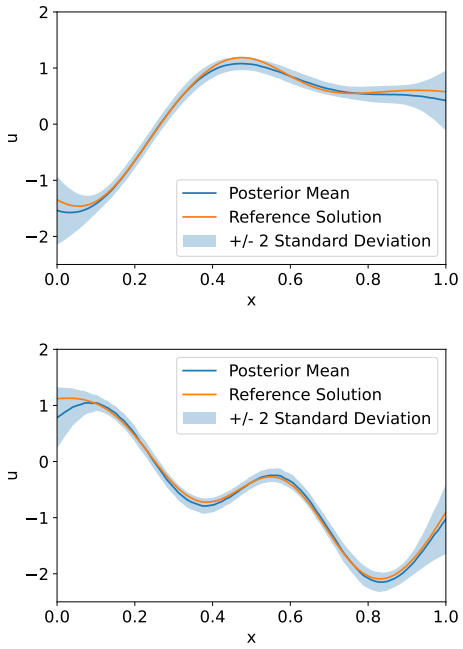


Fig. 13 Bayesian Inverse Problem for Reaction-Diffusion PDE: 100 observed data points with $\sigma^2 = 0.01$ and a 100-dimensional parametric input.

training data. We computed an approximate posterior based on the algorithm presented in section 2.5 and compared it with the true PDE-input. For the Gaussian mixture models involved for the prior and subsequently the posterior on $\boldsymbol{b}$ we used two mixture components i.e. $M = 2$ in Equations (27), (29).

Firstly, we considered permeability fields represented with respect to 64 known feature functions as described in section 3.3. The 64 coefficients c (Equation (36)) represented the sought PDE-inputs and a uniform prior in $[0, 1]^{64}$ was employed. The results in terms of the permeability field u can be seen in the following Figures. The obtained posterior is in good agreement with the ground truth, e.g. the PDE-input field used to generate the data with the PDE-solver.

In particular, in Figure 15 we assumed that 3844 observations of the PDE-output were available, on a 62×62 regular grid. The data that was synthetically generated was contaminated with Gaussian noise with $\sigma^2 = 0.001$ (see Equation (23)). In Figure 16 we increased the noise level and subsequently the posterior uncertainty was slightly higher but the posterior mean is still close to the ground truth. In Figure 17 we used $\sigma^2 = 0.01$ but decreased the number of observations by 50% to 1922. As expected, the posterior uncertainty increased again but still encapsulated the ground truth.

Finally, we considered the case where the PDE-input is represented on a regular 8×8 grid as in section 3.3.1. The discretized GP described therein was used as the prior. In Figure 18 we compare the ground truth with the posterior mean and standard deviation as obtained from 3844 observations on a 62×62 regular grid and for a noise level of $\sigma^2 = 0.01$ (see Equation (23)). In Figure 19 we used lower noise with $\sigma^2 = 0.001$ level and, as expected, the posterior uncertainty was lower and the posterior mean was closer to the ground truth. In Figure 20 we again choose the previous noise level but decreased the number of observations by half, to 1922. As expected, the posterior uncertainty increased but still encapsulated the ground truth.

4 Conclusions

We introduced an invertible DeepONet architecture for constructing data-driven surrogates of PDEs with parametric inputs. The use of the RealNVP architecture in the branch-network enables

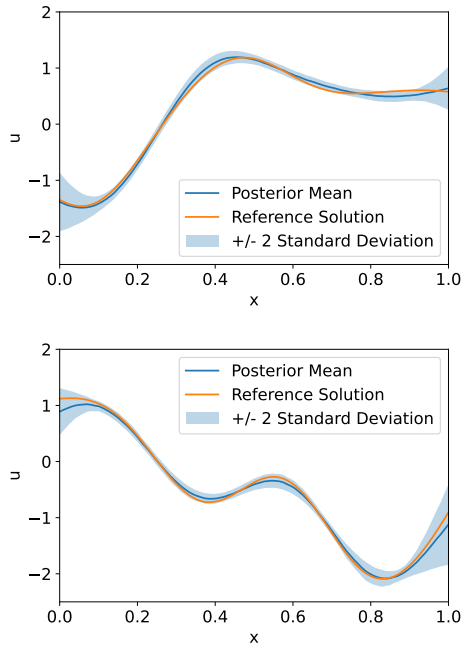


Fig. 14 Bayesian Inverse Problem for Reaction-Diffusion PDE: 25 observed data points with $\sigma^2 = 0.001$ and a 100-dimensional parametric input.

one to obtain simultaneously accurate approximations of both the forward and the inverse map (i.e. from PDE-solution to PDE-input). The latter is particularly useful for deterministic and stochastic (Bayesian), PDE-based, inverse problems for which accurate solutions can be readily obtained once the proposed DeepONet has been trained offline. The training framework can make use of expensive, labeled data (i.e. PDE input-output pairs) as well as inexpensive, unlabeled data (i.e. only PDE-inputs) by incorporating residuals of the governing PDE and its boundary/initial conditions into the loss function. The use of labeled data was generally shown to improve predictive accuracy and especially in terms of the inverse map which is something that warrants further investigation.

