

Uncertainty Quantification in Scientific Machine Learning: Theory, Methods, and Software

by

Zongren Zou

B.Sc., Peking University, Beijing, China, 2017

M.Sc., Harvard University, Cambridge, MA, USA, 2019

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Division of Applied Mathematics at Brown University

PROVIDENCE, RHODE ISLAND

February 2025

© Copyright 2025 by Zongren Zou

This dissertation by Zongren Zou is accepted in its present form
by the Division of Applied Mathematics as satisfying the
dissertation requirement for the degree of Doctor of Philosophy.

Date_____

George Em Karniadakis, Ph.D., Advisor

Recommended to the Graduate Council

Date_____

Jérôme Darbon, Ph.D., Reader

Date_____

Houman Owhadi, Ph.D., Reader

Date_____

Daniel M. Tartakovsky, Ph.D., Reader

Approved by the Graduate Council

Date_____

Thomas A. Lewis, Dean of the Graduate School

Vita

Zongren Zou began his undergraduate studies at Peking University in September 2013, majoring in Theoretical and Applied Mechanics. He graduated with a Bachelor of Science degree in July 2017. Subsequently, he pursued graduate studies at Harvard University, earning a Master of Science degree in Computational Science and Engineering in May 2019. In September 2019, Zongren joined the Division of Applied Mathematics at Brown University to pursue a Doctor of Philosophy degree.

Zongren's research focuses on scientific machine learning and artificial intelligence for science, with a particular emphasis on uncertainty quantification. Under the mentorship of Professor George Em Karniadakis, Zongren has authored 17 publications, including one in *SIAM Review*. He has also been invited to present his research at eight conferences and research institutes.

Zongren has substantial teaching experience. At Brown University, he served as a teaching assistant for Introduction to Computational Linear Algebra (Fall 2020) and Monte Carlo Simulation with Applications to Finance (Spring 2021). At Harvard University, he worked as a teaching fellow for Advanced Scientific Computing (Fall 2018), and as an undergraduate, he was a teaching assistant for Theoretical Mechanics at Peking University (Fall 2015). Additionally, Zongren was invited to deliver guest lectures on Uncertainty Quantification in Scientific Machine Learning at Brown University (Spring 2023) and Yale University (Fall 2023).

Beyond academics, Zongren has actively contributed to professional service. He has reviewed 20 manuscripts as a peer reviewer for 10 scientific journals, including *SIAM Journal on Scientific Computing*, *SIAM/ASA Journal on Uncertainty Quantification*, *Journal of Computational Physics*, *Computer Methods in Applied Mechanics and Engineering*, and *Neurocomputing*. Moreover, Zongren organized and hosted the weekly CRUNCH seminar, a global online forum on machine learning, for over two years.

Publications

* indicates equal contributions or alphabetical order by last names.

Journal papers

1. Kevin Linka, Amelie Schäfer, Xuhui Meng, **Zongren Zou**, George Em Karniadakis and Ellen Kuhl. Bayesian Physics Informed Neural Networks for real-world nonlinear dynamical systems. *Computer Methods in Applied Mechanics and Engineering* 402 (2022) 115346.
2. Apostolos F. Psaros, Xuhui Meng, **Zongren Zou**, Ling Guo and George Em Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics* 477 (2023) 111902.
3. Minglang Yin*, **Zongren Zou***, Enrui Zhang, Cristina Cavinato, Jay D. Humphrey and George Em Karniadakis. A generative modeling framework for inferring families of biomechanical constitutive laws in data-sparse regimes. *Journal of the Mechanics and Physics of Solids* 181 (2023) 105424.
4. Zhen Zhang, **Zongren Zou**, Ellen Kuhl and George Em Karniadakis. Discovering a reaction-diffusion model for Alzheimer’s disease by combining PINNs

- with symbolic regression. *Computer Methods in Applied Mechanics and Engineering* 419 (2024) 116647.
5. **Zongren Zou**, Xuhui Meng, Apostolos F. Psaros and George Em Karniadakis. NeuralUQ: A comprehensive library for uncertainty quantification in neural differential equations and operators. *SIAM Review* 66 (1) (2024) 161-190.
 6. **Zongren Zou**, Xuhui Meng and George Em Karniadakis. Correcting model misspecification in physics-informed neural networks (PINNs). *Journal of Computational Physics* 505 (2024) 112918.
 7. Paula Chen*, Tingwei Meng*, **Zongren Zou***, Jérôme Darbon and George Em Karniadakis. Leveraging multi-time Hamilton-Jacobi PDEs for certain scientific machine learning problems. *SIAM Journal on Scientific Computing* 46 (2) (2024) C216–C248.
 8. Khemraj Shukla*, Juan Diego Toscano*, Zhicheng Wang*, **Zongren Zou*** and George Em Karniadakis. A comprehensive and FAIR comparison between MLP and KAN representations for differential equations and operator networks. *Computer Methods in Applied Mechanics and Engineering* 431 (2024) 117290.
 9. **Zongren Zou***, Tingwei Meng*, Paula Chen*, Jérôme Darbon and George Em Karniadakis. Leveraging viscous Hamilton-Jacobi PDEs for uncertainty quantification in scientific machine learning. *SIAM/ASA Journal on Uncertainty Quantification* 12 (4) (2024) 1165-1191.
 10. **Zongren Zou**, Xuhui Meng and George Em Karniadakis. Uncertainty quantification for noisy inputs-outputs in physics-informed neural networks and neural operators. *Computer Methods in Applied Mechanics and Engineering* 433 (2025) 117479.

11. Khemraj Shukla*, **Zongren Zou***, Chi Hin Chan, Additi Pandey, Zhicheng Wang and George Em Karniadakis. NeuroSEM: A hybrid framework for simulating multiphysics problems by coupling PINNs and spectral elements. *Computer Methods in Applied Mechanics and Engineering* 433 (2025) 117498.

Conference paper

1. Paula Chen*, Tingwei Meng*, **Zongren Zou***, Jérôme Darbon and George Em Karniadakis. Leveraging Hamilton-Jacobi PDEs with time-dependent Hamiltonians for continual scientific machine learning, in: *6th Annual Learning for Dynamics & Control Conference, PMLR*, 2024, pp. 1-12.

Preprints

1. **Zongren Zou** and George Em Karniadakis. L-HYDRA: Multi-head physics-informed neural networks. *arXiv preprint arXiv:2301.02152* (2023).
2. **Zongren Zou***, Adar Kahana*, Enrui Zhang, Eli Turkel, Rishikesh Ranade, Jay Pathak and George Em Karniadakis. Large scale scattering using fast solvers based on neural operators. *arXiv preprint arXiv:2405.12380* (2024).
3. Mario De Florio, **Zongren Zou**, Daniele E. Schiavazzi and George Em Karniadakis. Quantification of total uncertainty in the physics-informed reconstruction of CVSim-6 physiology. *arXiv preprint arXiv:2408.07201* (2024).
4. Tingwei Meng*, **Zongren Zou***, Jérôme Darbon and George Em Karniadakis. HJ-sampler: A Bayesian sampler for inverse problems of a stochastic process by

leveraging Hamilton-Jacobi PDEs and score-based generative models. *arXiv preprint arXiv:2409.09614* (2024).

5. Juan Diego Toscano, Vivek Oommen, Alan John Varghese, **Zongren Zou**, Nazanin Ahmadi Daryakenari, Chenxi Wu and George Em Karniadakis. From PINNs to PIKANs: Recent advances in physics-informed machine learning. *arXiv preprint arXiv:2410.13228* (2024).

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Professor George Em Karniadakis, for his invaluable guidance, unwavering support, and exceptional mentorship throughout my doctoral journey at Brown University over the past five and a half years. His profound insights, steady encouragement, and passion for research have been a constant source of inspiration. I feel truly fortunate to have had the opportunity to learn and grow under his guidance and to be part of the CRUNCH group.

I would also like to extend my heartfelt thanks to the members of my dissertation committee for their time, thoughtful feedback, and constructive suggestions, which have significantly enhanced the quality of this work.

I am particularly grateful to Professor Jérôme Darbon from Brown University and Professor Xuhui Meng from Huazhong University of Science and Technology for their insightful guidance and generous support throughout my Ph.D. journey. Their expertise and encouragement have been invaluable.

I owe a special debt of gratitude to my collaborators and all current and former colleagues, whose contributions through insightful discussions, shared ideas, and camaraderie have enriched my research and made this journey deeply fulfilling. I am also thankful to my friends at Brown University and beyond, whose kindness and encouragement have helped me overcome challenges along the way. Special thanks

to the Division of Applied Mathematics at Brown University for their invaluable guidance and support throughout these years.

To my family, thank you for your unconditional love, unwavering belief in me, and endless support. Your sacrifices and encouragement have been my greatest source of strength and motivation.

Finally, I express my sincere appreciation to the funding agencies and institutions that have supported my research. This work would not have been possible without their generous resources and support. These include:

1. the Division of Applied Mathematics and the Graduate School at Brown University,
2. the ARO MURI project (“Fractional PDEs for Conservation Laws and Beyond: Theory, Numerics and Applications”, W911NF-15-1-0562),
3. the University of Rochester CRCNS subaward (“CRCNS Research Proposal: Waste-clearance flows in the brain inferred using physics-informed neural networks”, R01AT012312), and
4. the ONR Vannevar Bush Faculty Fellowship (“The Next Generation of Operator Regression Networks: Theory, Algorithms, Applications”, N00014-22-1-2795).

I would also like to acknowledge the support from ANSYS and PredictiveIQ.

Contents

Vita	iv
Acknowledgments	x
1 Introduction	1
2 Multi-head physics-informed neural networks for learning functional priors and uncertainty quantification	10
2.1 Introduction	11
2.2 Learning functional priors via multi-head architectures and normalizing flows	15
2.2.1 Multi-head physics-informed neural networks (MH-PINNs) . .	15
2.2.2 Estimating the distribution of the head via normalizing flows (NFs)	18
2.3 Leveraging functional priors in solving downstream tasks	20
2.3.1 Regularization in optimization problem	21
2.3.2 Prior distribution in Bayesian inference	21
2.4 Results	23
2.4.1 Function approximation	24
2.4.2 Nonlinear ODE system	28
2.4.3 Reaction-diffusion equation	32

2.4.4	2D steady-state nonlinear Allen-Cahn equation	34
2.4.5	A high-dimensional parametric Helmholtz equation	37
2.5	Summary	40
3	Leveraging viscous Hamilton-Jacobi (HJ) PDEs for uncertainty quantification in scientific machine learning	42
3.1	Introduction	43
3.2	Connecting viscous HJ PDEs to Bayesian inference in machine learning	46
3.2.1	Bayesian inference in machine learning	46
3.2.2	Connection to viscous HJ PDEs	47
3.3	Riccati-based methodology	51
3.3.1	Connection using Gaussian prior to Riccati ODEs	52
3.3.2	Updating the likelihood	54
3.3.3	Updating the prior: tuning hyperparameters	58
3.4	Examples in scientific machine learning	59
3.4.1	Solving a boundary-value ODE problem	61
3.4.2	Solving the 1D steady-state advection-diffusion equation . . .	66
3.4.3	Solving the 2D Helmholtz equation	72
3.5	Summary	78
4	Quantifying uncertainty arising from noisy inputs-outputs	80
4.1	Introduction	81
4.2	Encoding physics in scientific machine learning	84
4.2.1	Physics-informed neural networks	86
4.2.2	Neural operators	87
4.3	Uncertainty quantification for noisy inputs-outputs	88
4.3.1	Physics-informed neural networks	91
4.3.2	Neural operators	94

4.3.3	Posterior estimate	95
4.4	Numerical examples	96
4.4.1	A nonlinear 1D Poisson equation with PINNs	97
4.4.2	Inverse ODE problem in diesel engine system	102
4.4.3	A nonlinear 2D steady-state Allen-Cahn equation with PINNs	105
4.4.4	1D Burgers' equation with FNOs	107
4.4.5	1D time-dependent reaction-diffusion equation with multi- input DeepONets	110
4.4.6	120-dimensional Darcy problem with DeepONets	112
4.5	Summary	114
5	Physical model misspecification and uncertainty	117
5.1	Introduction	118
5.2	Model misspecification in PINNs for learning governing equations	121
5.2.1	Learning governing equations from data using PINNs	121
5.2.2	Correcting the model misspecification in PINNs	123
5.2.3	Uncertainty quantification	126
5.3	Results and discussion	129
5.3.1	Pedagogical example: An ODE system	129
5.3.2	Reaction-diffusion equation	138
5.3.3	2D non-Newtonian flows	141
5.4	Summary	149
6	NeuralUQ: A comprehensive library for uncertainty quantifica- tion in scientific machine learning	151
6.1	Introduction	152
6.2	Uncertainty quantification for scientific machine learning	155
6.2.1	Scientific machine learning	155

6.2.2	Uncertainty quantification	158
6.2.3	Performance evaluation and post-training calibration	162
6.3	Overview and details of NeuralUQ	163
6.3.1	Library overview	163
6.3.2	Library design	165
6.3.3	Details of NeuralUQ components	166
6.3.4	Customization	171
6.4	Demonstration examples	173
6.4.1	Kraichnan-Orszag system	174
6.4.2	SIRD model for COVID-19	177
6.4.3	Inverse KdV problem	181
6.4.4	100-dimensional Darcy problem	182
6.5	Summary	186
7	Summary and Future Work	189
	Appendix	196
A	Appendix for Chapter 2	196
A.1	Details of NN architectures and training hyperparameters	197
A.2	Details for performing Bayesian inference	199
A.3	Multi-task learning with multi-head neural networks	199
A.3.1	Basis function learning and synergistic learning	200
A.3.2	Multi-Task Learning (MTL) versus Single-Task Learning (STL)	202
B	Appendix for Chapter 3	205
B.1	Details of the hyperparameters in the numerical examples	206
B.2	Additional results for example 1	206
B.3	Additional results for example 2	208

C	Appendix for Chapter 4	209
C.1	Details of hyperparameters in the numerical experiments	210
C.2	Additional results	213
C.2.1	1D nonlinear Poisson equation with PINNs	213
C.2.2	1D time-dependent reaction-diffusion with DeepONets	214
D	Appendix for Chapter 5	216
D.1	Details of hyperparameters in numerical experiments	217
D.2	Additional results for the ODE system	218
D.3	Additional results for the 1D reaction-diffusion equation	219
D.4	Details for the computations in lattice Boltzman equation model	221
E	Appendix for Chapter 6	222
E.1	Glossary for frequently used terms	223
E.2	Customization of surrogates	223
E.3	Customization of prior and inference	225
E.4	Data generation	227

List of Tables

2.1	Computational cost for different NFs and PI-GANs to approximate the Gaussian process defined in (2.16) with correlation length $l = 0.1$. The proposed method using MH-PINNs and NFs is a two-step method and hence its computation is decomposed into two parts: training MH-PINNs referred to as stage (1) and training NFs referred to as stage (2). Sampling time is evaluated as the average time taken to generate 10,000 samples of u in (2.15).	29
2.2	Estimate of λ and L_2 relative error of u for the downstream inverse problem on (2.15) using our approach and the vanilla PINN method. The reference value for λ is 2.3609. For the noisy-data case, the error is evaluated on the predicted mean.	31
2.3	L_2 relative error of u for solving (2.17) with sparse data of the initial condition using our approach and the vanilla PINN method. For the noisy-data case, the error is evaluated on the predicted mean.	34
2.4	L_2 relative error of u for solving (2.19) with sparse data of the source term using our approach and the vanilla PINN method. For the noisy-data case, the error is evaluated on the predicted mean.	36
2.5	L_2 relative error of u for solving (2.21) with sparse data of the source term using our approach and the vanilla PINN method. For the noisy-data case, the error is evaluated based on the predicted mean.	40

3.1	Relative L_2 errors of the predicted means of u and f when using continual learning and our Riccati-based approach to solve (3.17) after different amounts of noisy measurements of f are incorporated into the learned models. We achieve high accuracy inferences after all the data is incorporated.	62
3.2	Relative L_2 errors of the predicted means of u and f using different amounts of noisy measurements of f to solve (3.20). Our Riccati-based approach updates the learned models without requiring retraining when new data becomes available or storing and processing large datasets in their entirety.	67
3.3	Relative L_2 errors of the predicted means of u and f when solving (3.22) using our Riccati-based approach and the sequential order of integration shown in Figure 3.8(a). The errors significantly reduce as more subdomains are visited but remain slightly high after all of the data is incorporated due to the high level of noise in the data.	73
4.1	Interpretations of inputs, outputs and NN models $\mathcal{H}$ in PINNs and NOs. We note that when solving hybrid problems with NOs, $\mathcal{G}$ is pre-trained, often offline based on clean and simulated data, and therefore remain fixed in reconstructing v and u from their noisy measurements. In contrast, PINNs are directly trained with the noisy inputs-outputs data.	89
4.2	PINN for Poisson equation: Inference of u and f with different training data for f . The error is defined as the relative L_2 error between the exact and the prediction.	97
4.3	Errors of solving (4.15) with noisy inputs-outputs data of f using different methods. The error is defined as the relative L_2 error between the exact and the predicted mean.	98
4.4	Inferences of λ (mean $\pm$ standard deviation) and relative L_2 errors of the predicted mean for u using different methods to solve the inverse problem. The reference solution for λ is 0.1.	101

4.5	Inferences of c_1, c_2 (mean $\pm$ standard deviation) for the inverse ODE problem given noisy inputs-outputs. The reference values for c_1 ($\times 10^{-4}$) and c_2 ($\times 10^{-3}$) are -2.0817 and -3.4465 , respectively [290].	102
4.6	Inferences of D and λ (mean $\pm$ standard deviation) and relative L_2 errors of the predicted mean for u . The reference values for D ($\times 10^2$) and λ are 1.	105
5.1	ODE system: Predictions for ϕ , u and f and the reconstruction $\tilde{u}$. The metric is relative L_2 error, and errors are computed between the reference and the mean of 20 PINNs trained independently. The exact value of λ is 1.5 and $\tilde{\lambda}$ denotes the PINN estimate of λ ($\tilde{\lambda} = 0.2$ is fixed in Case (C)). We note that $\tilde{u}$ is reconstructed by solving the identified ODE system with the identified ϕ , the inferred f and the initial condition u_0 . Cubic spline data interpolation is employed to approximate the function f from the prediction of f on a uniform grid.	131
5.2	ODE system: Relative L_2 error for the reconstructed u from B-PINNs. Here the computational errors are calculated using the predicted mean from B-PINNs.	134
5.3	ODE system: Discovered equations from $\phi(u)$ using a symbolic regression method [58] (see Fig D.1 for plots of the discovered models). In all three cases, the symbolic regression is performed on the predicted mean from ensemble PINNs and B-PINNs.	137
5.4	2D non-Newtonian Cavity flow: Errors of the predicted means from ensemble PINNs. The error of u/v is defined as the relative L_2 error, while the error of f_x/f_y the mean squared error since the reference solutions are $f_x = 0$ and $f_y = 0$.	148