In the case of Bayesian formulations in particular, we showed that the availability of the inverse map can lead to highly-efficient approximations of the sought posterior without the need of additional PDE solves and without any cumbersome sampling (e.g. due to MCMC, SMC) or iterations (e.g. due to SVI).

The performance of the proposed strategy was demonstrated on several PDEs with modest- to high-dimensional parametric inputs and its efficiency was assessed in terms of the amounts of labeled vs unlabeled data. Furthermore, the approximate posterior obtained was in very good agreement with the exact posterior obtained with the reference solver and MCMC. The accuracy persisted for various levels of noise in the data as well as when changing the number of available observations. We note finally that unbiased estimates with respect to the exact posterior could be readily obtained with Importance Sampling and by using the approximate posterior as the importance sampling density. This would nevertheless imply additional PDE solves which we would expect to be modest in number given the accuracy of the approximation i.e. the proximity of the Importance Sampling density with the actual posterior.

A Influence of the amount of data

This section contains additional results as obtained for the antiderivative example and for different amounts of training data. We chose exactly the same settings as described in Section 3.1 and varied only the amount of labeled and unlabeled training data. In Figure 21 we plot the relative error in the foward and inverse map with regards to the amount of unlabeled training data. The color indicates the amount of labeled training data used, i.e. blue curves correspond to 1% labeled training data, whereas red curves correspond to 100% labeled training data.

We observe that although the relative errors decrease with the addition of more data, the benefit is more pronounced with the addition of labeled data.

B Comparison with MCMC

In the main part of this article we already showed that the true parameter input is encapsulated by the posterior. In this section we compare the approximate posterior computed with the reference posterior obtained by MCMC.
In particular, for two, randomly-chosen cases in the reaction-diffusion example, the true posterior was computed using the NUTS sampler from the Blackjax library (Lao and Louf, 2020). As is

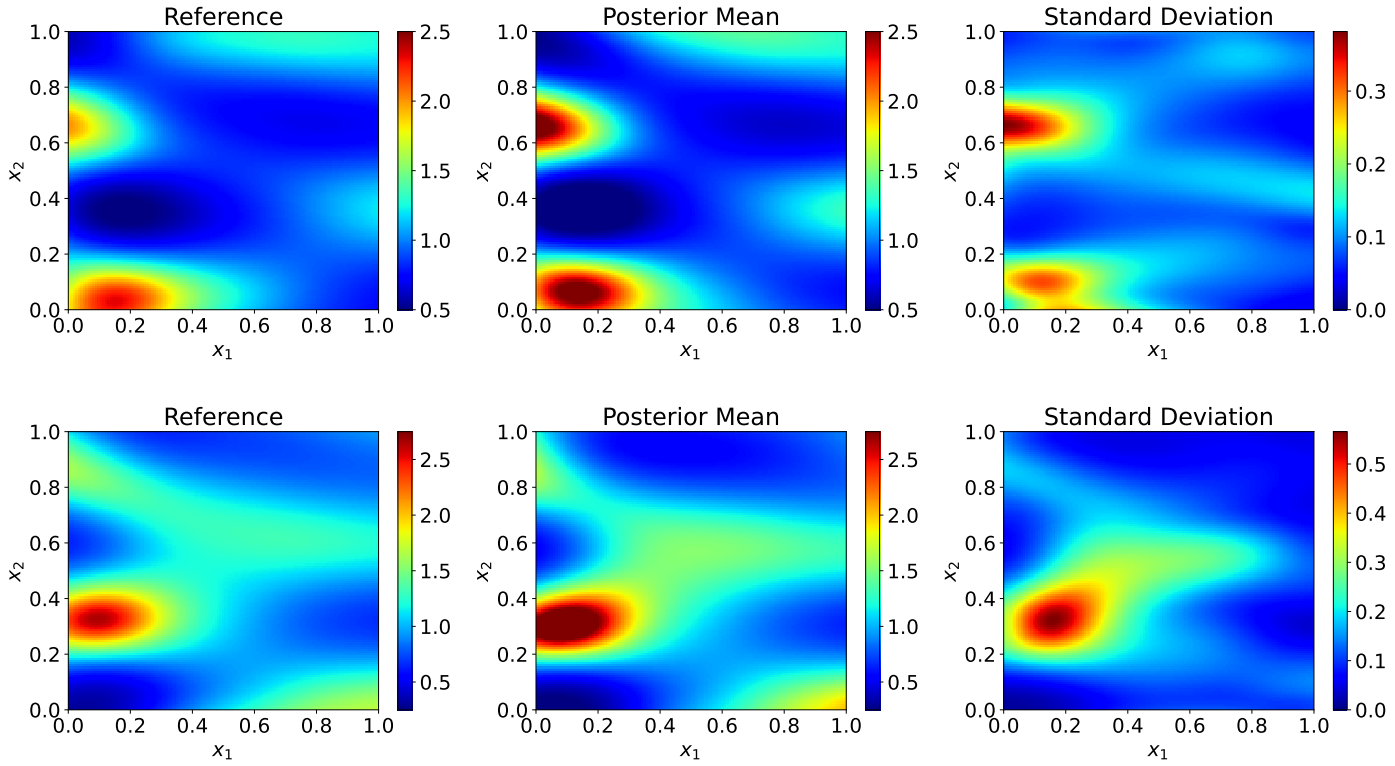


Fig. 15 Bayesian Inverse Problem for Darcy-type PDE: 3844 observed data points with $\sigma^2 = 0.001$ and a 64-dimensional parametric input representing feature coefficients.

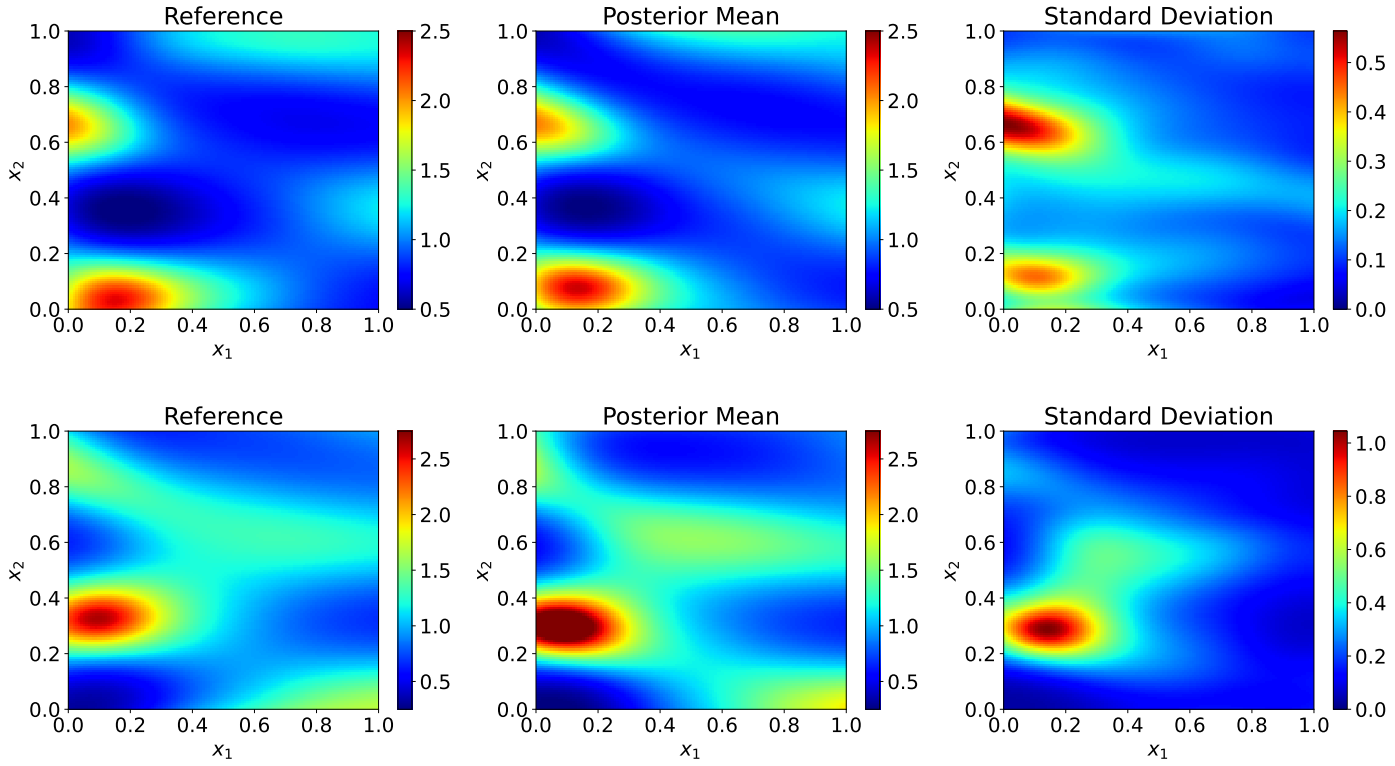


Fig. 16 Bayesian Inverse Problem for Darcy-type PDE: 3844 observed data points with $\sigma^2 = 0.01$ and a 64-dimensional parametric input representing feature coefficients.