6.1	Summary of the UQ methods implemented in NeuralUQ. NeuralUQ includes three different UQ method families i.e., <i>Samplable</i> , <i>Variational</i> , and <i>Trainable</i> . Each UQ method family has various processes and inference methods, e.g., BNNs+HMC in Samplable; BNNs+MFVI Variational; and FNN+DEns in Trainable. Gens: Generators in GANs. Short and detailed descriptions for these inference methods are provided in Table 6.2 and [234, 252, 311, 125, 23], respectively. Note that BNNs/Generators/FNNs combined with known ODEs/PDEs are used for UQ for neural ODE/PDE problems, while BNNs/Generators combined with pretrained DeepONets and DeepONets using DEns are used for UQ in neural operator problems. . . .	166
6.2	Overview of the inference methods in NeuralUQ; adopted from [234]. HMC, MALA, LD, NUTS, and MFVI are inference methods corresponding to the Bayesian framework in Table 6.1; and DEns to the deterministic one. Note that MCD can be interpreted and implemented in both frameworks [89]. In the present study, we treat it as a Bayesian method.	170
6.3	Inverse Kraichnan-Orszag problem: Predictions corresponding to a and b using different methods. The reference values are $a = b = 1$. .	176
6.4	Inverse Kraichnan-Orszag problem: Performance evaluation of the considered methods on test dataset. Note that NLL is computed using the total uncertainty. Smaller RL2E and NLL mean better accuracy for the predictions of mean and uncertainty, respectively. RL2E corresponds to relative L_2 error, whereas NLL to the negative log likelihood.	177

6.5	Inverse Kraichnan-Orszag problem: Computational time for HMC and DEns for the case in Figure 6.5. The computational time for a deterministic PINN model is also included for comparison. For HMC, the number of burn-in steps is set to 3,000, the step size is 0.01, and the leap-frog step is 50. Further, the adaptive step size technique of [9] is also utilized to achieve acceptable acceptance rates. We collect 1,000 posterior samples after the burn-in phase to compute the predicted means and uncertainties in Figure 6.5. For DEns, 50 PINNs are trained simultaneously and independently, using the vectorization techniques of [1]. Further, we employ the Adam optimizer [152] for the training of PINNs with 20,000 iterations in both DEns and the deterministic PINN method. We note that this problem falls into the small-data regime, and thus full batch training is employed in all three methods. All the computations are performed on one AMD Ryzen 9 6900HS CPU. Additional implementation details can be found in https://github.com/Crunch-UQ4MI	177
6.6	Inverse KdV problem: Predictions corresponding to λ_1 and λ_2 using HMC and DEns; the reference values are $\lambda_1 = 1.5$ and $\lambda_2 = 0.25$. . .	182
A.1	L_2 relative errors, from MTL, for 1,000 tasks, using different initialization methods. The errors are displayed in the format of mean $\pm$ standard deviation, computed over all tasks.	201
A.2	L_2 relative errors of f , from MTL with MH-NNs and STL with NNs, on 100 tasks. Different initialization methods are used for the heads in MH-NNs. The errors are displayed in the format of mean $\pm$ standard deviation, computed over all 100 tasks.	203
A.3	L_2 relative errors of f , from MTL with MH-NNs, on 500 tasks equipped with insufficient data. The errors are displayed in the format of mean $\pm$ standard deviation, computed over all 500 tasks.	204

B.1	Step size h of RK4 when tuning σ in the hyperparameter tuning example in Section 3.4.1.2. $\sigma_0 \rightarrow \sigma_1$ denotes the case where σ is tuned from σ_0 to σ_1	206
D.1	Different initial and boundary conditions for the reaction-diffusion equation. The initial/boundary conditions are for Case (C) of Section 5.3.2.	220
D.2	The identified mathematical expressions of the reaction term $\phi(u)$ in the reaction-diffusion equation (5.11) using symbolic regression and data from multiple initial/boundary conditions. The exact is $2u(1-u)$. The data for the symbolic regression are displayed in Figure D.2. . .	221
E.1	Glossary of frequently used terms in this chapter.	223
E.2	Nonlinear diffusion-reaction problem: Predictions for k_r with different priors. The reference solution is $k_r = 0.2$	226

List of Figures

1.1	A qualitative breakdown of <i>total</i> uncertainty describing the contributions from data (noisy, gappy); physical models (misspecification, stochasticity); noise models (misspecification); neural networks (architecture, hyperparameters, overparameterization); and posterior inference. Aleatoric uncertainty is due to noisy data and cannot be reduced. Epistemic uncertainty is due to noisy and limited data, as well as neural network overparameterization. Note that all uncertainties in this figure are either modeled, addressed, or discussed in this dissertation.	4
1.2	(a) An example of approximating a function from its noisy and gappy measurements (adapted from [315]). (b) Utilizing an informative functional prior in solving a time-dependent PDE with sparse and noisy measurements of the initial condition (adapted from Section 2.4.3 of Chapter 2). (c) An example of ignoring and modeling the input noise when solving an inverse PDE problem with noisy inputs-outputs data (adapted from Section 4.4.1 of Chapter 4). (d) An example of physical model misspecifications in PINNs, in which a Newtonian flow is assumed while the data comes from a non-Newtonian flow (Section 5.3.3.1 of Chapter 5).	6

- 2.1 A schematic view of the proposed approach for learning functional priors. In (a), the architecture of multi-head physics-informed neural networks (MH-PINNs) with M heads is displayed. All M functions share the same body NN, denoted as **Body**, and each is associated with a head NN, parameterized by H^k , $k = 1, \dots, M$, and a specific PINN task $\mathcal{T}^k$ with the loss function $\mathcal{L}^k$ and data $\mathcal{D}^k$. These functions $u^k, k = 1, \dots, M$ are solutions to M ODEs/PDEs, which may differ in the source term f^k , boundary term b^k , and/or model parameter λ^k (see (2.1) for the problem formulation). In (b), we present the breakdown of our approach: (1) training a MH-PINN to obtain a shared representation (body NN), denoted as **Body**^{*}, and multiple heads, denoted as $\{H^{k,*}\}_{k=1}^M$, and (2) estimating the distribution of the head, denoted as $p(H)$, from these heads via density estimators such as normalizing flows NF. The estimated density is denoted as $p_{\phi^*}(H)$. Then the functional prior is represented by the learned body NN and the estimated density. (A NN parameter with a superscript * indicates that it has been trained.) 13
- 2.2 An illustration of a normalizing flow consisting of K invertible transforms, denoted by $T_i, i = 1, \dots, K$ (adapted from [220]). Here $\mathbf{z}_0$ represents the base random variable whose distribution is tractable and known, and $\mathbf{z}_K$ represents the target random variable (H) whose distribution is to be estimated. 18

2.3	Results for approximating the stochastic function defined in (2.14) (a functional prior) and solving the downstream few-shot regression tasks. (a) Left : 1,000 samples generated from the exact distribution; middle : 1,000 samples generated from the learned generator; right : statistics computed from samples in which we refer to the interval of mean ± 2 standard deviations as bound. (b)/(c) Results for the downstream tasks. Left : results for clean-data cases using our method, the transfer learning (TL) method in [71], and regular NN method; middle : results for noisy-data cases using our method with HMC for posterior estimation and UQ; right : results for the same noisy-data cases using our method with LA for posterior estimation and UQ.	25
2.4	Results for regression on an out-of-distribution (OOD) function: (a) a clean-data case in which (2.10) is minimized, and (b) a noisy-data case in which the posterior distribution (2.11) is estimated with HMC for UQ.	27
2.5	Approximating the Gaussian process defined in (2.16) with different correlation lengths (l) as the source term in (2.15) using different NFs: spectra of the correlation structure from the learned functional priors. We also compare to the PI-GAN method [317]. The covariance matrix is constructed using 10,000 generated samples and eigen-values are averaged over ten functional priors trained independently.	28
2.6	Results for the inverse problems of the ODE system (2.15) with initial conditions hard-encoded in the NN modeling. (a) Left : 1,000 samples of f , generated from the exact distribution; middle : 1,000 samples of f , generated from the learned functional prior; right : statistics computed from samples. The bound is defined as the same as in the caption of Figure 2.3. (b) Predicted f and u using PINNs and our approach, for the downstream inverse problem with clean data. (c) Predicted means and uncertainties of f and u using our approach, for the downstream inverse problem with noisy data.	30

2.7	Solving (2.17) with sparse data of the initial condition u_0 by leveraging learned prior using the proposed method. The boundary conditions are hard-encoded. (a) Left: 1,000 training samples of u_0 ; middle: 1,000 samples of $u(0, \cdot)$ from the learned generator; right: statistics computed from samples. The bound is defined as the same as in the caption of Figure 2.3. (b) Predicted u at $t = 0, 0.5, 1$ using our approach and the PINN method with clean measurements. (c) Predicted means and uncertainties of u at $t = 0, 0.5, 1$ using our approach with HMC for posterior estimate, with noisy measurements.	33
2.8	Results from solving (2.19) given sparse data of the source term (100 clean measurements randomly sampled across the domain): (a) inferences of u and (b) inferences of f . From left to right we present the reference of u/f , the inference of u/f using the presented method to leverage the prior knowledge, and the inference of u/f using the vanilla PINN method without leveraging the learned functional prior. Black crosses represent locations of the measurements of f	35
2.9	Solving (2.19) given data with high level noise: 100 noisy data of f are randomly generated across the domain and corrupted by additive Gaussian noise with mean zero and standard deviation 0.15 (approximate 100% relative noise level): (a) inferences of u and (b) inferences of f . From left to right we present the reference of u/f , the predicted mean of u/f , the absolute error of u/f , and the predicted uncertainty of u/f . Black crosses represent locations of the measurements of f	36
2.10	Results from solving (2.21) given sparse data of the source term (100 clean measurements randomly sampled across the domain): (a) inferences of u and (b) inferences of f . From left to right we present the reference of u/f , the inference of u/f using the presented method to leverage the prior knowledge, and the inference of u/f using the vanilla PINN method without leveraging the learned functional prior. Black crosses represent locations of the measurements of f	38

2.11	Solving (2.21) given data with high level noise: 300 noisy data of f are randomly generated across the domain and corrupted by additive Gaussian noise with mean zero and standard deviation 0.1 (approximate 100% relative noise level): (a) inferences of u and (b) inferences of f . From left to right we present the reference of u/f , the predicted mean of u/f , the absolute error of u/f , and the predicted uncertainty of u/f . Black crosses represent locations of the measurements of f . .	39
3.1	(See Section 3.2) Illustration of a connection between a Bayesian inference problem in scientific machine learning (top) and the solution to a multi-time viscous HJ PDE (bottom). The colors indicate the associated quantities between problems. This color scheme is reused in the subsequent illustrations of our connection. The arrow labels describe how the boxed quantities are related. For example, the posterior mean in the learning problem is equivalent to the spatial gradient of the solution to the multi-time viscous HJ PDE (red).	43
3.2	(See Section 3.2) Mathematical formulation of the connection between a Bayesian inference problem with linear model and Gaussian likelihood (top) and the solution to a multi-time viscous HJ PDE with quadratic Hamiltonian (bottom). The content of this illustration matches that of Figure 3.1 by replacing each term in Figure 3.1 with its corresponding mathematical expression. The colors indicate the associated quantities. The arrow labels describe how the boxed quantities are related.	48

3.3	(See Section 3.3.1) Mathematical formulation of the connection between a Bayesian inference problem with linear model, Gaussian likelihood, and Gaussian prior (top) and the solution to a multi-time viscous HJ PDE with quadratic Hamiltonian and quadratic initial condition (bottom). The content of this illustration is a special case of the connection in Figure 3.2, where the prior is Gaussian (set $g(\boldsymbol{\theta}) = \frac{1}{2}\langle \boldsymbol{\theta}, \Lambda^{-1}\boldsymbol{\theta} \rangle$ in Figure 3.2). The colors indicate the associated quantities. The arrow labels describe how the boxed quantities are related.	52
3.4	Results of solving (3.17) using continual learning and our Riccati-based approach. (a) and (b) show the predicted mean and uncertainty of u and f , respectively, after the 101th, 151th, and 201th noisy data point of f becomes available. Our Riccati-based approach naturally coincides with the continual learning framework while inherently avoiding catastrophic forgetting. Our approach allows us to incrementally update the learned models without accessing the historical data ($\bullet$), while also providing a quantitative metric for our confidence in the learned models in the form of the predicted uncertainties ($\blacksquare$). We observe that regions of low predicted uncertainty generally coincide with regions of high inference accuracy, which implies that this confidence metric is a good indicator of the reliability of the model. .	63

- 3.5 Results of tuning the standard deviation σ of the prior using our Riccati-based approach to solve (3.17). (a) shows how the validation error changes as σ is tuned. (b) shows the flow with respect to σ of the predicted mean of u at different times ($\tau = 0.25, 0.5, 0.75$). The black dot represents the value using the original choice of $\sigma = 1$, and the arrows represent the direction of the flow of solutions as we increase σ from 1 (**red**; i.e., we evolve the corresponding Riccati ODE backward) or decrease σ from 1 (**blue**; i.e., we evolve the corresponding Riccati ODE forward). In contrast to conventional ML approaches, our Riccati-based approach allows the hyperparameter to be tuned continuously without retraining on or access to the training data. . . 65
- 3.6 Results of solving (3.20) using large-scale data with high noise level and our Riccati-based approach. (a) shows the corresponding predicted mean and uncertainty (**■**) of u . (b) shows the inference/fitting of f using 1,000, 5,000, and 100,000 noisy data points. Note that we do not show the predicted uncertainty of f for clarity of presentation. Our Riccati-based approach updates the learned models incrementally using information from one data point (**•**) at a time, which provides significant computational and memory savings in big data regimes. . . 68

3.7	Intermediate result for adding the 46th data point when using active learning and our Riccati-based approach to solve (3.20). The left side of (a) and (b) shows the inferences of f and u , respectively, before the placement of the new sensor and incorporation of the 46th data point. The right side of (a) and (b) shows the updates to f and u after these actions. The vertical black dashed line represents the location where the 46th sensor will be placed. Note that we zoom in on the region around the new sensor instead of showing the full domain for clarity of comparison. Following an active learning framework, we repeat the following steps: compute the predicted mean and uncertainty of u and f , place a new sensor at the location of the largest predicted uncertainty (■) of f , and obtain a new measurement of f (*) at the new sensor location. Our Riccati-based approach naturally complements this framework as it can efficiently perform repeated updates of the predicted mean and uncertainty.	70
3.8	Results of solving (3.22) with noisy data of f using our Riccati-based approach while traversing the domain sequentially. (a) displays the 7×7 uniform domain decomposition and the sequential order of integration of the domain. (b) shows the absolute errors of the predicted means of u and f after all 49 subdomains have been visited. (c) and (d) show 1D slices of the inferences and predicted uncertainties (■) of u and f , respectively. These results show how our Riccati-based approach can naturally be extended to solve higher-dimensional problems. Namely, by leveraging data streaming, our Riccati-based approach is able to overcome the computational and memory challenges related to the higher dimension of the PDE and subsequently increased dataset size.	74

3.9	Intermediate results of solving (3.22) using our Riccati-based approach while traversing the domain sequentially. We collect data from each subdomain sequentially following the pattern in Figure 3.8(a). (a) and (b) shows the absolute errors of the inference of u and f , respectively. The white crosses indicate the subdomains that have already been visited. Our Riccati-based approach incrementally updates our predictions as we traverse the domain, which provides computational and memory advantages over having to process or store all of the data from the entire domain at once.	75
3.10	Domain decomposition with multi-level subdomain traversal. The level one subdomain is marked in red , the level two subdomains are marked in blue numbers, the level three subdomains are marked in green numbers, and the level four subdomains are marked in black	76
3.11	Intermediate results of solving (3.22) using our Riccati-based approach while traversing the domain using the multi-level order in Figure 3.10. (a) and (b) show the absolute errors of the inference of u and f , respectively. The level one subdomain is marked by the white cross, the level two subdomains are marked by white stars, and the level three subdomains are marked by white circles. Our Riccati-based approach incrementally incorporates data points into the predictions invariant to the order of the data. As such, the end result after all the data is incorporated is identical to Figure 3.8(b)-(d), despite the intermediate results differing.	77

4.1	A schematic view of two prevalent ways of encoding physics in SciML. In (a) , we present the framework of physics-informed neural networks (PINNs) [244], where the physics is encoded by modeling $u(x)$ using a NN, parameterized by θ and denoted by $\mathcal{M}_\theta$, and approximating f, b via automatic differentiation and the ODE/PDE. In (b) , we represent the physics by pretrained neural operators (NOs), which are trained offline to learn the solution operators of (4.2), often based on clean and simulated data. The pretrained NOs effectively encode the underlying physics within the NN architecture and parameters [160], while in the downstream tasks such as solving ODEs/PDEs and tackling hybrid problems, they remain fixed and serve as equation-free surrogates. Here $\mathbf{v}_\psi$ and $\mathbf{w}_\psi$ denote discretizations of the input functions $v(\xi^v)$ and $w(\xi^w)$, respectively, which are parameterized by ψ . Note that in these two methods, the inputs and outputs (color-coded by blue and red, respectively) differ, making it crucial to develop uncertainty quantification (UQ) approaches to handle noisy inputs-outputs in both scenarios.	85
4.2	An example of using pretrained NOs to encode physics in the hybrid problem. In particular, we use a multi-input DeepONet [138] to represent the physics which connects u, v and w . Here, the multi-input DeepONet, denoted by $\mathcal{G}$, takes as input w and v and outputs u , while $\otimes$ denotes elementwise multiplication. In the hybrid problem, u, v, w are reconstructed with UQ from their noisy measurements and the physics, represented by the pretrained multi-input DeepONet, enabling synergistic learning, as shown in the predicted mean and uncertainty. See Section 4.4.5 for details.	87

4.3	Results from solving (4.15) in the B-PINN framework based on noisy inputs-outputs data of f using different approaches. The boundary condition is clean and hard-encoded in the modeling. From left to right are results from the original B-PINN method which ignores the noise in x , our approach which models the noise in x , and the recasting method which recasts the noise in x as a heteroscedastic noise in f via first-order Taylor expansion.	98
4.4	Results from solving (4.15) with λ unknown based on noisy inputs-outputs data of u and noisy outputs data of f . The boundary condition is clean and hard-encoded in the modeling. We note that in this case data of u are noisy in both the input (heteroscedastic noise; defined in (4.16)) and the output while data of f are noisy in the output but clean in the input, demonstrating the flexibility of our approach. From left to right are results from the original B-PINN method, our approach, and the recasting method.	100
4.5	Predictions for (a) p_{im} and (b) p_{em} for solving the inverse ODE problem in the diesel engine with noisy inputs-outputs data. Here we employ the original B-PINN method which ignores the input noise (left) and our approach which models the input noise (right). 40 noisy inputs-outputs data of p_{im} and p_{em} and 40 noisy outputs data of f_{im} and f_{em} are used as the training data.	102
4.6	Results from solving the inverse problem of (4.22) based on noisy inputs-outputs data with D and λ unknown. In (c) we show the locations of the measurements of u , in which blue circles represent the actual locations and black crosses represent the measured locations.	105