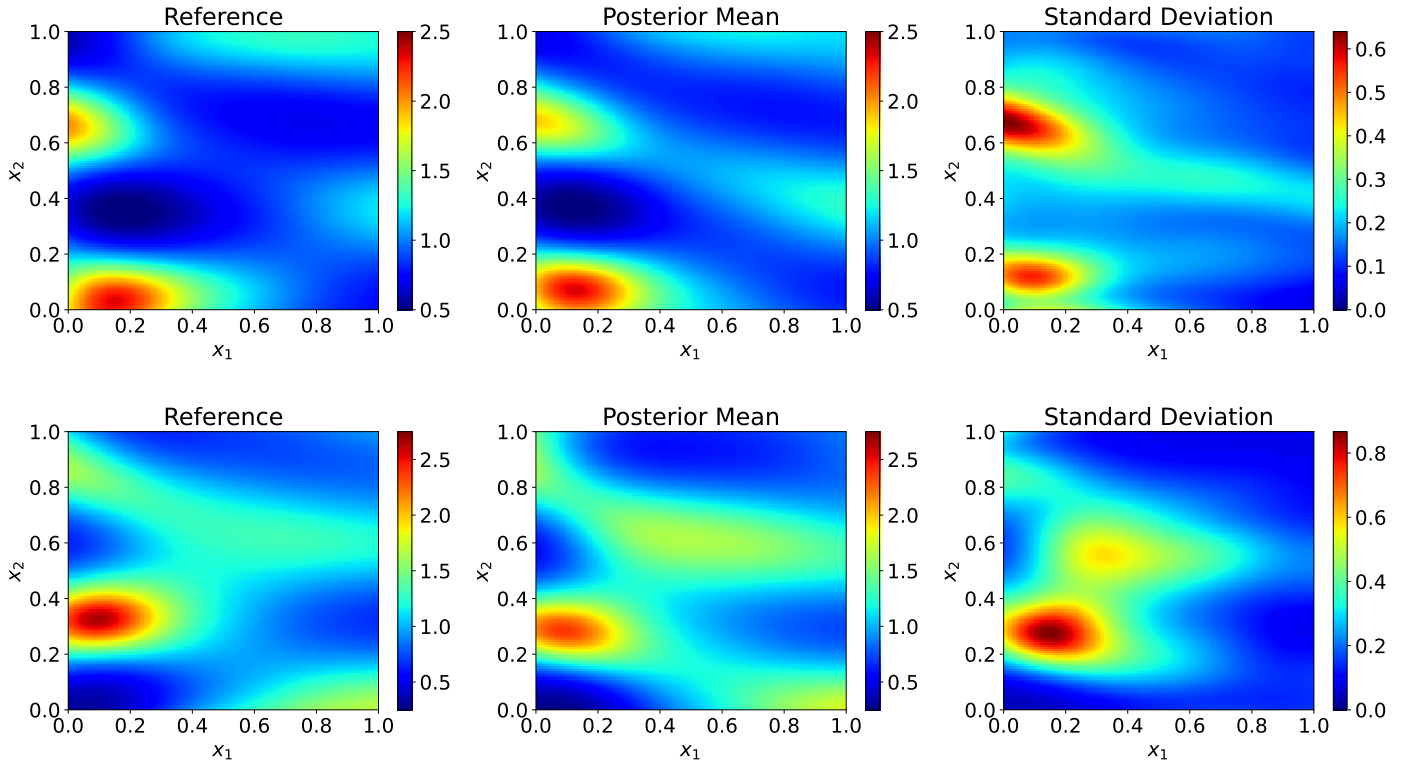


Fig. 17 Bayesian Inverse Problem for Darcy-type PDE: 1922 observed data points with $\sigma^2 = 0.01$ and a 64-dimensional parameter input representing feature coefficients.