4.7	Burgers' equation: Reconstructing $u _{t=0}$ and $u _{t=1}$ from their noisy measurements and a pretrained FNO, which represents the physics and serves as an equation-free surrogate. From left to right are results from our approach with correctly specified input noise, the reference method, and our approach but misspecifying the input noise to a small value (0.001) in constructing the likelihood. Here, the reference method refers to the deterministic inference in which a maximum a posterior (MAP) estimate is performed without UQ.	108
4.8	Burgers' equation: Comparison between our approach which reconstructs $u _{t=0}$ and $u _{t=1}$ synergistically (left) and a non-synergistic method (right), in which these two functions are reconstructed independently from their respective sparse and noisy measurements without access to data of the other function.	108
4.9	1D time-dependent reaction-diffusion equation with heteroscedastic diffusion term: Reconstructions with uncertainties of u, k, f , as displayed in (a) and (b), from their sparse and noisy measurements and a pretrained multi-input DeepONet, which represents the physics in (4.25). White circles represent locations of measurements of u . In (c), a non-synergistic method is employed to reconstruct k and f independently from their respective measurements without access to data of the other function or data of u	110
4.10	120-dimensional Darcy problem: Reconstructions with uncertainties of $\log(\lambda)$ (shown in (a)) and u (shown in (b)) from their sparse and noisy measurements as well as a pretrained DeepONet representing the physics in (4.27). The white cross stands for the location where λ is observed while the white circle stands for the location where u is observed.	113

5.1	A schematic view of PINNs for learning governing equations from data. On the left, a general framework of PINNs is displayed where σ denotes the activation function, θ denotes the DNN parameters, u_θ denotes the output of the DNN that is used to approximate u , and f_θ/b_θ are computed via automatic differentiation [244]. An example on learning governing equations with PINNs is illustrated on the right panel. Here a 1D Burgers equation is learned ($\lambda_1 = 0, \lambda_2 = 1, \lambda_3 = -\nu$ where ν denotes the viscosity) given data on u	121
5.2	In (a), a breakdown of total uncertainty in PINNs is presented. In (b), results from PINNs encountering model misspecifications are displayed, where a physical model of Newtonian fluid is assumed but the real physics, which generates the data, is non-Newtonian. The PINN method is employed to identify the homogeneous viscosity with different belief weights in the loss function, i.e. pairs of w_{PDE} and $w_{\mathcal{D}_u}$ in (5.2), resulting in different pairs of PDE loss $\mathcal{L}_{PDE}$ and data loss $\mathcal{L}_{\mathcal{D}_u}$, shown in the left of (b). Due to the model misspecification, the PDE loss $\mathcal{L}_{PDE}$ and the data loss $\mathcal{L}_{\mathcal{D}_u}$ cannot be minimized simultaneously. Three results are presented in the right of (b), corresponding to three cases in the left, + (data are fitted well but the physics is not satisfied), * (the physics is satisfied but data are not fitted), and x (neither data nor physics are fitted). We note that the boundary condition is hard-encoded in the modeling. Details can be found in Section 5.3.3.1.	123

5.3	An overview of the proposed approach for correcting the misspecified physics. Here we assume that the nonlinear differential operator $\mathcal{F}$ is misspecified as $\tilde{\mathcal{F}}_\lambda$ such that $\tilde{\mathcal{F}}_\lambda[u] \neq f$, which leads to poor performance of PINNs in learning governing equations. Compared to the conventional PINN method shown in Figure 5.1, an additional DNN, parameterized by ψ , is used to model the discrepancy. We note that s_ψ serves two roles: (1) correcting the equation such that data of u can be explained, and (2) modeling the discrepancy of the physics. With posterior samples of ψ , we obtain the prediction as well as the model uncertainty of the discrepancy. We also note that DNNs in this figure, namely u_θ and s_ψ , are deterministic NNs in ensemble PINNs and Bayesian NNs (BNNs) in B-PINNs.	124
5.4	ODE system: Predictions for u , f and ϕ from ensemble PINNs with clean and sufficient data. Ensemble PINNs with (a) the correct physical model, (b) a misspecified physical model, and (c) the misspecified physical model plus a correction. Blue circles: Representative measurements for u and f ; Red dashed line: Predicted mean; Black solid line: Reference solution.	132
5.5	ODE system: Predictions for u , f and ϕ from B-PINNs with noisy and gappy data. B-PINNs with (a) correct physical model, (b) misspecified physical model, and (c) misspecified physical model plus a correction. Blue circles: Noisy measurements for u and f ; Red dashed line: Predicted mean from B-PINNs; Black solid line: Reference solution. Our approach alleviates the model misspecification issue significantly, providing more accurate predictions. Compared to correctly specified model, the increase in the predicted uncertainty is caused by the lack of knowledge in the physical model and could be considered as model uncertainty.	133

5.6	ODE system: Solving the identified system with the inferred f and ϕ from B-PINNs with the initial condition $u_0 = 0$. The inferred f and ϕ come from (a) correct physical model, (b) misspecified physical model, and (c) our approach. Red dashed line: Predicted mean; Black solid line: Reference solution. It shows that the misspecified physical model leads to wrong reconstruction $\tilde{u}$ while our approach is able to correct it. Due to the lack of knowledge of the model, our approach yields larger uncertainty than the correct physical model.	134
5.7	Time-dependent reaction-diffusion system: Predicted mean for u and ϕ from B-PINNs with noisy data of u and f and differently specified reaction models. Misspecified model (u^2) impairs the performance of B-PINNs significantly in terms of the accuracy in predicting u and ϕ . and the proposed approach is able to alleviate the issue caused by model misspecification; see Figure 5.8 for UQ.	139
5.8	Time-dependent reaction-diffusion system in slices ($t = 0.5, 0.75$): Predicted uncertainty for u and ϕ from B-PINNs with noisy data of u and f and differently specified reaction models. Representative predicted uncertainties for (a) u , and (b) ϕ . Red dashed line: Predicted mean from B-PINNs; Black solid line: Reference solution. Our approach is able to improve the the predicted mean and the predicted uncertainty significantly. The decrease in the accuracy of predicted mean and increase in the predicted uncertainty of our approach are caused by the lack of knowledge over the reaction model.	140
5.9	2D non-Newtonian channel flow: Correcting the misspecified physics with two different approaches, i.e. (a) modeling the discrepancy in the equation $s(y)$ with a DNN, and (b) modeling the unknown space-dependent viscosity $\mu(y)$ with a DNN. In (c), the PINN training losses are presented and they converge to smaller values for both $\mathcal{L}_{\mathcal{D}_u}$ and $\mathcal{L}_{PDE}$ in the case where we add a DNN as correction as compared to the one in which we use a DNN to approximate the viscosity.	143

5.10	2D non-Newtonian channel flow: Predictions for u , f and the discrepancy s from B-PINNs with noisy data. B-PINNs with (a) misspecified model, and (b) misspecified model plus a correction.	145
5.11	Non-Newtonian cavity flow: Predictions from ensemble PINNs for u and v in contours. Red dashed line: Predicted mean; Black solid line: Reference solution.	148
5.12	Non-Newtonian cavity flow: Predictions for u, v, f_x, f_y in 1D slices. Red dashed line: predicted mean; black solid line: reference solution; uncertainty bound: predicted uncertainty.	149
6.1	Schematic overview of a SciML problem scenario. As shown in (a), SciML problems of interest include, but are not limited to, forward, inverse and mixed ODEs/PDEs, and operator learning problems. Next, in the pie chart of (b) a qualitative breakdown of total uncertainty is shown, describing the contributions from data (noisy, gappy); physical models (misspecification, stochasticity); neural networks (architecture, hyperparameters, overparameterization); and posterior inference. Note that we mainly address two types of uncertainty in this chapter, i.e., aleatoric (data noise) and epistemic (NN overparameterization and limited data) uncertainties. The latter is related to the complex loss landscapes and random initialization of NNs. Addressing physical model or NN architecture uncertainties, indicatively, is left as future work. As shown in (c), active learning for improving the predictions can subsequently be performed, i.e., to acquire new inputs according to an acquisition function that is based on the output uncertainty from part (b).	153

6.2	A unified view of the UQ models studied in Section 6.2.2; adopted from [234]. The model $\mathcal{M}_1$ can be a NN or a generator network, or nothing. The model $\mathcal{M}_2$ can be a PDE or an operator, such as a pretrained DeepONet. The quantities ϕ , ψ can be the solution u and the source f in (6.1), or vice-versa; x represents the space-time input domain; and θ represents the parameters of the NN or the input to the generator. Further, “ χ ” is used for modeling additional fields and can represent, for example, the model parameters λ in (6.1). The dashed arrow indicates that the connection may not exist for some problem scenarios.	159
6.3	Library design overview. In the design phase, a model is constructed that includes processes representing a proposed solution family and likelihoods or loss functions representing the data distribution. Processes include surrogates, which can be construed as assumed functional forms, and priors distributions or variables for the parameters of the surrogates, depending on the type of method used. Likelihoods or loss functions are constructed using the data and the underlying neural ODE/PDE. To select the solution function samples that best explain the data, a posterior inference method is combined with the model. Finally, the obtained function samples can be calibrated using additional data. In the revision phase, metrics are used for evaluating the performance and re-designing the solution procedure.	164
6.4	A breakdown of the Process module. A Process instance contains a surrogate as well as the attributes for its parameters. Processes, used in different UQ method families, have different surrogates and attributes for their parameters, which are denoted with different line colors in this figure. Different combinations of the components yield different processes, e.g., an fnn surrogate combined with a prescribed prior distribution for its parameters results in a BNN.	167

6.5	Inverse Kraichnan-Orszag problem: Predictions corresponding to x_1 , x_2 , and x_3 using HMC (a) and DEns (b). Epistemic uncertainty is denoted by $2\bar{\sigma}_e$, whereas total uncertainty by $2\bar{\sigma}$	175
6.6	Inverse Kraichnan-Orszag problem: Predictions for x_1 with different numbers of measurements on x_1 , using (a) HMC and (b) DEns. In each row, the numbers of training data for x_1 are 11, 22, and 44, from left to right. Epistemic uncertainty is denoted by $2\bar{\sigma}_e$, whereas total uncertainty by $2\bar{\sigma}$. The predicted epistemic uncertainty decreases as we increase the number of training data, whereas the uncertainty arising from data noise, i.e., aleatoric uncertainty, cannot be reduced by adding more training data.	178
6.7	Inverse SIRD problem: Predictions for I , R , D and also β , γ and ν , using HMC (a) and DEns (b). Total uncertainty is denoted by $2\bar{\sigma}$. .	179
6.8	Inverse KdV problem: Predictions for $u(x, t)$ at different times, as obtained by using HMC and DEns. (a) Locations of training data for u and f ; black circles are used for u and magenta cross symbols for f . (b) HMC; $t = -1.5$ in the left side and $t = 0.4$ in the right side. (c) DEns; $t = -1.5$ in the left side and $t = 0.4$ in the right side. Epistemic uncertainty is denoted by $2\bar{\sigma}_e$, whereas total uncertainty by $2\bar{\sigma}$. Note that the scaling for t is the same as in [21]. One could also use a different scaling for t to make it positive only.	180
6.9	100-dimensional Darcy problem (Case A): Predictions for $\log(\lambda)$ and u from PA-GAN-FP+HMC. Training data for λ are shown with circles, whereas training data for u with cross symbols. Total uncertainty is denoted by $2\bar{\sigma}$	184
6.10	100-dimensional Darcy problem (Case A): Predictions for $\log(\lambda)$ and u from PA-BNN-FP+HMC. Training data for λ are shown with circles, whereas training data for u with cross symbols. Total uncertainty is denoted by $2\bar{\sigma}$	185
6.11	100-dimensional Darcy problem: Predictions obtained from U-DeepONet and corresponding to u . Total uncertainty is denoted by $2\bar{\sigma}$	185

A.1	The effect of different initialization methods of the head, in basis functions learning, few-shot learning, and functional prior learning. (a) Samples of 20 basis functions from MH-NNs, trained for approximating 1,000 f generated from (2.14), using, from left to right, RN (0.05), GU and RN (1) initialization methods. (b) 1,000 training samples of f . (c) Results for two downstream few-shot regression tasks, using TL method without the regularization informed by the learned PDF, as opposite to the proposed approach. (d) Results for functional prior learning, using, from left to right, RN (0.05), GU and RN (1) initialization methods.	202
A.2	Results for three tasks with insufficient data from MTL with MH-NNs, using different initialization methods over the head, and from STL with NNs with the same architecture. We note that tasks with sufficient data and tasks with insufficient data are treated equally in MTL.	204
B.1	Deleting two outliers ($\times$) sequentially from the trained model using our Riccati-based approach to solve (3.17). The left column of (a) and (b) shows the inferences of u and f , respectively, after the initial training with all data points. The middle column shows the inferences after the first outlier is deleted, and the right column shows the inferences after two outliers are deleted. The outliers are deleted by solving the associated Riccati ODEs backward in time. This process only uses the results of the previous training step and information of the point to be deleted, which provides potential computational advantages over more standard SciML approaches that would otherwise require retraining on the entire remaining dataset.	207

B.2	Intermediate result for adding the 47th data point when using active learning and our Riccati-based approach to solve (3.20). The left side of (a) and (b) shows the inferences of f and u , respectively, before the placement of the new sensor and incorporation of the 47th data point. The right side of (a) and (b) shows the updates to f and u after these actions. The vertical black dashed line represents the location ($x = 0$) where the 47th sensor will be placed. Note that we zoom in on the region around the new sensor instead of showing the full domain for clarity of comparison. Our Riccati-based approach naturally complements this active learning framework as it can efficiently perform repeated updates of the predicted mean and uncertainty.	208
C.1	The effect of noisy inputs-outputs in solving (4.15) using PINNs. Three scenarios are considered: clean data, noisy output but clean input data, and clean output but noisy input data of f . The noises for input and output data are additive Gaussian noises with the same standard deviation 0.01. Comparison shows that noise in the input data has more significant effect upon the performance of PINNs than noise in the output data. L_2 regularization is employed to prevent overfitting in all two noisy-data scenarios. Quantitative results can be found in Table 4.2.	213
C.2	Noise models for the input of u considered in different PINN problems.	213
C.3	The homogeneous reaction-diffusion equation: Reconstructing f and $u _{t=1}$ from their sparse and noisy measurements, in which a pre-trained DeepONet serves as an equation-free surrogate that encodes the physics. From left to right we present results from our approach with correctly specified input noise, the reference method (a MAP estimate), our approach but misspecifying the scale of the input noise as a relatively small value (0.01), and a non-synergistic method. . . .	214

D.1	ODE system: Plots of the discovered equation $\phi(u)$ using symbolic regression, following our approach in correcting the model misspecification; see Table 5.3 for the mathematical expressions. In (a), the symbolic regression is performed on the predicted mean from ensemble PINNs. In (b) and (c), the symbolic regression is performed on the predicted mean as well as the upper and lower bounds of the uncertainty band ($\text{mean} \pm 2\text{stds}$) from B-PINNs. We note that the symbolic regression is performed on $\phi(u)$ instead of $\phi(t)$	218
D.2	Data for the identification of the reaction term $\phi(u)$, obtained from correcting the misspecified physics in three initial/boundary conditions (presented in Table D.1) independently using our approach. The exact ϕ - u relation is $\phi = 2u(1 - u)$	220
E.1	1D regression problem: Predictions for u using different surrogates. (a) BNN; (b) GAN-FP, and (c) $A \sin(\omega x)$. $2\bar{\sigma}$: Total uncertainty. . .	224
E.2	Nonlinear diffusion-reaction problem: Predictions for u and f from VI with different parameterized posteriors. (a) Mean-field Gaussian, (b) Mean-field Bernoulli.	226

CHAPTER ONE

Introduction

Accurate modeling and prediction of the dynamic response of physical systems remain an open scientific problem, despite decades of successful research. Persistent challenges include solving ill-posed problems, such as those with unknown parameters or boundary conditions, assimilating imprecise and/or incomplete data, and reducing computational costs, particularly for high-dimensional systems. In this regard, physics-informed machine learning and, more broadly, scientific machine learning (SciML) have emerged as promising interdisciplinary research areas. These approaches offer innovative tools for addressing many of the aforementioned challenges. For comprehensive reviews of recent advances in this field, see [142, 60, 7, 305, 281].

Machine learning, particularly deep learning which uses neural networks (NNs), excels at processing large volumes of data and making rapid and accurate predictions; see, e.g., [104, 122]. SciML builds on these capabilities by integrating observational data with available physical models. Broadly, SciML techniques can be classified into three main categories:

1. **Data-driven emulation of physical systems:** These approaches utilize large datasets to emulate physical systems. Examples include the deep operator network (DeepONet) developed by [178], with variations discussed in [107, 298] and theoretical insights provided by [165]. An alternative approach, the Fourier neural operator (FNO), is presented in [171].
2. **Physics-encoded architectures:** These methods incorporate physics directly into the NN architecture design, as exemplified by the work of [64, 65].
3. **Physics-informed optimization:** In these approaches, physical constraints are included as soft penalties in the NN optimization process. Notable examples include the physics-informed neural networks (PINNs) proposed by [244], with extensions detailed in [162, 269, 219, 134, 200, 151], and theoretical analyses

found in [266, 203].

These techniques are particularly advantageous in scenarios where conventional solvers face limitations. For instance, they are effective in addressing mixed problems with partial observational data and incomplete knowledge of the underlying physical system (e.g., [244]). They can seamlessly incorporate noisy or multi-fidelity data (e.g., [199, 198, 49, 50]) and generate physically consistent predictions, even for extrapolation tasks (e.g., [315]). Additionally, they eliminate the need for computationally expensive mesh generation (e.g., [244]) and benefit from open-source software that is continuously developed and extended (e.g., [180, 121]).

Quantifying prediction uncertainty, particularly in scenarios involving noisy or limited data and NN overparameterization, is crucial for the reliable application of deep learning in critical domains such as physical and biological systems. Among the various uncertainty quantification (UQ) methods in deep learning, the Bayesian framework has been the most prominent and widely studied [209, 184, 210, 19, 164, 95, 109, 211, 124, 182, 88, 308, 24, 133, 175, 325, 287]. Alternative approaches include methods based on ensembles of NN optimization iterates or independently trained NNs [163, 129, 189, 93, 149, 226, 185, 84, 119, 240, 216, 57], evidential frameworks [188, 264, 8, 44, 158, 43], combining NNs with Gaussian processes [61, 241, 167, 191, 4, 6], and the conformal prediction framework [10, 72]. While Bayesian methods and ensemble-based techniques are discussed extensively in this chapter, readers seeking additional insights are encouraged to consult recent review articles on UQ in deep learning [233, 141, 37, 102, 177, 272, 293, 94, 2, 131, 207, 336, 139].

Conventionally in machine learning and data science, uncertainty is often categorized into *aleatoric* uncertainty and *epistemic uncertainty* [2, 234]. Aleatoric uncertainty, also known as data uncertainty, arises from the inherent variability within the

data itself. This type of uncertainty is caused by random noise or variability in the data distribution and is considered irreducible, as it stems from intrinsic properties of the data rather than limitations of the model or knowledge. In contrast, epistemic uncertainty, often referred to as knowledge or model uncertainty, originates from a lack of understanding about the underlying processes or model. This type of uncertainty occurs when the model does not have enough data or information to make accurate predictions. While aleatoric uncertainty can only be mitigated by improving the quality of the data, epistemic uncertainty can be reduced through acquiring additional data, refining the model, or enhancing data quality [2, 234].

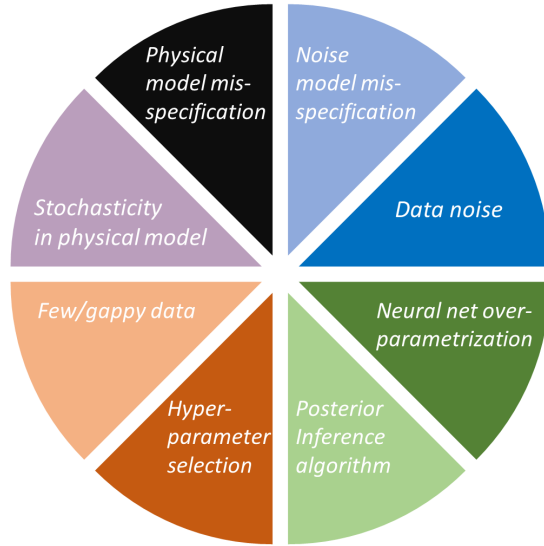


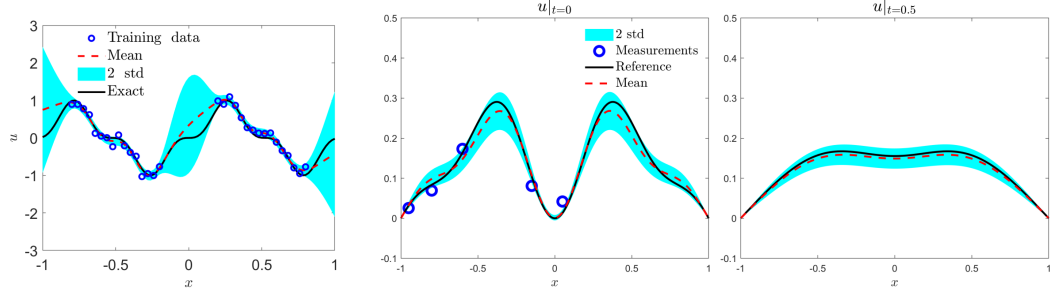
Figure 1.1: A qualitative breakdown of *total* uncertainty describing the contributions from data (noisy, gappy); physical models (misspecification, stochasticity); noise models (misspecification); neural networks (architecture, hyperparameters, overparameterization); and posterior inference. Aleatoric uncertainty is due to noisy data and cannot be reduced. Epistemic uncertainty is due to noisy and limited data, as well as neural network overparameterization. Note that all uncertainties in this figure are either modeled, addressed, or discussed in this dissertation.

In the context of SciML, which often involves differential equations such as ordinary/partial differential equations (ODEs/PDEs) with unknown or uncertain terms and parameters, UQ becomes an even more challenging task. Figure 1.1 summarizes the various sources of uncertainty typically considered in these settings. Despite

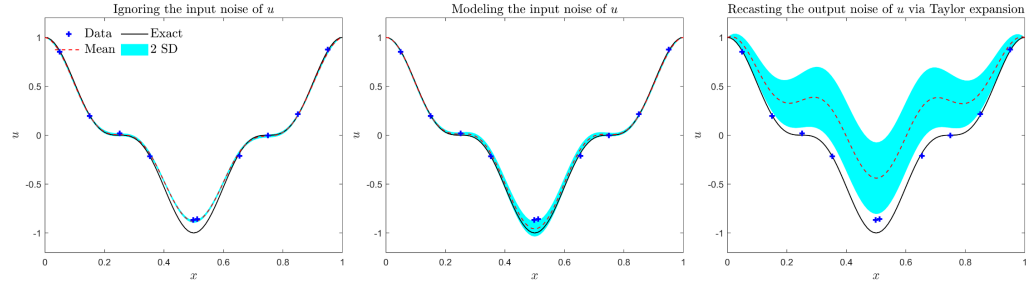
recent progress in integrating UQ with data-driven or physics-informed optimization objectives [285, 110, 322, 215, 224, 314, 324, 316, 14, 96, 128, 143, 317, 319, 17, 66, 86, 90, 173, 198, 41, 205, 297, 315, 286, 112, 115], UQ methods are still scarcely utilized within SciML. One reason for this limited adoption is the relatively nascent state of UQ within the broader deep learning community, where it is still developing and has yet to gain widespread trust and understanding. Additionally, the incorporation of physical constraints, which are critical for many SciML problems, introduces additional complexity and computational overhead compared to standard deep learning techniques. These factors further contribute to the hesitation within the machine learning community to embrace UQ in SciML applications.

A significant body of existing work on UQ in SciML has primarily focused on uncertain, imprecise, or incomplete data, which refers to uncertainty over the exact value of the sought solution, the model parameter, the source term, and/or the boundary term of the differential equation [234, 281, 315, 318, 174, 321, 320, 317, 335, 92, 112, 206, 173, 329, 175, 90, 325, 136]. For example, when solving a PDE, the initial condition can only be resolved by noisy or partial measurements at some discrete points rather than being known precisely across the domain. Figure 1.2(a) illustrates an example of approximating a function with UQ from noisy and gappy observations [315].

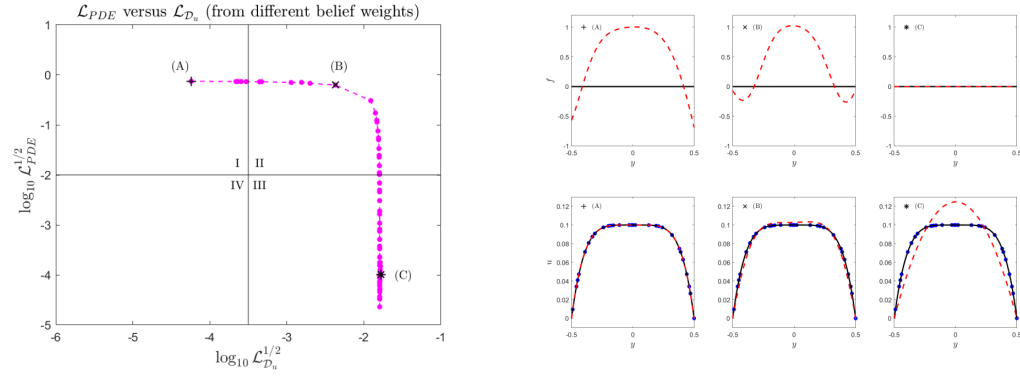
This dissertation delves deeper into UQ in SciML and addresses challenges including data sparsity, high noise levels, high dimensionality, interpretability, large-scale data, incremental and continual learning, hyperparameter tuning, physical model misspecifications, noise model misspecifications, and the implementation of various inference algorithms. Consequently, all uncertainties depicted in Figure 1.1 are either modeled, addressed, or discussed in this dissertation, providing a comprehensive and systematic investigation of UQ in SciML is presented.



(a) Regressing noisy and gappy data [315]. (b) Solving a PDE with sparse and noisy data by leveraging an informative functional priors (adapted from Section 2.4.3 of Chapter 2).



(c) Addressing noisy inputs in solving an inverse PDE problem (adapted from Section 4.4.1 of Chapter 4).



(d) When PINNs encounter model misspecifications (adapted from Section 5.3.3.1 of Chapter 5).

Figure 1.2: (a) An example of approximating a function from its noisy and gappy measurements (adapted from [315]). (b) Utilizing an informative functional prior in solving a time-dependent PDE with sparse and noisy measurements of the initial condition (adapted from Section 2.4.3 of Chapter 2). (c) An example of ignoring and modeling the input noise when solving an inverse PDE problem with noisy inputs-outputs data (adapted from Section 4.4.1 of Chapter 4). (d) An example of physical model misspecifications in PINNs, in which a Newtonian flow is assumed while the data comes from a non-Newtonian flow (Section 5.3.3.1 of Chapter 5).

The remainder of this dissertation is organized as follows:

1. In Chapter 2, we address challenges related to data sparsity and high noise levels in solving ODE/PDE problems with UQ by effectively leveraging functional priors, learned from prior information and historical data. Specifically, we incorporate a multi-head architecture within PINNs and present a method, termed multi-head physics-informed neural networks (MH-PINNs), for learning functional priors, which are then utilized either as regularization terms in deterministic setting or as prior distributions in Bayesian inference. Figure 1.2(b) provides an example of solving a time-dependent PDE with sparse and noise measurements of the initial condition using a learned functional prior.
2. In Chapter 3, we establish a novel theoretical connection between Bayesian inference and Hamilton-Jacobi (HJ) PDEs for UQ in SciML, offering a new interpretation of UQ problems and developing an efficient Riccati-based methodology. This connection reveals that the posterior mean and covariance can be recovered from the spatial gradient and Hessian of the solution to a viscous HJ PDE. The proposed methodology provides computational advantages in continuously updating the model predictions with UQ. In this chapter, we address challenges of interpretability, large-scale data, incremental/continual learning, and hyperparameter tuning.
3. Chapter 4 focuses on noise model misspecifications, particularly ignoring noise in input measurements. We develop a Bayesian approach for quantifying uncertainty arising from noisy inputs and outputs in PINNs and neural operators (NOs). Noisy inputs include spatial-temporal coordinate measurements in PINNs and input function measurements in NOs. The proposed method enables PINNs to handle noisy input-output scenarios (see Figure 1.2(c)) and

allows pretrained NOs to process noisy measurements of both input and output functions with UQ.

4. In Chapter 5, we investigate model misspecifications and uncertainties, presenting a framework to correct misspecified models and quantify uncertainty when solving ODEs/PDEs. PINNs encode assumed physical models, which may be imperfect, and other deep neural networks (DNNs) capture discrepancies between these models and observational data. Due to the expressivity of DNNs, this method reduces computational errors arising from model misspecifications, enabling applications of PINNs to complex systems with partially known physics. Additionally, Bayesian PINNs (B-PINNs) and ensemble PINNs quantify uncertainties from noisy or gappy data in the discovered governing equations. An example of when PINNs encounter model misspecification, where a Newtonian flow is assumed, but the underlying physics that generates the data is non-Newtonian, is illustrated in Figure 1.2(d).
5. In Chapter 6, we address the challenges of implementations of various inference algorithms for UQ in SciML and present a comprehensive open-source library, termed NeuralUQ, for UQ in neural differential equations and neural operators. We provide an educational tutorial, for employing UQ methods for SciML in a convenient and structured manner. The library, designed for both educational and research purposes, supports multiple modern UQ methods and SciML models. It is based on a succinct workflow and facilitates flexible employment and easy extensions by the users. We first present a tutorial of NeuralUQ and subsequently demonstrate its applicability and efficiency in four diverse examples, involving dynamical systems and high-dimensional parametric and time-dependent PDEs.
6. We summarize the contributions of this dissertation, discuss its limitations,

and outline directions for future work in Chapter 7.

This dissertation provides a structured and in-depth examination of UQ in SciML, addressing theoretical, computational, and practical challenges in this emerging field.

CHAPTER TWO

Multi-head physics-informed neural networks for learning functional priors and uncertainty quantification

2.1 Introduction

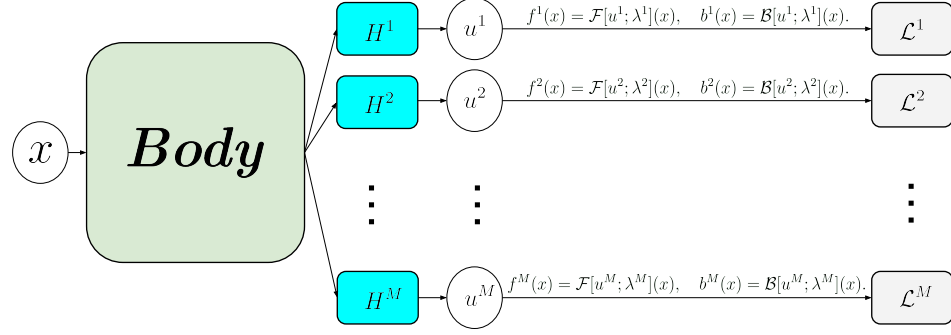
Solving ordinary/partial differential equations (ODEs/PDEs) with incomplete and uncertain information of the equation can pose significant challenges for both classical numerical methods and scientific machine learning (SciML) approaches due to limited knowledge of the problem setup. For instance, consider solving a time-dependent PDE where the initial condition is only partially specified, available only through sparse or noisy measurements. In such cases, classical numerical methods cannot be directly applied because the problem lacks a fully defined specification. Although recently developed SciML techniques, such as physics-informed neural networks (PINNs) [244], can be applied in theory, they often yield low accuracy and unreliable results due to the inherent ill-posedness of these problems [142, 60, 281, 234, 201]. In these situations, effectively leveraging the prior knowledge and historical information as well as quantifying uncertainty due to limited information is essential for achieving accurate and reliable solution inference. A recent work [201] has proposed to use physics-informed generative adversarial networks (PI-GANs) [105, 317] to learn functional priors and posteriors for addressing such challenges.

In machine learning, hard-parameter sharing architectures, especially the multi-head structures, have demonstrated significant success in learning informative representations, which can provide useful information for new but similar tasks. A *multi-head neural networks* (MH-NN) typically consists of: (1) a shared *body* NN, and (2) multiple distinct *heads* NNs, each associated with a specific output and machine learning task (see Figure 2.1 for a schematic). This structure enforces shared body NN across all outputs, which can lead to a meaningful shared representation in the body through successful training. Originally introduced for multi-task learning

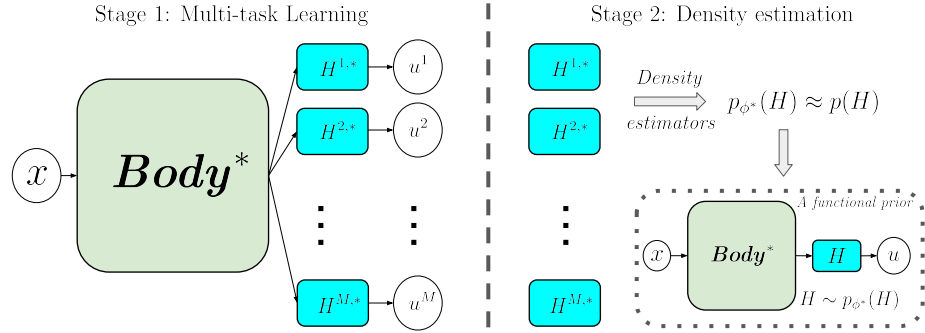
(MTL) in [40, 39], this architecture aimed to achieve inductive transfer among tasks to reduce generalization error. The goals of using MH-NNs in MTL are multi-fold: to enhance performance across all tasks, to learn robust and transferable representations, and to leverage auxiliary tasks to boost the learning of primary tasks. Recently, MH-NNs have also proven effective in meta-learning [294, 239]. In the SciML community, similar NN architectures have been applied to solving differential equations. For example, in [228], multiple ODEs/PDEs were solved concurrently in a MTL framework using the multi-head architecture. Other works [16, 71, 227] considered the shared body as an informative representation, enabling fast and accurate transfer learning. This architecture has also been integrated into PINNs for solving single ODE/PDE problems with uncertainty quantification (UQ) [318], known as multi-output PINNs. Overall, MH-NNs have shown strong capability in capturing useful, shared representations, making them an attractive choice for complex scientific learning tasks.

While the multi-head architecture has proven effective for solving ODE/PDE problems, most prior work has focused solely on leveraging the shared body as an informative representation for downstream tasks, overlooking valuable information contained within the individual heads. Although the body encapsulates certain prior knowledge that can be useful for similar ODE/PDE problems, it may be insufficient for achieving high accuracy, especially when data are highly sparse or corrupted by significant noise. Furthermore, UQ becomes more challenging, as defining a prior distribution is not as straightforward as it is with conventional neural networks [234, 315, 348, 201].

In this chapter, we develop multi-head physics-informed neural networks (MH-PINNs) and equip them with density estimators, specifically normalizing flows (NFs) [220, 157], to learn from prior information and historical data, which can be lever-



(a) The architecture of MH-PINNs.



(b) Equipping MH-PINNs with density estimators for learning functional prior and solving downstream tasks with UQ.

Figure 2.1: A schematic view of the proposed approach for learning functional priors. In (a), the architecture of multi-head physics-informed neural networks (MH-PINNs) with M heads is displayed. All M functions share the same body NN, denoted as **Body**, and each is associated with a head NN, parameterized by H^k , $k = 1, \dots, M$, and a specific PINN task $\mathcal{T}^k$ with the loss function $\mathcal{L}^k$ and data $\mathcal{D}^k$. These functions $u^k, k = 1, \dots, M$ are solutions to M ODEs/PDEs, which may differ in the source term f^k , boundary term b^k , and/or model parameter λ^k (see (2.1) for the problem formulation). In (b), we present the breakdown of our approach: (1) training a MH-PINN to obtain a shared representation (body NN), denoted as **Body***, and multiple heads, denoted as $\{H^{k,*}\}_{k=1}^M$, and (2) estimating the distribution of the head, denoted as $p(H)$, from these heads via density estimators such as normalizing flows NF. The estimated density is denoted as $p_{\phi^*}(H)$. Then the functional prior is represented by the learned body NN and the estimated density. (A NN parameter with a superscript $*$ indicates that it has been trained.)

aged to solve ODE/PDE problems accurately with UQ under sparse and noisy data conditions. We refer to such prior knowledge of function as *functional prior*. Our approach to learning a functional prior consists of two main steps: (1) training a MH-PINN to obtain a shared representation, or body NN, along with multiple head

NNs, and (2) estimating a distribution from these multiple heads using an NF. In this chapter, fully-connected NNs are selected as the backbone architecture, where all nonlinear hidden layers form the body and the final linear layer functions as each head. This setup enables the density estimators to operate on low-dimensional variables, making estimation more efficient. The learned functional prior comprises the trained body and the estimated density function. When solving a new but similar ODE/PDE problem, the trained body is reused as an informative representation, while a new head is optimized in a deterministic setting or inferred with UQ in a Bayesian framework. The density function serves as either a regularization term or a prior distribution for the new head. We will showcase this straightforward yet effective approach with a series of examples in Section 2.4 and demonstrate that this density function is essential for achieving accurate and reliable inferences, particularly when data are sparse or noisy.

The main contributions of this work are as follows:

1. We introduce MH-PINNs equipped with density estimators to learn functional priors, providing a richer representation of prior information than the learned body alone in MH-PINNs.
2. We tackle downstream problems with sparse and noisy data by leveraging these learned functional priors for improved accuracy, robustness and uncertainty quantification.

The remainder of the chapter is organized as follows. In Section 2.2, we formulate the problem and present our two-step approach for learning functional priors using multi-head architectures and normalizing flows (NFs). Section 2.3 details how the learned functional prior can be applied to solve downstream ODE/PDE prob-

lems, incorporating prior knowledge and historical information. In Section 2.4, we present computational experiments, including a high-dimensional parametric PDE, all addressing scenarios with sparse or noisy data. We summarize in Section 2.5.

2.2 Learning functional priors via multi-head architectures and normalizing flows

In this section, we first describe how to integrate multi-head architectures into PINNs to learn representations from solving historical tasks in Section 2.2.1. Then, in Section 2.2.2, we will present how MH-PINNs can be turned into functional priors, encapsulating prior information in a compact form using density estimators, specifically normalizing flows.