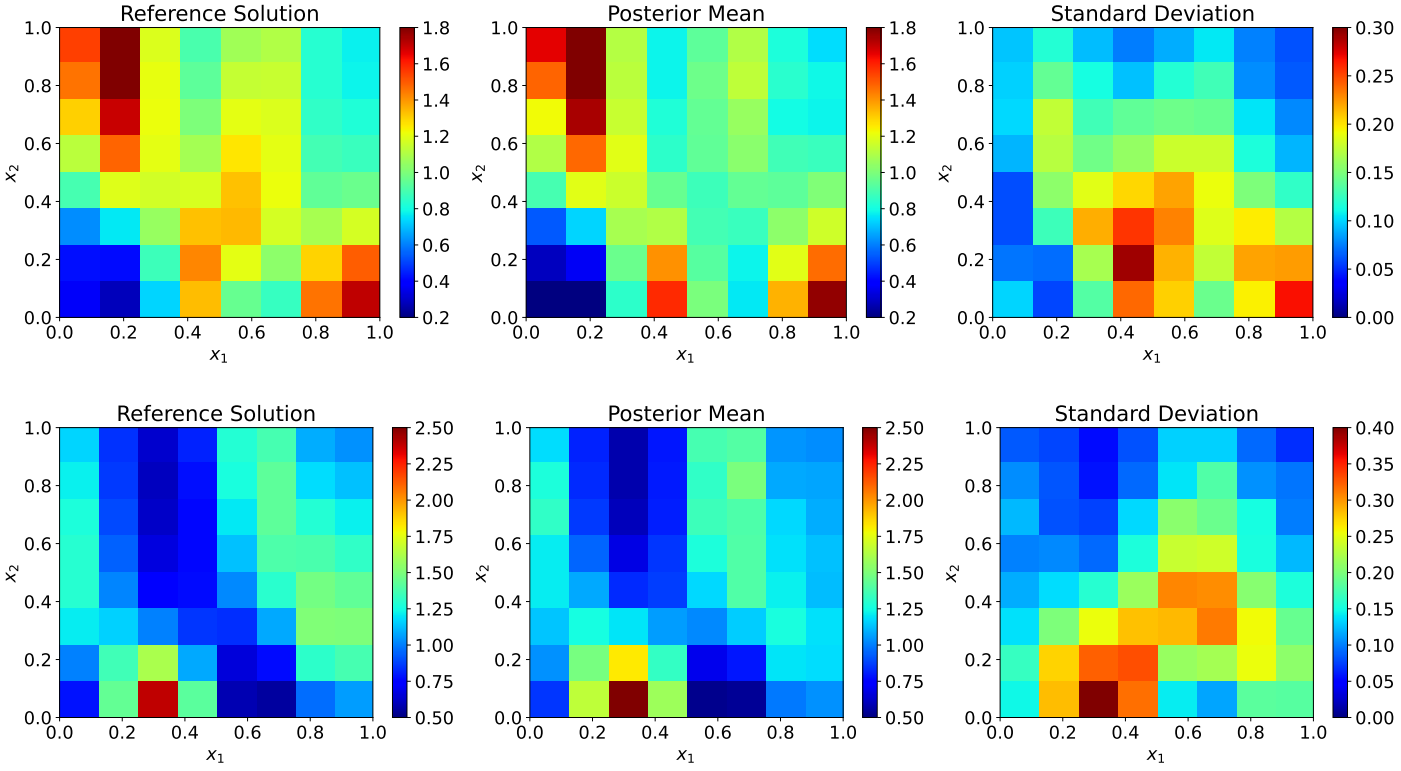


Fig. 18 Bayesian Inverse Problem for Darcy-type PDE: 3844 observations with $\sigma^2 = 0.01$ and a 64-dimensional, discretized permeability field.

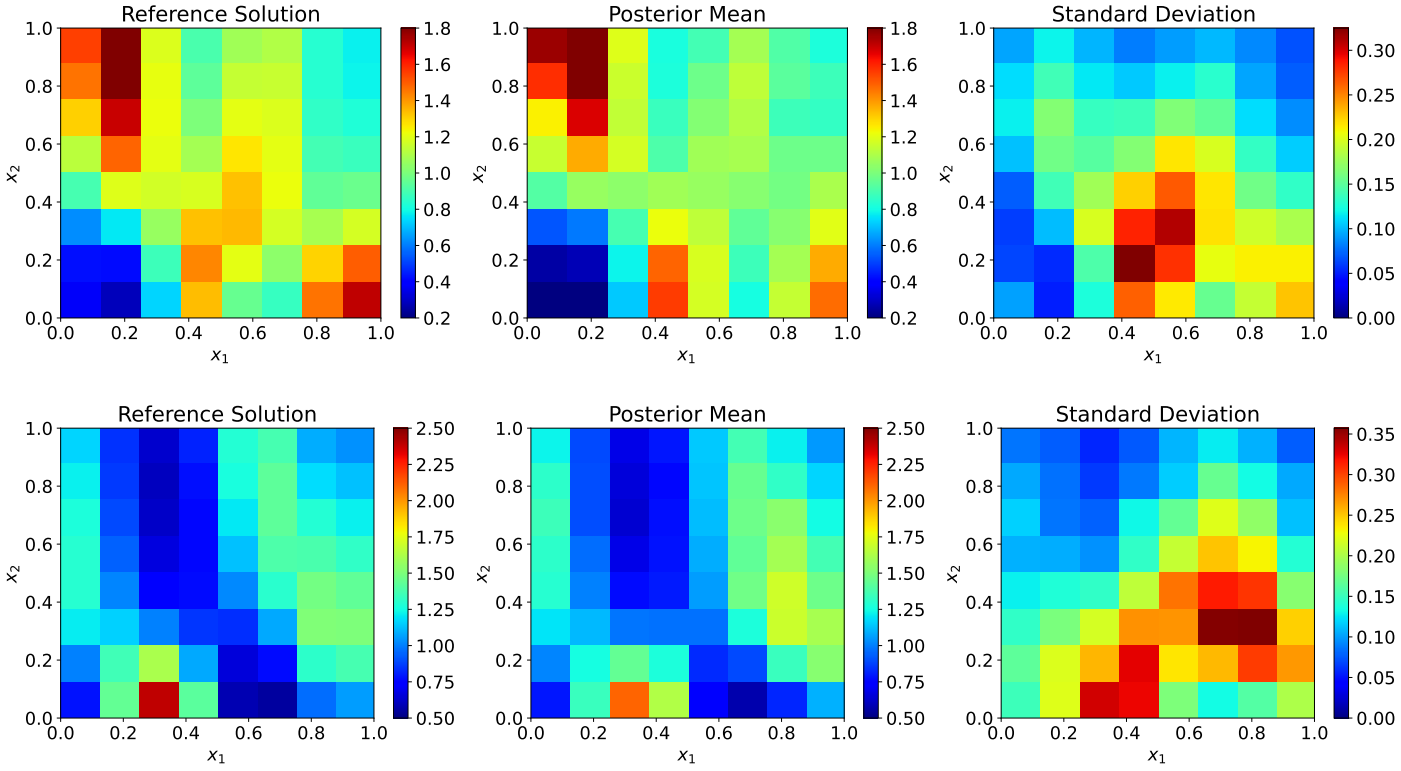


Fig. 19 Bayesian Inverse Problem for Darcy-type PDE: 3844 observations with $\sigma^2 = 0.001$ and a 64-dimensional discretized permeability field.