2.2.1 Multi-head physics-informed neural networks (MH-PINNs)

We assume that we have a family of tasks, $\{\mathcal{T}^k\}_{k=1}^M$ where M denotes the number of tasks and k denotes the index of task $\mathcal{T}^k$. Each task $\mathcal{T}^k$ is a physics-informed regression problem as in PINNs [244, 180, 32, 34, 346, 169, 268, 334], in which we wish to solve the following nonlinear ODE/PDE with PINNs given data $\mathcal{D}^k$:

$$\mathcal{F}[u^k; \lambda^k](x) = f^k(x), x \in \Omega, \quad (2.1a)$$

$$\mathcal{B}[u^k; \lambda^k](x) = b^k(x), x \in \partial\Omega, \quad (2.1b)$$

where u^k denotes the solution for task $\mathcal{T}^k$, λ^k the model parameter, f^k the source term, and b^k the boundary term. Here we assume the domain Ω , the differential operator $\mathcal{F}$ and the boundary operator $\mathcal{B}$ are the same across all tasks. The data for $\mathcal{T}^k$ can be expressed as $\mathcal{D}^k := \left\{ \{x_{i,f}^k, f_i^k\}_{i=1}^{N_f^k}, \{x_{i,b}^k, b_i^k\}_{i=1}^{N_b^k}, \{x_{i,u}^k, u_i^k\}_{i=1}^{N_u^k} \right\}$ where N_f^k, N_b^k, N_u^k denote the number of data for the source term, the boundary term and the solution, respectively. We note that $N_u^k = 0$ when λ^k is known and the problem becomes solving (2.1). The task $\mathcal{T}^k$ is described as approximating u_k , f^k , and/or b^k from data $\mathcal{D}^k$ and physics, which is achieved by optimizing the following PINN loss function with gradient-descent methods [255]:

$$\mathcal{L}^k(\theta^k; \mathcal{D}^k) = w_u^k \mathcal{L}_u^k(\theta^k) + \mathcal{L}_{PDE}^k(\theta^k), \quad (2.2)$$

where

$$\mathcal{L}_u^k(\theta^k) = \frac{1}{N_u^k} \sum_{i=1}^{N_u^k} \|u_{\theta^k}(x_{i,u}^k) - u_i^k\|^2, \quad (2.3a)$$

$$\mathcal{L}_{PDE}^k(\theta^k) = \frac{w_f^k}{N_f^k} \sum_{i=1}^{N_f^k} \|\mathcal{F}[u_{\theta^k}; \lambda^k](x_{i,f}^k) - f_i^k\|^2 + \frac{w_b^k}{N_b^k} \sum_{i=1}^{N_b^k} \|\mathcal{B}[u_{\theta^k}; \lambda^k](x_{i,b}^k) - b_i^k\|^2, \quad (2.3b)$$

where θ^k denotes the NN parameter for task $\mathcal{T}^k$, w_u^k, w_f^k, w_b^k are the weighting coefficients, and $\|\cdot\|$ is the l^2 -norm. These tasks and associated data, i.e. $\{\mathcal{T}^k, \mathcal{D}^k\}_{k=1}^M$ are considered as historical information and will be used for learning functional priors.

Many numerical methods including recently developed SciML approaches often tackle $\{\mathcal{T}^k\}_{k=1}^M$ independently, without leveraging knowledge across tasks. The PINNs method [244] was originally designed to solve these tasks independently using NNs and also yields M uncorrelated results. In this chapter, we employ the multi-head architecture to construct MH-PINNs and instead treat $\{\mathcal{T}^k\}_{k=1}^M$ as a whole. We

denote the parameter of the body by ψ and the parameter of the head associated with the k th PINN for $\mathcal{T}^k$ by H^k , $k = 1, \dots, M$. All PINNs are therefore enforced to share the same body and possess their own heads, i.e. $\theta^k = \{\psi, H^k\}$. By this notation, we rewrite the output of the k th PINN as

$$u_{\theta^k}(x) = u_\psi(x; H^k), x \in \Omega. \quad (2.4)$$

We solve all M tasks by optimizing the following loss function $\mathcal{L}_{MTL}$ of ψ and $\{H^k\}_{k=1}^M$:

$$\mathcal{L}_{MTL}(\psi, \{H^k\}_{k=1}^M; \{\mathcal{D}^k\}_{k=1}^M) = \frac{1}{M} \sum_{k=1}^M \mathcal{L}^k(\{\psi, H^k\}; \mathcal{D}^k). \quad (2.5)$$

We denote the optimized parameters by ψ^* and $\{H^{k,*}\}_{k=1}^M$ for the shared body and M heads, respectively.

The existing work involving multi-head structures for solving ODEs/PDEs typically utilizes the learned body NN, which is a representation learned from solving $\{\mathcal{T}^k\}_{k=1}^M$, for other new downstream tasks, such that the prior knowledge and historical information can be leveraged to address similar problems [16, 71, 227]. For example, in [71], transfer learning was conducted by reusing the learned body NN and optimizing the individual PINN loss function with respect to the parameter of head. Specifically, for a new task with data $\mathcal{D} := \left\{ \{x_{i,f}, f_i\}_{i=1}^{N_f}, \{x_{i,b}, b_i\}_{i=1}^{N_b}, \{x_{i,u}, u_i\}_{i=1}^{N_u} \right\}$ and model parameter λ , the following loss function $\mathcal{L}$ is optimized:

$$\begin{aligned} \mathcal{L}(H; \mathcal{D}) = & \frac{w_u}{N_u} \sum_{i=1}^{N_u} \|u_\psi(x_{i,u}; H) - u_i\|^2 + \frac{w_f}{N_f} \sum_{i=1}^{N_f} \|\mathcal{F}[u_\psi; \lambda](x_{i,f}; H) - f_i\|^2 \\ & + \frac{w_b}{N_b} \sum_{i=1}^{N_b} \|\mathcal{B}[u_\psi; \lambda](x_{i,b}; H) - b_i\|^2. \end{aligned} \quad (2.6)$$

We denote the optimized head parameter for the new task by H^* , and the solution to the ODE/PDE problem of the new task can then be expressed as $u_\psi(x, H^*), x \in$

Ω . This treatment of the learned body NN has proven effective in the literature for deterministic inference when data are sufficient. However, we point out and empirically show in Section 2.4 that the loss function defined in (2.6) is not enough to handle data sparsity or high-level noise of $\mathcal{D}$, and the information contained in the head NN has to be leveraged as well for accurate inferences and uncertainty quantification (UQ). The details will be discussed in Section 2.3.

2.2.2 Estimating the distribution of the head via normalizing flows (NFs)

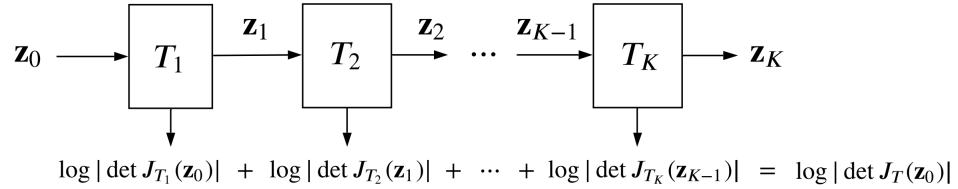


Figure 2.2: An illustration of a normalizing flow consisting of K invertible transforms, denoted by $T_i, i = 1, \dots, K$ (adapted from [220]). Here $\mathbf{z}_0$ represents the base random variable whose distribution is tractable and known, and $\mathbf{z}_K$ represents the target random variable (H) whose distribution is to be estimated.

We note that by utilizing the learned body NN with the optimized parameter ψ^* and probabilistic modeling of the parameter of the head NN (H), i.e. treating H as a random variable, we obtain a functional prior which represents the prior in the functional space [201] and can be expressed as:

$$u_{\psi^*}(x; H), x \in \Omega, H \sim p(H), \quad (2.7)$$

where $p(H)$ denotes the probability density function (PDF) of H . In this regard, M heads obtained from the training of MH-PINNs, i.e. $\{H^{k,*}\}_{k=1}^M$, are considered as samples from the distribution $p(H)$ and can be used for estimating

$p(H)$. In this chapter, we propose to learn the distribution of H using normalizing flows (NFs), which are a class of invertible NNs that transform a probability density through a sequence of invertible mappings and have been extensively used for density estimation, generative modeling, and variational inference in both conventional [220, 157, 74, 221, 153] and scientific machine learning communities [315, 197, 112, 327, 300]. An example of a specific NF is presented in Fig. 2.2. We denote the PDF of the NF by $\hat{p}_\phi(H)$, where ϕ denotes the parameter of NF.

We choose NFs over other commonly used generative models, e.g., generative adversarial networks (GANs) [105], variational auto-encoders (VAEs) [154], or diffusion models [123], because the NF serves as both a density estimator and a generative model. Unlike these generative models, NFs explicitly learn the distribution $p(H)$ from samples of H by minimizing the negative log likelihood:

$$-\frac{1}{M} \sum_{k=1}^M \log \hat{p}_\phi(H^{k,*}). \quad (2.8)$$

The exact density $\hat{p}_\phi(H^{k,*})$ can be computed efficiently for any sample because, by construction, NFs are invertible and the Jacobian determinant of the invertible mapping is relatively easy to compute [220, 157]. We denote the optimized parameter of NF by ϕ^* , and in the end, the functional prior represented by the body NN and the learned distribution of the head NN is expressed as:

$$u_{\psi^*}(x; H), x \in \Omega, H \sim \hat{p}_{\phi^*}(H). \quad (2.9)$$

It is worth noting that in previous works on physics-informed generative methods [317, 335, 112], NNs are trained by structured data of u^k , f^k , and/or b^k . Our method, on the other hand, learns through samples of the head, which are obtained from MH-

PINNs in the first step. This learning strategy brings two substantial advantages: (1) flexibility in dealing with unstructured data, e.g., unstructured data; (2) simplicity and controllability of the training by decoupling the physics-informed learning and the generative modeling.

2.3 Leveraging functional priors in solving downstream tasks

In this section, we describe the details of how to leverage the functional prior learned from historical data using the two-step method presented in the previous section to solve downstream ODE/PDE problems when data, expressed as $\mathcal{D} = \left\{ \{x_{i,f}, f_i\}_{i=1}^{N_f}, \{x_{i,b}, b_i\}_{i=1}^{N_b}, \{x_{i,u}, u_i\}_{i=1}^{N_u} \right\}$, are sparse and/or noisy. We note that in the new downstream task, the model parameter λ can be known or unknown. Recall that in Section 2.2.1, we mentioned that existing work on utilizing multi-head structures for solving ODEs/PDEs reuses the body NN and finds a new head such that the PINN loss function (2.6) for the downstream task is minimized. Although this strategy is efficient and accurate when data for the downstream task are sufficient, it often fails when data are sparse and/or noisy and it does not provide predicted uncertainty, which is crucial for reliable inferences when data are incomplete or uncertain. In this chapter, we propose two methods, which leverage both the learned body NN and the distribution of the head NN:

1. For deterministic setting and when data are sparse but clean or corrupted by low-level noise, we add to the PINN loss function (2.6) a regularization term for H , $-\log \hat{p}_{\phi^*}(H)$, such that the prior knowledge of the head NN is provided

to regularize the behavior of H in the optimization problem.

2. For Bayesian inference and when UQ is needed, we estimate the posterior distribution in which the likelihood is constructed as it is in regular Bayesian inference for solving ODEs/PDEs and the prior is provided by $\hat{p}_{\phi^*}(H)$.

The details are presented below.

2.3.1 Regularization in optimization problem

Limited data often leads to overfitting and/or poor inter-/extrapolation performance in directly minimizing the PINN loss function (2.6) for new downstream tasks. In this regard, regularizing the head according to its distribution is capable of preventing overfitting and providing additional prior knowledge for better inter-/extrapolation. The regularized PINN loss function (denoted as $\tilde{\mathcal{L}}$) is expressed as follows:

$$\tilde{\mathcal{L}}(H; \mathcal{D}) = \mathcal{L}(H; \mathcal{D}) - \alpha \log p(H) \approx \mathcal{L}(H; \mathcal{D}) - \alpha \log p_{\phi^*}(H), \quad (2.10)$$

where $\mathcal{L}(H; \mathcal{D})$ is the regular PINN loss function (2.6) for data $\tilde{\mathcal{D}}$ and parameter H and $\alpha \geq 0$ is the coefficient to adjust the regularization effect. The approximation in (2.10) is due to the approximation error in estimating the distribution of H through NFs.

2.3.2 Prior distribution in Bayesian inference

When data are noisy and UQ is of concern, as opposed to point estimate obtained by minimizing the regularized loss function (2.10), one may turn to Bayesian inference

and estimate the posterior distribution. Similar as in [315, 175, 201, 348, 345, 347], by Bayes' theorem, the posterior distribution of H (denoted as $p(H|\mathcal{D})$) is established as follows:

$$p(H|\mathcal{D}) \propto p(\mathcal{D}|H)p(H) \approx p(\mathcal{D}|H)\hat{p}_{\phi^*}(H). \quad (2.11)$$

The likelihood $p(\mathcal{D}|H)$ can be expressed as follows when data are i.i.d. and the noise is additive Gaussian with mean zero and standard deviation $\sigma_u, \sigma_f, \sigma_b$ for measuring data of u , f , and b , respectively:

$$p(\mathcal{D}|H) = \prod_{i=1}^{N_u} p(u_i|x_i^u, H, \psi^*) \prod_{i=1}^{N_f} p(f_i|x_i^f, H, \psi^*) \prod_{i=1}^{N_b} p(b_i|x_i^b, H, \psi^*), \quad (2.12)$$

where

$$p(u_i|x_i^u, H, \psi^*) = \frac{1}{\sqrt{2\pi}\sigma_u} \exp\left(-\frac{\|u_{\psi^*}(x_i^f; H) - u_i\|^2}{2\sigma_u^2}\right), \quad (2.13a)$$

$$p(f_i|x_i^f, H, \psi^*) = \frac{1}{\sqrt{2\pi}\sigma_f} \exp\left(-\frac{\|\mathcal{F}[u_{\psi^*}, \lambda](x_i^f; H) - f_i\|^2}{2\sigma_f^2}\right), \quad (2.13b)$$

$$p(b_i|x_i^b, H, \psi^*) = \frac{1}{\sqrt{2\pi}\sigma_b} \exp\left(-\frac{\|\mathcal{B}[u_{\psi^*}, \lambda](x_i^b; H) - b_i\|^2}{2\sigma_b^2}\right). \quad (2.13c)$$

The prior $p(H)$ is approximated by the NF $\hat{p}_{\psi^*}(H)$. Intractability of the posterior distribution (2.11) requires approximation methods, among which Markov Chain Monte Carlo methods, such as Hamiltonian Monte Carlo (HMC) [211], generally provide the most accurate estimation. Moreover, the relatively low dimension of H also enables the use of Laplace's approximation (LA) [145] efficiently, which is employed in this chapter as an alternative to HMC.

2.4 Results

In this section, we test our method for five diverse benchmarks. The first one is a pedagogical function regression, in which we aim to demonstrate the capability of the proposed method in learning functional priors and the importance of leveraging learned functions prior to addressing sparse and noisy data in downstream tasks with UQ. In the second example, we consider a nonlinear ODE system, and we first test our method in approximating stochastic processes with the presence of a differential operator in which we compare different NFs and also compare with physics-informed GANs (PI-GANs) [317, 201] in learning functional priors. We then show that the learned functional prior can also be used for solving inverse problems. In the third example, we solve a 1D time-dependent reaction-diffusion equation with sparse data of the initial condition by leveraging prior knowledge. The fourth and fifth examples are a 2D steady-state nonlinear Allen-Cahn equation and a 2D 120-dimensional parametric Helmholtz equation, respectively. In both cases, we address data sparsity and high-level noise (100% relative noise scale).

In all examples unless stated otherwise, historical data $\{\mathcal{D}^k\}_{k=1}^M$ for learning functional priors are assumed to be noise-free and sufficient, while $\mathcal{D}$ in downstream tasks are assumed to be sparse and/or noisy, making downstream tasks of the few-shot type. Except for the first example, results from Bayesian inference are obtained by employing HMC. The predicted mean is denoted as μ and the predicted standard deviation is denoted as σ and are computed from the posterior samples of functions or unknown parameters. The predicted uncertainty is defined as 2σ in this chapter. Additional details of the numerical experiments, such as NN architectures and training strategies, can be found in Appendices A.1 and A.2. A more detailed investigation of the presented method in MTL and synergistic learning for the function

approximation example is provided in Appendix A.3.

2.4.1 Function approximation

We start with a function regression problem using only data and no physics, which is a degenerate instance of (2.1) with $\mathcal{F}$ being fixed as an identity operator, no $\mathcal{B}$ and b^k , and $u^k = f^k$ being task-specific. In this case, $\mathcal{D}^k$ and $\mathcal{D}$ are given as $\{(x_i^k, f_i^k)\}_{i=1}^{N_k}$ and $\{(x_i, f_i)\}_{i=1}^N$, respectively, and historical tasks $\{\mathcal{T}^k\}_{k=1}^M$ and the downstream task $\tilde{\mathcal{T}}$ is defined as approximating functions f^k and f from $\mathcal{D}^k$ and $\mathcal{D}$, respectively. The stochastic function f in this example is defined as follows:

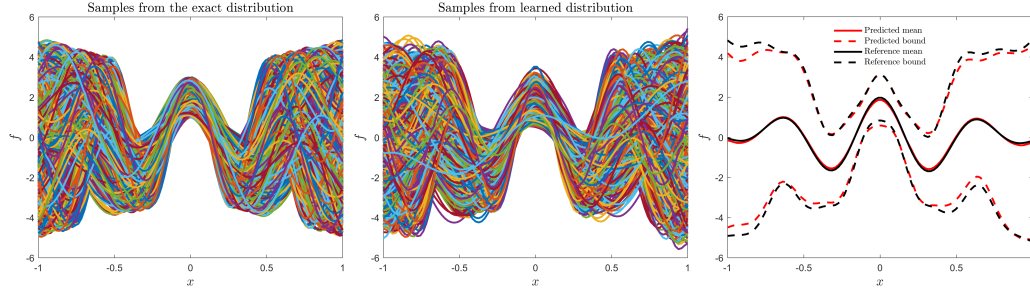
$$\begin{aligned} f(x) &= A \cos(\omega x) + 2\beta x, x \in [-1, 1], \\ A &\sim U[1, 3], \omega \sim U[2\pi, 4\pi), \mathcal{P}(\beta = \pm 1) = 0.5, \end{aligned} \tag{2.14}$$

where U stands for uniform distribution and $\mathcal{P}(\Xi)$ denotes the probability of the event Ξ . Our goals are: (1) learn a functional prior of f from data with MH-NNs and NFs, and (2) leverage the learned functional prior in solving downstream few-shot regression tasks. Specifically, two downstream tasks are tackled:

1. Regressing $2 \cos(2\pi x) + 2x$ from four measurements.
2. Regressing $2 \cos(4\pi x) - 2x$ from five measurements.

In both downstream tasks, measurements are equidistantly distributed on $[-0.9, -0.1]$ and the domain for the regression problem is $[-1, 1]$.

For the training of MH-NNs and NFs, $M = 1,000$ f subject to (2.14) are sampled (and displayed in Fig. 2.3(a)), each of which forms a regression task with 40 mea-



(a) Learning functional prior from historical data.

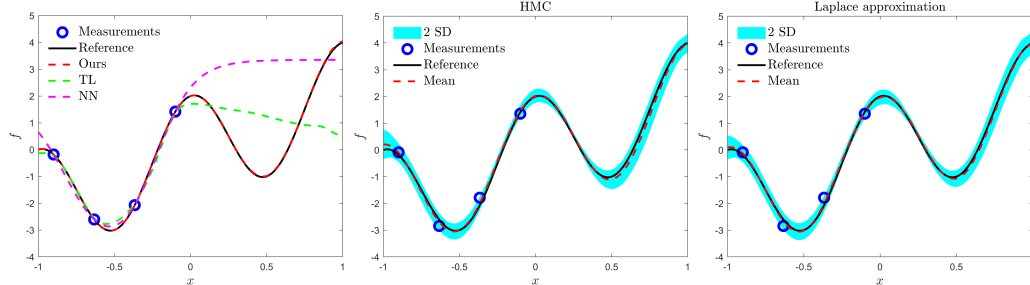
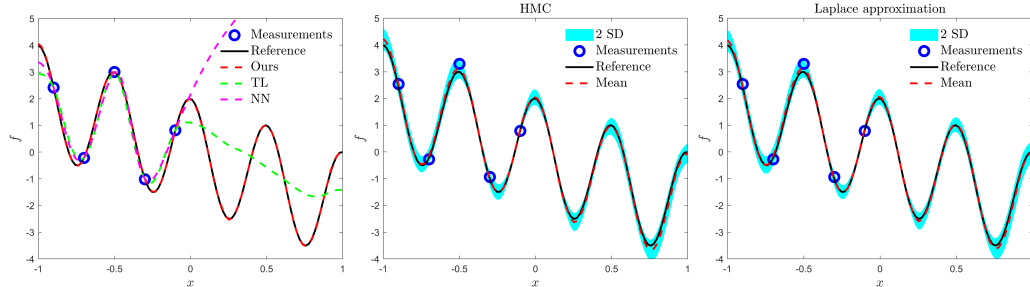
(b) Regressing $2 \cos(2\pi x) + 2x$ from only four measurements.(c) Regressing $2 \cos(4\pi x) - 2x$ from only five measurements.

Figure 2.3: Results for approximating the stochastic function defined in (2.14) (a functional prior) and solving the downstream few-shot regression tasks. (a) **Left**: 1,000 samples generated from the exact distribution; **middle**: 1,000 samples generated from the learned generator; **right**: statistics computed from samples in which we refer to the interval of mean ± 2 standard deviations as bound. (b)/(c) Results for the downstream tasks. **Left**: results for clean-data cases using our method, the transfer learning (TL) method in [71], and regular NN method; **middle**: results for noisy-data cases using our method with HMC for posterior estimation and UQ; **right**: results for the same noisy-data cases using our method with LA for posterior estimation and UQ.

measurements sampled equidistantly on $[-1, 1]$ as data. Both clean and noisy data cases are considered in these two downstream tasks. In clean-data cases, the regression problem is solved by fine-tuning the head using gradient descent, while in noisy-data

cases, Bayesian inference is adopted and the posterior distribution (2.11) is estimated using HMC and LA for UQ. The noise is assumed to be additive Gaussian noise with mean zero and standard deviation 0.2. In addition, for clean-data cases, we compare with (1) fitting data with a NN in which no prior knowledge is leveraged and all NN parameters are trained, and (2) a transfer learning method proposed in [71], in which the pretrained body NN is reused but without regularization provided by the estimated density of the head.

Results for approximating the stochastic function f and solving the downstream tasks are presented in Figure 2.3. As shown in Figure 2.3(a), our method approximates the stochastic function f well, demonstrating the capability of MH-NNs in learning functional priors. In solving the downstream tasks with clean data, L_2 regularization is imposed in the transfer learning method and regular NN method to prevent overfitting, particularly when only four or five measurements are available. Figures 2.3(b) and (c) illustrate that our approach yields accurate predictions and performs significantly better than the other two methods in both tasks, especially in regions without measurements. By comparing our approach with the NN method, we can see that prior knowledge of f is learned from $\{\mathcal{T}_k\}_{k=1}^M$ and transferred successfully to new downstream tasks. Furthermore, a comparison with the transfer learning method reveals that the prior knowledge is encoded in both the body and (the distribution of) the head.

In noisy-data cases, we observe that for both tasks and posterior estimating methods, the predictions are accurate and trustworthy: the predicted means agree with the references, and the errors are bounded by the predicted uncertainties. Notably, the predicted uncertainties do not increase over the interval $[0, 1]$ and exhibit periodic patterns even in the absence of measurements. This behavior is due to the informative prior learned by the MH-NNs and NFs, which is imposed on the head

in Bayesian inference.

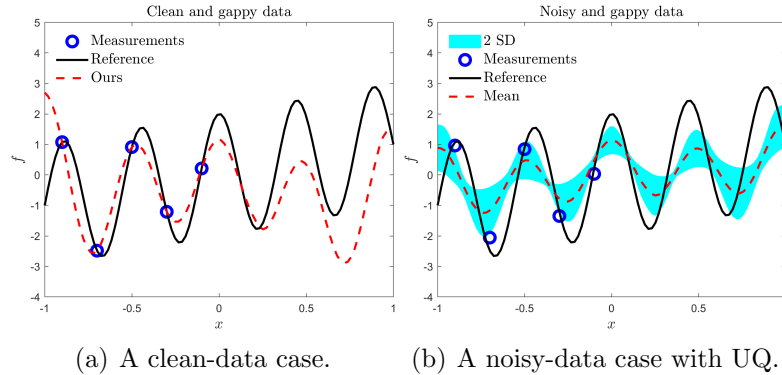


Figure 2.4: Results for regression on an out-of-distribution (OOD) function: (a) a clean-data case in which (2.10) is minimized, and (b) a noisy-data case in which the posterior distribution (2.11) is estimated with HMC for UQ.

The target functions in the previously considered downstream tasks are in-distribution, which allows them to be well-regressed even with limited and noisy data, as they belong to the function space on which the functional prior was trained. However, when downstream task functions are out-of-distribution (OOD), our approach struggles to produce accurate predictions, as shown in Figure 2.4. In this case, the target function is defined as $2 \cos(4.5\pi) + x$ with both ω and β being OOD and data are corrupted by the same noise type when Bayesian inference is performed. The predictions do not match the reference because the target function lies outside the training distribution. However, we still observe predicted periodic patterns from the learned functional prior.

2.4.2 Nonlinear ODE system

In this example, we consider the following ODE, which describes the motion of a pendulum with an external force [178]:

$$u_{tt} = -\lambda \sin(u) + f(t), \quad (2.15)$$

where f is the external force and λ is a model parameter. Here we assume that the initial conditions $u(0) = u_t(0) = 0$ are known.

2.4.2.1 Forward problem

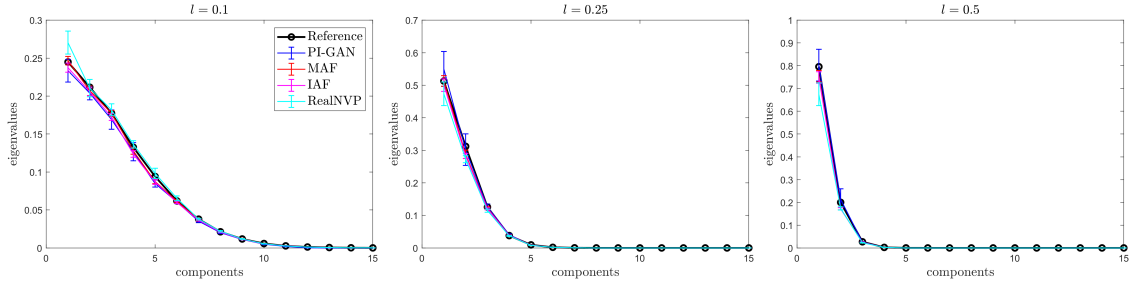


Figure 2.5: Approximating the Gaussian process defined in (2.16) with different correlation lengths (l) as the source term in (2.15) using different NFs: spectra of the correlation structure from the learned functional priors. We also compare to the PI-GAN method [317]. The covariance matrix is constructed using 10,000 generated samples and eigen-values are averaged over ten functional priors trained independently.

We first assume that $\lambda = 1$ in (2.15) is known and we have access to historical data of f , i.e. $\{\mathcal{D}^k\}_{k=1}^M$ where $\mathcal{D}^k = \{(x_i^k, f_i^k)\}_{i=1}^{N_f^k}$, from which we employ our approach to learn a functional prior. We note that by the construction of PINNs and our method, we consequently obtain a functional prior for both f and u . In this case, we assume that f follows a Gaussian process with the following squared exponential

kernel function:

$$f(t) \sim GP(0, K), t \in [0, 1], K(x, x') = \exp\left(-\frac{|x - x'|^2}{2l^2}\right), \quad (2.16)$$

where the correlation length l is set to 0.1, 0.25, 0.5. As discussed in Section 2.3, many types of NFs have been developed recently for generative modeling and density estimation and our approach is compatible with all NFs. Specifically in this example, we compare three popular NFs, RealNVP [74], MAF [221] and IAF [153], and eventually compare with PI-GAN [317] in learning functional priors which approximate the Gaussian process defined in (2.16) with different correlation lengths.

	MAF	IAF	RealNVP	PI-GAN
Stage (1)	134s	134s	134s	N/A
Stage (2)	252s	3939s	245s	N/A
Total	386s	4073s	379s	3243s
Sampling	1.98×10^{-1} s	1.48×10^{-2} s	1.50×10^{-2} s	2.29×10^{-3} s

Table 2.1: Computational cost for different NFs and PI-GANs to approximate the Gaussian process defined in (2.16) with correlation length $l = 0.1$. The proposed method using MH-PINNs and NFs is a two-step method and hence its computation is decomposed into two parts: training MH-PINNs referred to as stage (1) and training NFs referred to as stage (2). Sampling time is evaluated as the average time taken to generate 10,000 samples of u in (2.15).

For the training of MH-PINNs and NFs as well as PI-GANs, we sample $M = 2,000$ realizations of the Gaussian process defined in (2.16) with a uniform mesh of size 65 on $[0, 1]$. The comparison results are presented in Figure 2.5 and Table 2.1. From the spectral analysis of the approximated Gaussian processes shown in Figure 2.5, we observe that MH-PINNs with MAF and IAF perform comparably to PI-GANs, while MH-PINNs with RealNVP perform slightly below. As shown in Table 2.1, the computational costs of MH-PINNs with MAF and RealNVP are significantly lower than PI-GANs, whereas MH-PINNs with IAF are more computationally expensive. Additionally, we record the computational cost for sampling across different learned functional priors. Table 2.1 shows that PI-GANs generate samples signifi-

cantly faster due to their reliance on relatively shallow NNs, as opposed to the deeper architectures required by NFs for expressivity.

Among the three NFs, IAF is the fastest for sampling, while MAF is the slowest; this pattern contrasts with training time and aligns with each NF’s properties: MAF is slower on the forward pass (used for sample generation) but faster on the inverse pass (used for density computation), while IAF exhibits the opposite behavior. Although MAF is slower for sampling, considering its efficient training and good performance in approximating the Gaussian process defined in (2.16), we select MAF as the density estimator of our approach in all other examples in this chapter.

2.4.2.2 Inverse problem

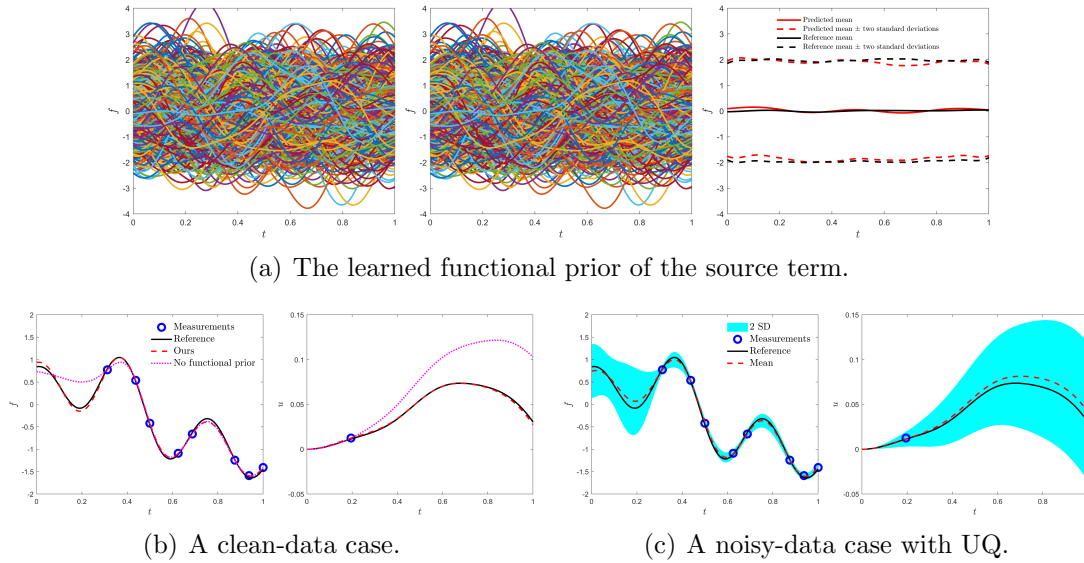


Figure 2.6: Results for the inverse problems of the ODE system (2.15) with initial conditions hard-encoded in the NN modeling. (a) **Left**: 1,000 samples of f , generated from the exact distribution; **middle**: 1,000 samples of f , generated from the learned functional prior; **right**: statistics computed from samples. The bound is defined as the same as in the caption of Figure 2.3. (b) Predicted f and u using PINNs and our approach, for the downstream inverse problem with clean data. (c) Predicted means and uncertainties of f and u using our approach, for the downstream inverse problem with noisy data.

Next, we assume λ in (2.15) is unknown, and some measurements of u and f are available for each task $\mathcal{T}^k$, i.e. $\mathcal{D}^k = \{\{x_i^k, f_i^k\}_{i=1}^{N_f^k}, \{x_i^k, u_i^k\}_{i=1}^{N_u^k}\}$. MH-PINNs are first employed to infer u^k as well as λ^k from data $\mathcal{D}^k$ and the equation, and the NFs are used to learn from samples of H^k and λ^k . To this end, the learned prior is for the joint distribution of u, f and λ . Here, we assume that f follows a truncated Karhuen-Loeve (KL)-expansion with eight leading terms of the Gaussian process defined in (2.16) with $l = 0.1$, and for each task $\mathcal{T}^k$, λ^k is set to $\frac{1}{2} \exp(\int_{[0,1]} (f^k(t))^2 dt)$ in which the integral is estimated via the trapezoidal rule on a uniform mesh of size 65. As for the downstream task $\mathcal{T}$, the target is to infer u and λ from sparse and/or noisy data of u and f .

We sample $M = 2,000$ of f from the KL-expansion of (2.16), and for each sample, the model parameter λ is computed and the solution u is obtained from solving (2.15) using Matlab *ode45*. For each task and the training of MH-PINNs and NFs, we assume that 33 measurements of f_k and nine measurements of u_k , equidistantly distributed on $[0, 1]$, are available, and initial conditions are hard encoded in the NN modeling. For the downstream task $\mathcal{T}$, we assume that one random measurement of u and eight random measurements of f are available with hard-encoded initial conditions as well. For the case with noisy measurements, we assume additive Gaussian noises with mean zero and standard deviation 0.05 and 0.005 for measurements of f and u , respectively. The reference solution is obtained by solving (2.15) with corresponding model parameter and source term.

	vanilla PINN (clean data)	Our approach (clean data)	Our approach (noisy data)
λ	0.8130	2.1351	2.3199 ± 0.2616
Error of u	58.56%	7.11%	12.77%

Table 2.2: Estimate of λ and L_2 relative error of u for the downstream inverse problem on (2.15) using our approach and the vanilla PINN method. The reference value for λ is 2.3609. For the noisy-data case, the error is evaluated on the predicted mean.

Results are shown in Figure 2.6 and Table 2.2, from which we observe that our method is able to learn the functional prior and produce accurate and trustworthy predictions for u , f and also λ in the downstream task with limited data, in both clean- and noisy-data cases. As shown, the PINN method yields unacceptable estimate over both u and λ due to lack of data and not leveraging historical information, while our approach is of much higher accuracy by integrating prior knowledge from $\{\mathcal{T}^k\}_{k=1}^M$ with the proposed approach.

2.4.3 Reaction-diffusion equation

We now test our method on the following one-dimensional nonlinear time-dependent reaction-diffusion equation with $u(1 - u)$ as the reaction model [3]:

$$u_t = Du_{xx} + ku(1 - u), t \in (0, 1], x \in (-1, 1), \quad (2.17a)$$

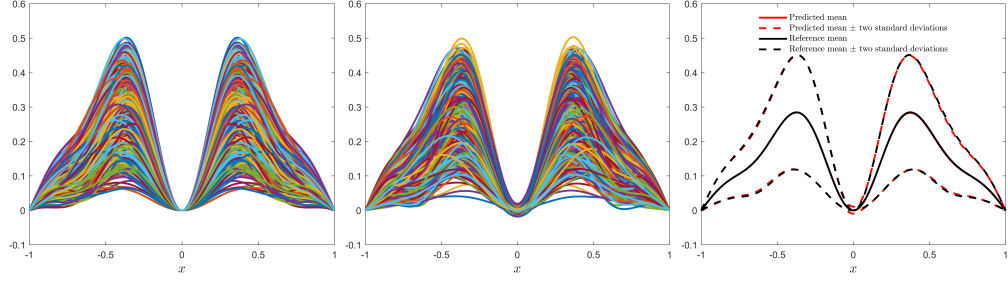
$$u(t, -1) = u(t, 1) = 0, t \in (0, 1], \quad (2.17b)$$

$$u(0, x) = u_0(x), x \in (-1, 1), \quad (2.17c)$$

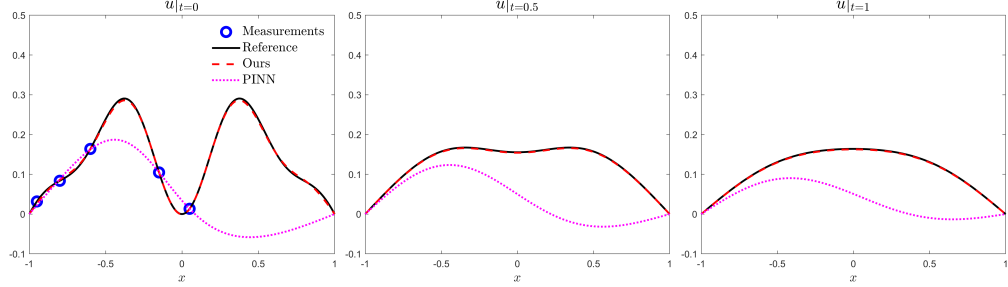
where $D = 0.1, k = 0.1$ are the model parameters and $u_0(x)$ is the initial condition function. In this example, we assume that the initial condition function comes from the following distribution:

$$u_0(x) = \frac{(x^2 - 1)}{K} \sum_{j=1}^K \xi_j (\cos^2(jx) - 1), x \in [-1, 1], \xi_j \sim U[0, 1], j = 1, \dots, K. \quad (2.18)$$

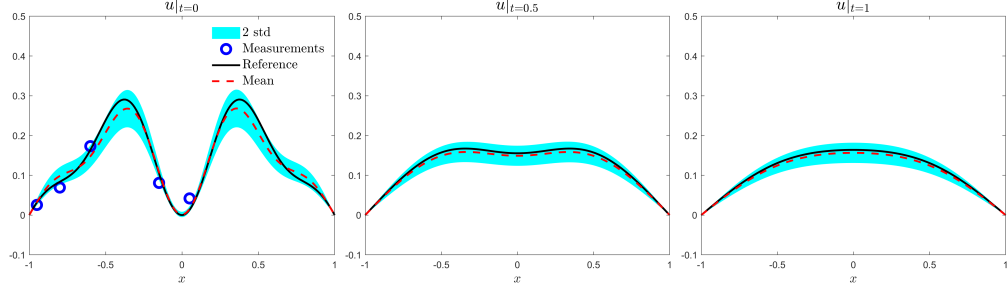
Unlike the previous example, here the stochasticity comes from the initial condition rather than the source term. Here, we set $K = 5$. This example corresponds to (2.1) with $\mathcal{F}^k$ and f^k being the same for all tasks, and u^k and b^k being task-specific. In addition to measurements of u_0 , points on which the PDE residuals are computed



(a) The learned functional prior of the initial condition.