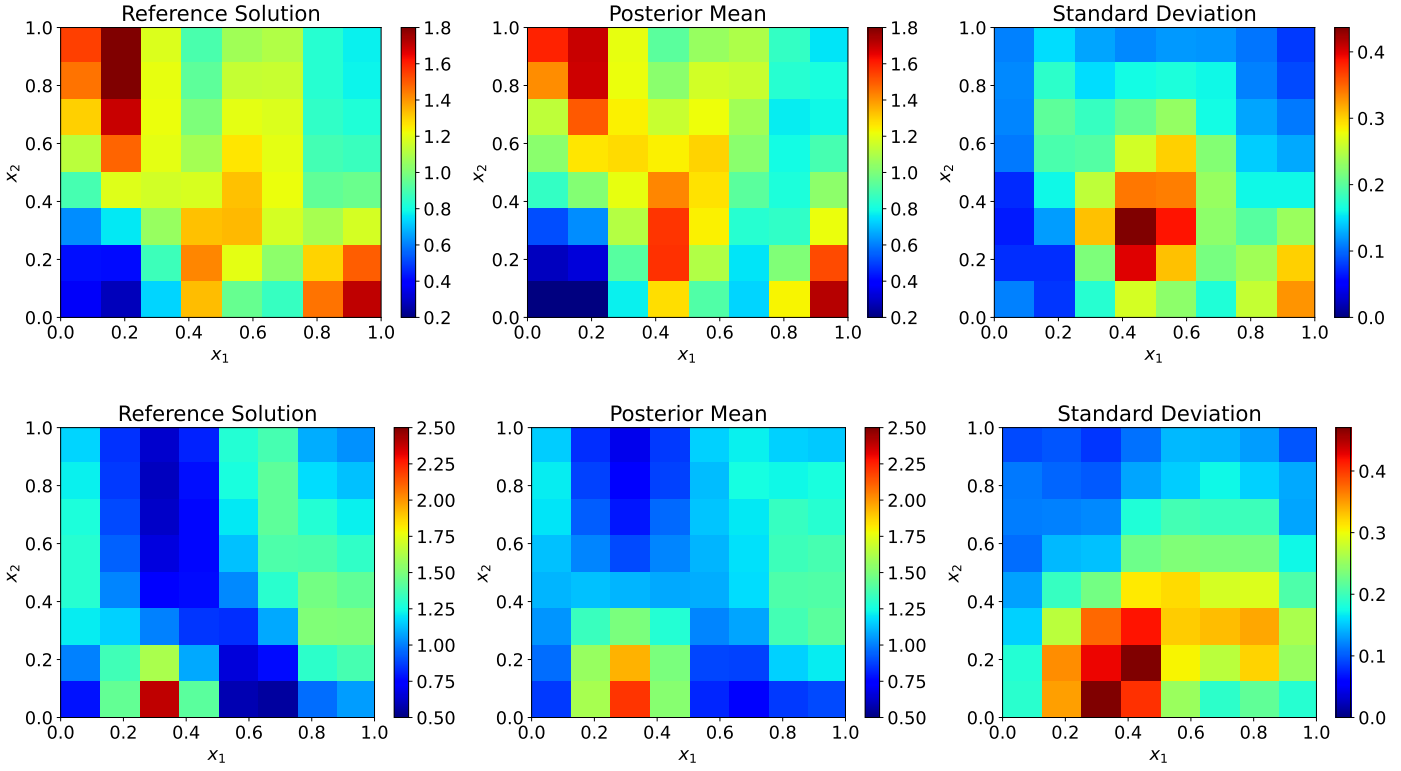


Fig. 20 Bayesian Inverse Problem for Darcy-type PDE: 1922 observations with $\sigma^2 = 0.01$ and a 64-dimensional discretized permeability field.

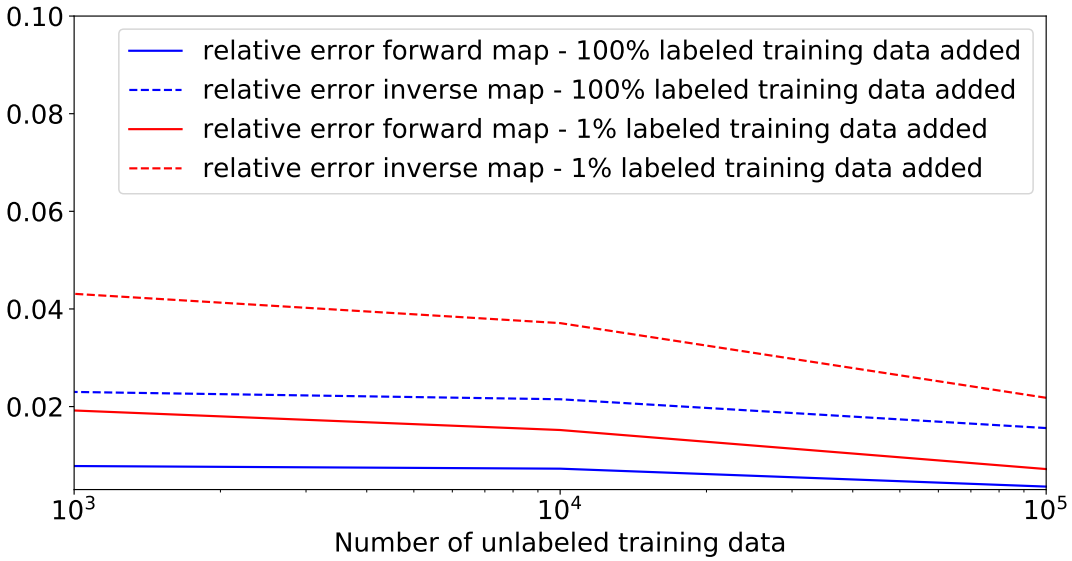


Fig. 21 Relative errors on test data for the forward and inverse map depending on the amount of labeled and unlabeled training data