(b) A clean-data case.



(c) A noisy-data case.

Figure 2.7: Solving (2.17) with sparse data of the initial condition u_0 by leveraging learned prior using the proposed method. The boundary conditions are hard-encoded. (a) **Left**: 1,000 training samples of u_0 ; **middle**: 1,000 samples of $u(0, \cdot)$ from the learned generator; **right**: statistics computed from samples. The bound is defined as the same as in the caption of Figure 2.3. (b) Predicted u at $t = 0, 0.5, 1$ using our approach and the PINN method with clean measurements. (c) Predicted means and uncertainties of u at $t = 0, 0.5, 1$ using our approach with HMC for posterior estimate, with noisy measurements.

are also required in both $\{\mathcal{T}^k\}_{k=1}^M$ and downstream tasks $\mathcal{T}$. Hence, data for tasks $\mathcal{T}^k, k = 1, \dots, M$ in this example are $\mathcal{D}^k = \{\{(t_{i,f}^k, x_{i,f}^k), 0\}_{i=1}^{N_f^k}, \{(0, x_{i,b}^k), b_i^k\}_{i=1}^{N_b^k}\}$.

For the training of the functional priors, we sample $M = 2,000$ functions $u_0(x)$ from (2.18) (displayed in Figure 2.7(a)). In each task $\mathcal{T}^k$, we form a physics-informed

	vanilla PINN (clean data)	Our approach (clean data)	Our approach (noisy data)
Error	81.87%	0.92%	5.53%

Table 2.3: L_2 relative error of u for solving (2.17) with sparse data of the initial condition using our approach and the vanilla PINN method. For the noisy-data case, the error is evaluated on the predicted mean.

regression task with 41 measurements of u_0 equidistantly sampled on $[-1, 1]$ as data for the initial condition, and a uniform mesh 21×41 on the temporal-spatial domain $[0, 1] \times [-1, 1]$ is used for the PDE loss. For the downstream tasks $\mathcal{T}$, we assume that five random measurements of u_0 are available and the same uniform mesh for imposing the physics is employed. The boundary conditions are hard encoded in the NN modeling in both $\{\mathcal{T}^k\}_{k=1}^M$ and $\mathcal{T}$. For the noisy-data case, the noise is assumed to be additive Gaussian noise with mean zero and standard deviation 0.02 for both measurements of u_0 and the equation. We present results in Figure 2.7 and Table 2.3. We can see that our method learns a good functional prior of u_0 , which provides informative prior knowledge for the downstream physics-informed regression tasks in which data are sparse and noisy. The prediction is accurate in both clean- and noisy-data cases, and the errors in the noisy case are bounded by the predicted uncertainty. The L_2 error of u (shown in Table 2.3) indicates that our approach outperforms the PINN method by a significant amount, and hence demonstrates the effectiveness of bringing prior knowledge to solving similar tasks.

2.4.4 2D steady-state nonlinear Allen-Cahn equation

We now move to a 2D steady-state nonlinear Allen-Cahn equation with Dirichlet boundary conditions [315]:

$$\lambda \Delta u + u(u^2 - 1) = f, x, y \in (0, 1), \quad (2.19a)$$

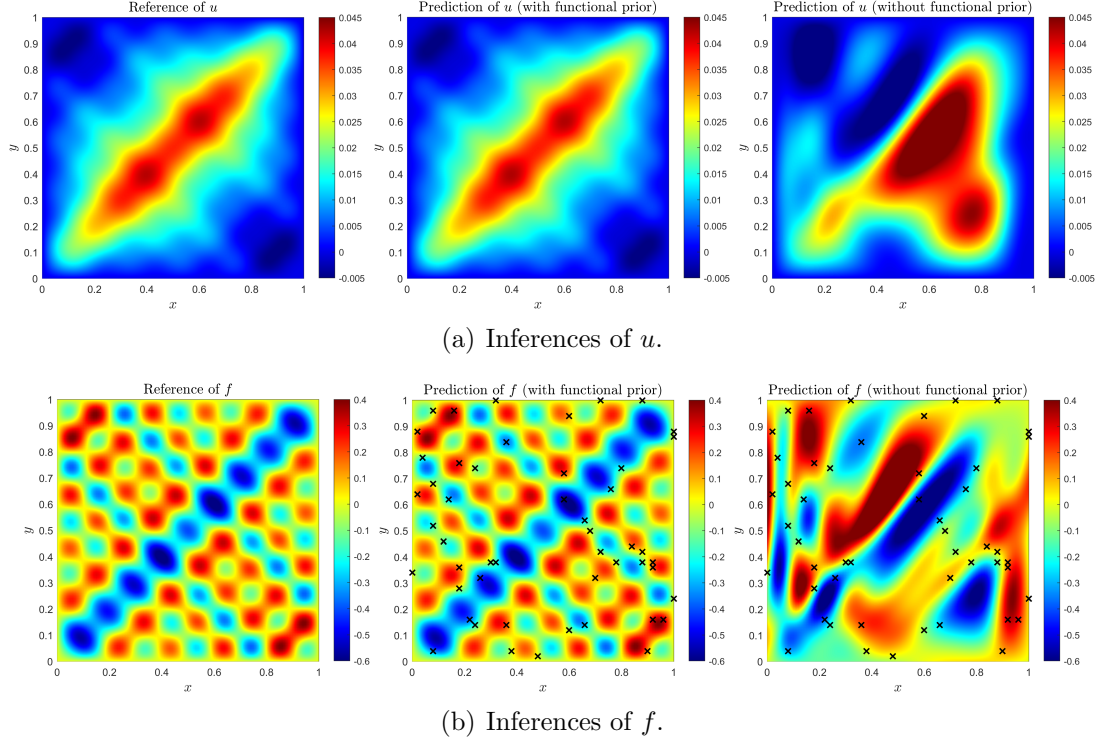


Figure 2.8: Results from solving (2.19) given sparse data of the source term (100 clean measurements randomly sampled across the domain): (a) inferences of u and (b) inferences of f . From left to right we present the reference of u/f , the inference of u/f using the presented method to leverage the prior knowledge, and the inference of u/f using the vanilla PINN method without leveraging the learned functional prior. **Black** crosses represent locations of the measurements of f .

$$u(x, 0) = u(x, 1) = u(0, y) = u(1, y) = 0, \quad (2.19b)$$

where $\lambda = 0.1$ is a constant and f is the source term. Here, we impose a distribution to f , which is derived from (2.19) and the following distribution of the solution u :

$$u(x, y) = \frac{1}{K} \sum_{j=1}^K \xi_j \frac{\sin(j\pi x) \sin(j\pi y)}{j^2 \pi^2}, x, y \in (0, 1), \xi_j \sim U[-1, 1], j = 1, \dots, K, \quad (2.20)$$

In this example, we use our method to learn a functional prior for both u and f from data of f and (2.19), and employ it to solve the downstream task $\mathcal{T}$ with insufficient information regarding the source term f . We set $K = 10$. Notably, this example corresponds to (2.1) with $\mathcal{F}^k, b^k$ being the same among tasks and f^k, u^k

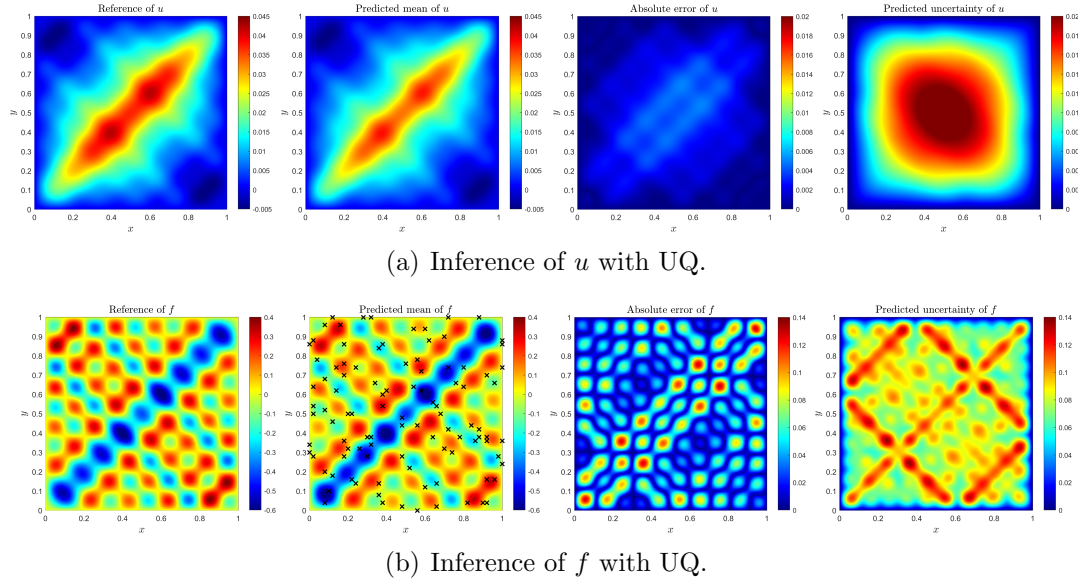


Figure 2.9: Solving (2.19) given data with high level noise: 100 noisy data of f are randomly generated across the domain and corrupted by additive Gaussian noise with mean zero and standard deviation 0.15 (approximate 100% relative noise level): (a) inferences of u and (b) inferences of f . From left to right we present the reference of u/f , the predicted mean of u/f , the absolute error of u/f , and the predicted uncertainty of u/f . **Black** crosses represent locations of the measurements of f .

being task-specific. The dataset for $\mathcal{T}^k$ is $\mathcal{D}^k = \{(x_{i,f}^k, y_{i,f}^k), f_i^k\}_{i=1}^{N_f^k}$.

	vanilla PINN (clean data)	Our approach (clean data)	Our approach (noisy data)
Error	73.14%	0.91%	13.91%

Table 2.4: L_2 relative error of u for solving (2.19) with sparse data of the source term using our approach and the vanilla PINN method. For the noisy-data case, the error is evaluated on the predicted mean.

For the training of functional prior, we sample $M = 2,000$ f from (2.20), each of which is resolved with a 51×51 uniform mesh on $[0, 1] \times [0, 1]$. In this example, we randomly sample another function from (2.20) and consider a clean-data case and a noisy-data case for the downstream tasks. In both cases, 100 sparse random measurements of the source term across the spatial domain are assumed to be available (see Figure 2.8(b) for the locations of the measurements). In the noisy-data case, we further assume that these measurements are corrupted by an additive Gaussian noise with mean zero and standard deviation 0.15. Notably, the average absolute

value of the clean measurements is $\frac{1}{N_f} \sum_{i=1}^{N_f} |f_i| = 0.15468$, which makes the standard deviation 0.15 of the additive Gaussian noise a high-level (approximate 100% relative) noise scale.

Results are presented in Table 2.4 and Figures 2.8 and 2.9 for the clean-data and noisy-data cases, respectively. We observe that leveraging prior information is essential for solving (2.19) with sparse and noisy data of the source term. Our approach yields accurate solutions and predictions of the source term, while the vanilla PINN method fails due to not utilizing functional priors. As shown in Figure 2.9, our approach is also able to provide UQ: the predicted means agree with the references and the errors are mostly bounded by the predicted uncertainties.

2.4.5 A high-dimensional parametric Helmholtz equation

The last example we test in this chapter is the 2D Helmholtz equation with stochastic source term and Dirichlet boundary conditions [229]:

$$(\lambda^2 - \nabla^2)u = f, x, y \in (0, 2\pi), \quad (2.21a)$$

$$u(x, 0) = u(x, 2\pi) = u(0, y) = u(2\pi, y) = 0, \quad (2.21b)$$

where λ^2 is the Helmholtz constant and f is defined as follows:

$$\begin{aligned} f(x, y) = \frac{4}{K} \{ & \sum_{j=1}^{K/4} \xi_j \sin(j \frac{x - \pi}{2\pi}) \sin(j \frac{y - \pi}{2\pi}) + \xi_{j+\frac{K}{4}} \sin(j \frac{x - \pi}{2\pi}) \cos(j \frac{y - \pi}{2\pi}) + \\ & \xi_{j+\frac{2K}{4}} \cos(j \frac{x - \pi}{2\pi}) \sin(j \frac{y - \pi}{2\pi}) + \xi_{j+\frac{3K}{4}} \cos(j \frac{x - \pi}{2\pi}) \cos(j \frac{y - \pi}{2\pi}) \}, \end{aligned} \quad (2.22)$$

where $\xi_j, j = 1, \dots, K$ are i.i.d. random variables subject to uniform distribution $U[-1, 1)$ and K represents the dimension of the randomness. In this example, we

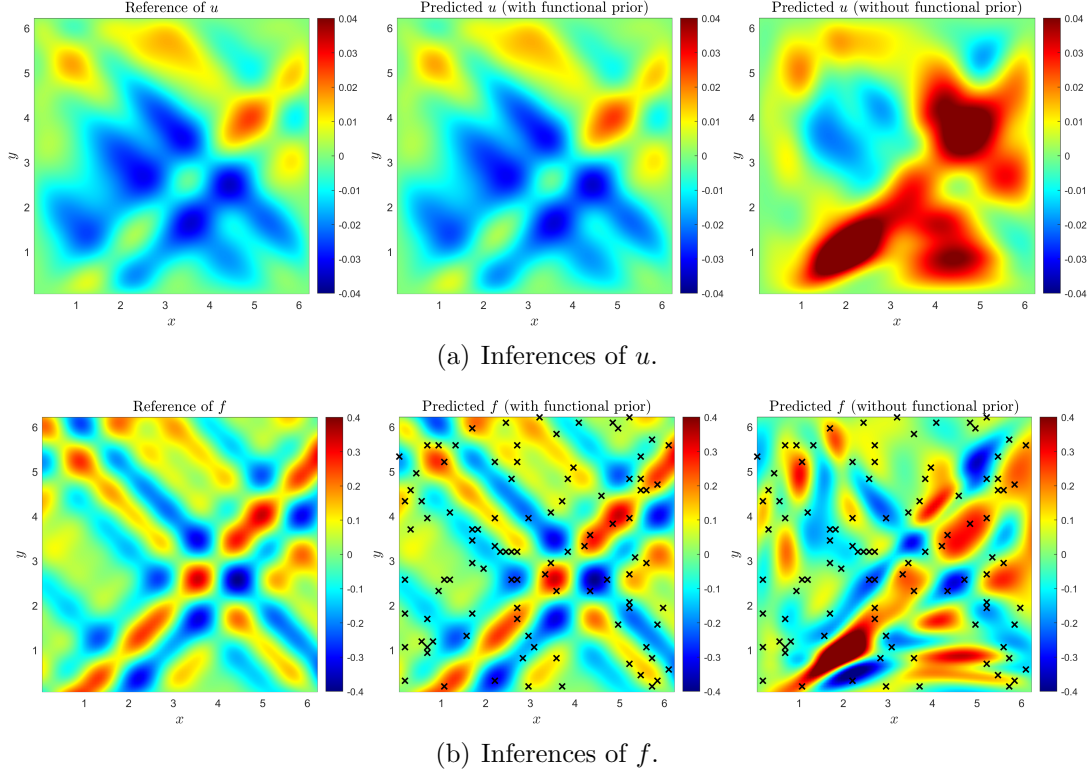


Figure 2.10: Results from solving (2.21) given sparse data of the source term (100 clean measurements randomly sampled across the domain): (a) inferences of u and (b) inferences of f . From left to right we present the reference of u/f , the inference of u/f using the presented method to leverage the prior knowledge, and the inference of u/f using the vanilla PINN method without leveraging the learned functional prior. **Black** crosses represent locations of the measurements of f .

consider the case where $\lambda^2 = 1$ is known in all $\{\mathcal{T}^k\}_{k=1}^M$, which corresponds to (2.1) with $\mathcal{F}^k, b^k$ being shared among tasks and u^k, f^k being task-specific. The dataset for $\mathcal{T}^k$ is $\mathcal{D}^k = \{(x_{i,f}^k, y_{i,f}^k), f_i^k\}_{i=1}^{N_f^k}$. The downstream tasks are defined as the same but with fewer and noisy measurements. We set $K = 120$ in (2.22), making it a 120-dimensional problem.

For the training of functional prior, we sample $M = 10,000$ functions from (2.22) and hence 10,000 tasks are solved simultaneously with MH-PINNs with boundary conditions hard-encoded in the NN modeling. Each sample of f is resolved by a 50×50 uniform mesh on $[0, 2\pi] \times [0, 2\pi]$ with boundary excluded. Reference solutions

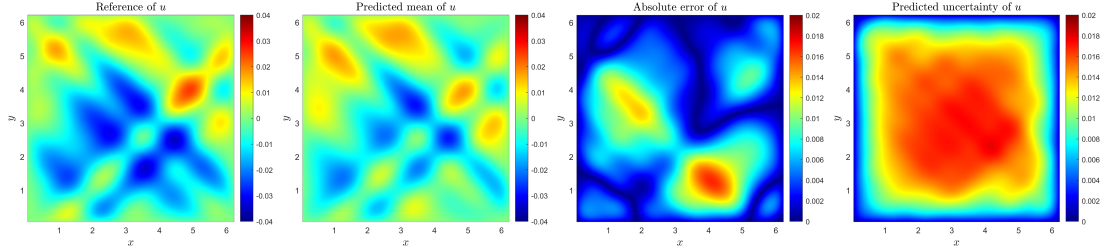
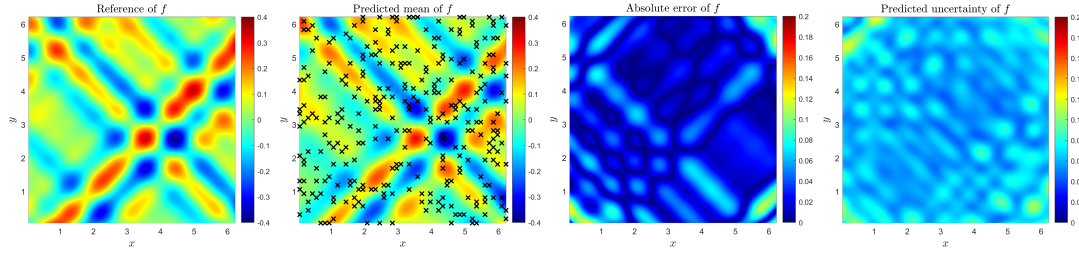
(a) Inference of u with UQ.(b) Inference of f with UQ.