the case with all MCMC-based inference schemes, these provide the reference posterior (asymptotically). The results shown in Figure 22 in terms of the posterior mean $\pm$ 2 posterior standard deviations indicate excellent accuracy of the posterior approximation proposed. While our method does not require any new forward model evaluation or model gradients, the MCMC algorithms require a forward model solve and its gradients for each sample. For the MCMC-based results displayed in total 40000 samples were generated.

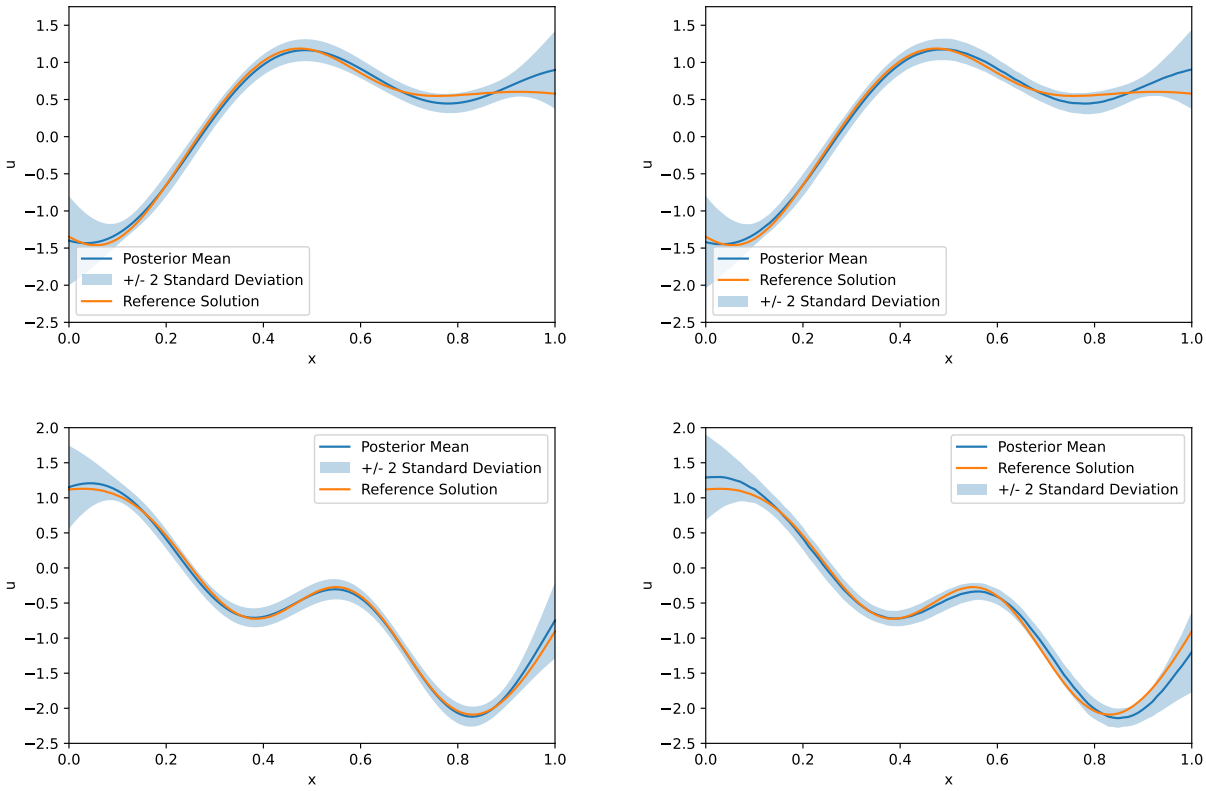


Fig. 22 Bayesian Inverse Problem for Reaction-Diffusion PDE: 100 observed data points with $\sigma^2 = 0.01$ and a 100-dimensional parameter input. Left: Posterior based on MCMC (NUTS), Right: Posterior obtained by our algorithm

References

Adler J, Öktem O (2017) Solving ill-posed inverse problems using iterative deep neural networks. Inverse Problems 33(12):124,007

Ardizzone L, Kruse J, Wirkert S, et al (2018) Analyzing inverse problems with invertible neural networks. arXiv preprint arXiv:180804730

Behrmann J, Grathwohl W, Chen RTQ, et al (2019) Invertible residual networks. ICML

Beskos A, Girolami M, Lan S, et al (2017) Geometric MCMC for infinite-dimensional inverse problems. Journal of Computational Physics 335:327–351. https://doi.org/10.1016/j.jcp.2016.12.041, URL https://www.sciencedirect.com/science/article/pii/S0021999116307033

Bhattacharya K, Hosseini B, Kovachki NB, et al (2020) Model reduction and neural networks for parametric pdes. arXiv preprint arXiv:200503180

Bishop CM, Nasrabadi NM (2006) Pattern recognition and machine learning. Springer

Bradbury J, Frostig R, Hawkins P, et al (2018) JAX: composable transformations of Python+NumPy programs. URL http://github.com/google/jax

Champion KP, Brunton SL, Kutz JN (2019) Discovery of nonlinear multiscale systems: Sampling strategies and embeddings. SIAM Journal on Applied Dynamical Systems 18(1):312–333

Detommaso G, Cui T, Marzouk Y, et al (2018) A Stein variational Newton method. In: Advances in Neural Information Processing Systems, pp 9169–9179

Dinh L, Sohl-Dickstein J, Bengio S (2016) Density estimation using real nvp. arXiv preprint arXiv:160508803

Franck IM, Koutsourelakis PS (2017) Multimodal, high-dimensional, model-based, Bayesian inverse problems with applications in biomechanics. Journal of Computational Physics 329:91–125. https://doi.org/10.1016/j.jcp.2016.10.039, URL http://www.sciencedirect.com/science/article/pii/S002199911630537X

Gin C, Lusch B, Brunton SL, et al (2019) Deep learning models for global coordinate transformations that linearize pdes. arXiv preprint arXiv:191102710

Kalia M, Brunton SL, Meijer HG, et al (2021) Learning normal form autoencoders for data-driven discovery of universal, parameter-dependent governing equations. arXiv preprint arXiv:210605102

Kaltenbach S, Koutsourelakis PS (2020) Incorporating physical constraints in a deep probabilistic machine learning framework for coarse-graining dynamical systems. Journal of Computational Physics 419:109,673

Kaltenbach S, Koutsourelakis PS (2021) Physics-aware, probabilistic model order reduction with guaranteed stability. ICLR

Karnakov P, Litvinov S, Koumoutsakos P (2022) Optimizing a discrete loss (odil) to solve forward and inverse problems for partial differential equations using machine learning tools. arXiv preprint arXiv:220504611

Karniadakis GE, Kevrekidis IG, Lu L, et al (2021) Physics-informed machine learning. Nature Reviews Physics 3(6):422–440. https://doi.org/10.1038/s42254-021-00314-5, URL https://www.nature.com/articles/s42254-021-00314-5, number: 6 Publisher: Nature Publishing Group

Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980

Kissas G, Seidman J, Guilhoto LF, et al (2022) Learning operators with coupled attention. arXiv preprint arXiv:220101032

Klus S, Nüske F, Koltai P, et al (2018) Data-Driven Model Reduction and Transfer Operator Approximation. Journal of Nonlinear Science 28(3):985–1010. https://doi.org/10.

1007/s00332-017-9437-7, URL https://doi.org/10.1007/s00332-017-9437-7

Koopman BO (1931) Hamiltonian Systems and Transformations in Hilbert Space. Proceedings of the National Academy of Sciences of the United States of America 17(5):315–318. URL https://www.jstor.org/stable/86114

Koutsourelakis P (2009) A multi-resolution, non-parametric, Bayesian framework for identification of spatially-varying model parameters. Journal of Computational Physics 228(17):6184–6211

Koutsourelakis P, Zabaras N, Girolami M (2016) Big data and predictive computational modeling. JCoPh 321:1252–1254

Lagaris IE, Likas A, Fotiadis DI (1998) Artificial neural networks for solving ordinary and partial differential equations. IEEE transactions on neural networks 9(5):987–1000

Lao J, Louf R (2020) Blackjax: A sampling library for JAX. URL http://github.com/blackjax-devs/blackjax

Lee K, Carlberg KT (2020) Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. Journal of Computational Physics 404:108,973

Li Z, Kovachki N, Azizzadenesheli K, et al (2020) Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:201008895

Logg A, Mardal KA, Wells G (2012) Automated solution of differential equations by the finite element method: The FEniCS book, vol 84. Springer Science & Business Media

Lu L, Jin P, Karniadakis GE (2019) Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. arXiv preprint arXiv:191003193

Lu L, Jin P, Pang G, et al (2021) Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. Nature Machine Intelligence 3(3):218–229

Mo S, Zabaras N, Shi X, et al (2019) Deep Autoregressive Neural Networks for High-Dimensional Inverse Problems in Groundwater Contaminant Source Identification. Water Resources Research 55(5):3856–3881. https://doi.org/10.1029/2018WR024638, URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2018WR024638

Raissi M, Perdikaris P, Karniadakis GE (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

Sander ME, Ablin P, Blondel M, et al (2021) Momentum residual neural networks. arXiv 210207870

Wainwright M, Jordan M (2008) Graphical models, exponential families, and variational inference. In: Foundations and Trends in Machine Learning, vol 1. p 1–305

Wang S, Wang H, Perdikaris P (2021) Learning the solution operator of parametric partial differential equations with physics-informed deeponets. arXiv preprint arXiv:210310974