Figure 2.11: Solving (2.21) given data with high level noise: 300 noisy data of f are randomly generated across the domain and corrupted by additive Gaussian noise with mean zero and standard deviation 0.1 (approximate 100% relative noise level): (a) inferences of u and (b) inferences of f . From left to right we present the reference of u/f , the predicted mean of u/f , the absolute error of u/f , and the predicted uncertainty of u/f . **Black** crosses represent locations of the measurements of f .

of u are obtained by solving (2.21) using the finite difference method with five-point stencil and a 101×101 uniform mesh on $[0, 2\pi] \times [0, 2\pi]$. In this example, we randomly sample another function from (2.22) and consider both a clean-data case and a noisy-data case for the downstream tasks. In the clean-data case, we assume that we have access to 100 clean measurements of the source term, which are randomly sampled across the spatial domain (shown in Figure 2.10(b)). In the noisy-data case, we assume that 300 measurements of the source term are available: they are randomly sampled across the spatial domain (shown in Figure 2.11(b)) and are corrupted by an additive Gaussian noise with mean zero and standard deviation 0.1. We note that the average absolute value of the exact of these 300 measurements is $\frac{1}{N_f} \sum_{i=1}^{N_f} |f_i| = 0.1008$, making the standard deviation 0.1 of the Gaussian noise a high-level (100% relative) noise scale.

	vanilla PINN (clean data)	Our approach (clean data)	Our approach (noisy data)
Error	189.49%	2.55%	49.45%

Table 2.5: L_2 relative error of u for solving (2.21) with sparse data of the source term using our approach and the vanilla PINN method. For the noisy-data case, the error is evaluated based on the predicted mean.

We present the results in Table 2.5 and in Figures 2.10 and 2.11 for the clean-data and noisy-data cases, respectively. We notice that leveraging prior knowledge using our approach significantly improves the results for both the solution and the source term when data are sparse and clean in solving (2.21), compared to the vanilla PINN method which does not consider any historical information. As shown in Table 2.5, the accuracy of the solution in the noisy-data case is not as good due to the high-level (100% relative) noise scale and high-dimensionality of this problem. However, it is still improved and the errors between the predicted means and the references are mostly bounded by the predicted uncertainties for both the solution and the source term.

2.5 Summary

We have developed multi-head physics-informed neural networks (MH-PINNs) combined with normalizing flows (NFs) for learning functional priors from data and physics. The method is executed in two stages: (1) training a MH-PINN for a shared representation and multiple heads, and (2) estimating a distribution from these multiple heads via an NF. We have demonstrated the effectiveness of this approach in learning functional priors in approximating stochastic functions, both with and without differential equations. We have also evaluated the performance and computational cost of various NFs in both training and sampling, and we have compared our approach to physics-informed generative adversarial networks (PI-

GANs) [317], which have been employed to address similar problems in the literature [201, 348, 234].

Additionally, we also proposed two methods within the PINN framework to leverage the learned functional prior for downstream physics-informed regression tasks, tackling the issues of data sparsity and high-level noise with uncertainty quantification (UQ). Specifically, for deterministic settings, we solved a regularized physics-informed regression problem, where the learned functional prior provides the regularization. For Bayesian inference, we constructed a posterior in which the learned functional prior serves as the prior distribution. We showcased these methods in a series of numerical examples that address sparse and noisy data: (1) a function approximation example with sparse measurements, (2) an inverse nonlinear ODE system with limited data of the solution and the source term, (3) a time-dependent PDE with incomplete information of the initial condition, (4) a 2D steady-state nonlinear Allen-Cahn equation with sparse data and high-level noise, and (5) a high-dimensional 2D parametric Helmholtz equation with sparse data and high-level noise. The presented approach provides an effective and adaptable way to integrate prior knowledge and historical data within the PINNs framework, making it readily applicable to a broad range of practical problems in computational science and engineering.

CHAPTER THREE

Leveraging viscous Hamilton-Jacobi (HJ) PDEs for uncertainty quantification in scientific machine learning

3.1 Introduction

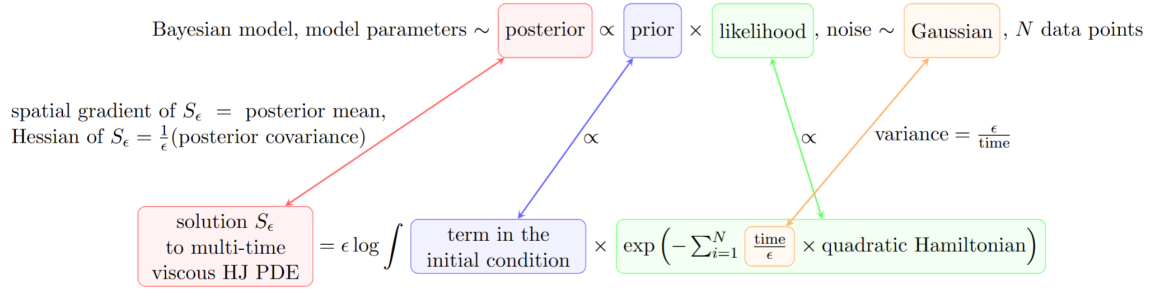


Figure 3.1: (See Section 3.2) Illustration of a connection between a Bayesian inference problem in scientific machine learning (**top**) and the solution to a multi-time viscous HJ PDE (**bottom**). The colors indicate the associated quantities between problems. This color scheme is reused in the subsequent illustrations of our connection. The arrow labels describe how the boxed quantities are related. For example, the posterior mean in the learning problem is equivalent to the spatial gradient of the solution to the multi-time viscous HJ PDE (**red**).

Scientific machine learning (SciML) is a recent and evolving field consisting of data-driven techniques for learning physics-based models [244, 142, 306, 60, 219, 242, 243, 344, 49, 260, 50, 346, 315, 201, 347, 196] that finds applications in diverse areas [190, 34, 32, 258, 204, 175, 325, 334, 67]. While deterministic SciML models demonstrate impressive predictive power, real-life applications often require quantitative metrics for the trustworthiness of learned models. To this end, uncertainty quantification (UQ) [131, 234, 348] has been integrated with SciML to create new learning approaches that produce quantifiably high-confidence models in the absence of a ground truth and use these confidence metrics to help inform the learning process.

In this chapter, we reinterpret certain Bayesian inference problems arising from SciML and UQ using the framework of viscous Hamilton-Jacobi partial differential equations (HJ PDEs) [79, 326, 81]. Specifically, we develop a new theoretical connection between these Bayesian inference problems and multi-time viscous HJ PDEs that shows that the posterior mean and covariance of the Bayesian model can be recovered from the spatial gradient and Hessian of the solution to an associated viscous

HJ PDE (Section 3.2). This connection is summarized in Figure 3.1. Our previous work in [47, 46] establishes similar theoretical connections between deterministic SciML problems and inviscid HJ PDEs. As such, the methodology in [47, 46] does not automatically extend to UQ. Meanwhile, the work in [63] establishes a similar link between Bayesian inference and viscous HJ PDEs but develops this connection via the Cole-Hopf transformation [79] and only considers the single-time case. In contrast, we do not rely on the Cole-Hopf transformation and consider multi-time HJ PDEs to enable our connection to SciML.

As a first exploration of this connection, we specialize to Bayesian inference problems with linear models, Gaussian likelihoods, and Gaussian priors (Section 3.3). In this case, the corresponding viscous HJ PDE (and hence, the Bayesian inference problem) can be solved using Riccati ODEs [193]. The resulting Riccati-based methodology can then be leveraged to efficiently update the model predictions (Section 3.3.2) and continuously tune hyperparameters (Section 3.3.3). To illustrate the potential advantages of this approach, we apply this Riccati-based methodology to several UQ-based examples from SciML (Section 3.4). In particular, we focus on noisy data cases and quantify *epistemic uncertainty* [131, 234, 348], which refers to the uncertainty associated with the model parameters. In Section 3.4.1, we demonstrate the potential computational advantages of our approach over more standard SciML/UQ techniques when incrementally updating the model predictions. Namely, we show how our Riccati-based approach naturally coincides with continual learning while inherently avoiding catastrophic forgetting despite updating the model predictions without accessing the historical data (Section 3.4.1.1). We also show how solving the Riccati ODEs provides a continuous flow of solutions with respect to the hyperparameters of the model, which allows the hyperparameters to be tuned continuously (Section 3.4.1.2). In Section 3.4.2, we demonstrate the versatility of

our Riccati-based approach in handling different learning scenarios, e.g., big data (Section 3.4.2.1) and active learning (Section 3.4.2.2) settings. In particular, these examples illustrate how uncertainty metrics can be leveraged to inform how we learn. In Section 3.4.3, we highlight the invariance of our Riccati-based approach to the order of the data, which provides flexibility in how the data is sampled.

The contributions of this work can be summarized as follows:

- new mathematical theory connecting certain Bayesian inference problems arising from SciML to viscous HJ PDEs,
- new Riccati-based methodology for solving Bayesian inference problems with linear models, Gaussian likelihoods, and Gaussian priors, and
- detailed experimental demonstrations of the potential computational advantages of this Riccati-based approach across a variety of learning applications.

This work presents exciting opportunities to advance the theoretical foundations of SciML and thus to create new interpretable SciML methods. While we demonstrate promising results, further research is needed to extend the work presented here to more general learning settings. We discuss some possible future research directions of this work in Section 3.5. Some additional technical details and numerical results are provided in the Appendix.

3.2 Connecting viscous HJ PDEs to Bayesian inference in machine learning

In this section, we establish a new theoretical connection between viscous HJ PDEs and Bayesian inference in machine learning (ML). Specifically, we formulate a general regression task in the Bayesian framework and then connect it to a viscous HJ PDE with quadratic Hamiltonian.

3.2.1 Bayesian inference in machine learning

Bayesian inference is often adopted in ML due to its effectiveness in integrating prior information, handling noisy data, providing uncertainty quantification and, hence, in making quantifiably trustworthy predictions. Consider the following model [306]:

$$\mathbf{y} = u_{\boldsymbol{\theta}}(\mathbf{z}) + \xi, \quad (3.1)$$

where $\mathbf{z} \in \mathbb{R}^\ell$ is the input, $\mathbf{y} \in \mathbb{R}^m$ is the observed output, $u_{\boldsymbol{\theta}} : \mathbb{R}^\ell \rightarrow \mathbb{R}^m$ is the model parameterized by $\boldsymbol{\theta} \in \mathbb{R}^n$, and $\xi \in \mathbb{R}^m$ is additive noise. The goal is to learn $u_{\boldsymbol{\theta}}$ from data $\mathcal{D} = \{(\mathbf{z}_i, \mathbf{y}_i)\}_{i=1}^N$, where $\mathbf{z}_i, \mathbf{y}_i$ denote measurements of $\mathbf{z}, \mathbf{y}$, respectively. Specifically, we learn the model $u_{\boldsymbol{\theta}}$ by estimating $\boldsymbol{\theta}$ from the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ (e.g., $\boldsymbol{\theta}$ could be the posterior mean estimator $\mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]$ or the maximum a posteriori (MAP) estimator $\arg \max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D})$). By Bayes' theorem, $p(\boldsymbol{\theta}|\mathcal{D})$ can be computed as

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}, \quad (3.2)$$

where $p(\mathcal{D}|\boldsymbol{\theta})$ is the likelihood, $p(\boldsymbol{\theta})$ is the prior, and $p(\mathcal{D}) = \int_{\mathbb{R}^n} p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$ is the marginal likelihood, which is independent of $\boldsymbol{\theta}$ and often treated as a normal-

izing constant. From these distributions, we obtain quantities that can be used to statistically analyze and quantify the uncertainty in the learned model arising from the noisy data and/or the prior on $\boldsymbol{\theta}$.

The posterior $p(\boldsymbol{\theta}|\mathcal{D})$ often becomes analytically intractable [234, 348], particularly when either the likelihood or the prior is complicated. In these scenarios, estimation techniques, such as Markov Chain Monte Carlo (MCMC) [211], are typically employed. Here, we create a new theoretical connection between HJ PDEs and Bayesian inference in ML that yields opportunities for new HJ PDE-based methods for estimating $\boldsymbol{\theta}$ and $p(\boldsymbol{\theta}|\mathcal{D})$. As a first exploration of this connection, we consider the case where the likelihood $p(\mathcal{D}|\boldsymbol{\theta})$ follows a Gaussian distribution and the model $u_{\boldsymbol{\theta}}$ depends linearly on $\boldsymbol{\theta}$. In other words, we consider the case where the data is perturbed by additive white noise (i.e., ξ is Gaussian), and we learn the model $u_{\boldsymbol{\theta}}(\mathbf{z}) = \Phi(\mathbf{z})\boldsymbol{\theta}$, where $\Phi : \mathbb{R}^{\ell} \rightarrow \mathbb{R}^{m \times n}$. With some assumptions on the prior, this scenario yields a connection to a viscous HJ PDE with a quadratic Hamiltonian.

3.2.2 Connection to viscous HJ PDEs

In this section, we detail the connection between certain Bayesian inference problems with Gaussian likelihood and viscous HJ PDEs with quadratic Hamiltonian. Specifically, we consider the case where we learn the model $u_{\boldsymbol{\theta}}(\mathbf{z}) = \Phi(\mathbf{z})\boldsymbol{\theta}$, where $\Phi : \mathbb{R}^{\ell} \rightarrow \mathbb{R}^{m \times n}$ and the posterior of the parameter $\boldsymbol{\theta}$ is estimated using the Bayesian framework described in Section 3.2.1. We assume that the likelihood is Gaussian given by $p(\mathcal{D}|\boldsymbol{\theta}) \propto \prod_{i=1}^N \exp\left(-\frac{1}{2\sigma_i^2}\|\Phi(\mathbf{z}_i)\boldsymbol{\theta} - \mathbf{y}_i\|^2\right)$, where σ_i^2 is the variance of the i -th data point $(\mathbf{z}_i, \mathbf{y}_i)$. We assume that the prior distribution is given by $p(\boldsymbol{\theta}) \propto \exp(-\frac{1}{\epsilon}(-\langle \mathbf{x}, \boldsymbol{\theta} \rangle + g(\boldsymbol{\theta})))$, where $\epsilon > 0$ is a hyperparameter and $g : \mathbb{R}^n \rightarrow \mathbb{R}$ is such that $\boldsymbol{\theta} \mapsto \exp(-\frac{1}{\epsilon}(-\langle \mathbf{x}, \boldsymbol{\theta} \rangle + g(\boldsymbol{\theta})))$ is integrable for every $\mathbf{x} \in \mathbb{R}^n$. Hence,

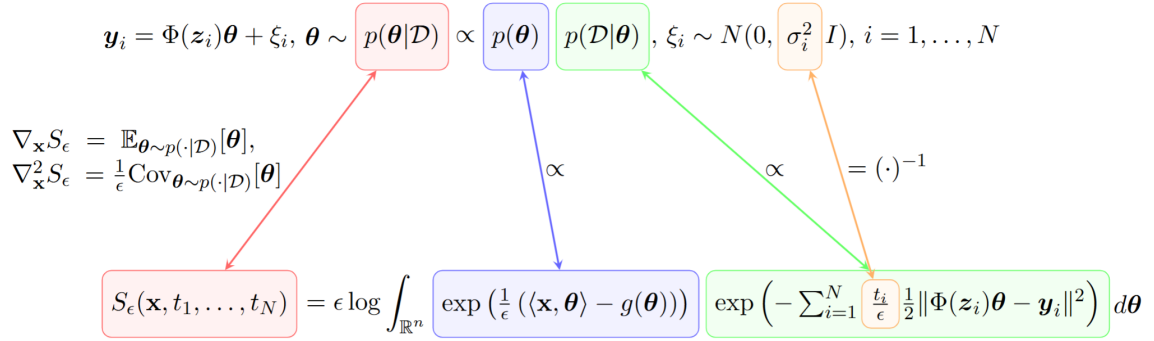


Figure 3.2: (See Section 3.2) Mathematical formulation of the connection between a Bayesian inference problem with linear model and Gaussian likelihood (**top**) and the solution to a multi-time viscous HJ PDE with quadratic Hamiltonian (**bottom**). The content of this illustration matches that of Figure 3.1 by replacing each term in Figure 3.1 with its corresponding mathematical expression. The colors indicate the associated quantities. The arrow labels describe how the boxed quantities are related.

if g is quadratic, then the prior distribution is also Gaussian. Note that we use $p(\boldsymbol{\theta}) \propto f(\boldsymbol{\theta})$ to denote that the probability density function $p(\boldsymbol{\theta})$ is proportional to $f(\boldsymbol{\theta})$, or, in other words, $p(\boldsymbol{\theta}) = \frac{f(\boldsymbol{\theta})}{\int_{\mathbb{R}^n} f(w)dw}$. Similarly, for conditional probabilities, $p(\boldsymbol{\theta}|\mathcal{D}) \propto f(\boldsymbol{\theta}, \mathcal{D})$ means that $p(\boldsymbol{\theta}|\mathcal{D}) = \frac{f(\boldsymbol{\theta}, \mathcal{D})}{\int_{\mathbb{R}^n} f(w, \mathcal{D})dw}$.

Remark 3.2.1. The particular form of the likelihood that we consider corresponds to the data $\mathcal{D}$ being produced by $\mathbf{y}_i = u_{\boldsymbol{\theta}}(\mathbf{z}_i) + \xi_i$, where $\xi_1, \dots, \xi_N$ are independent, each ξ_i represents Gaussian noise with zero mean and covariance matrix $\sigma_i^2 I$ (I is the $m \times m$ identity matrix), and $u_{\boldsymbol{\theta}}(\mathbf{z}) = \Phi(\mathbf{z})\boldsymbol{\theta}$. The results in this chapter have a straightforward generalization to the case where $\xi_1, \dots, \xi_N$ are independent, each ξ_i is a general Gaussian, and $u_{\boldsymbol{\theta}}(\mathbf{z}) = \Phi(\mathbf{z})\boldsymbol{\theta} + c(\mathbf{z})$, where $c: \mathbb{R}^\ell \rightarrow \mathbb{R}^m$ is a function independent of $\boldsymbol{\theta}$. However, for simplicity, we only present the case where $c(\mathbf{z}) = 0$.

First consider the case where we have $N = 1$ data point $(\mathbf{z}_1, \mathbf{y}_1)$. Denote $\Phi_1 := \Phi(\mathbf{z}_1)$. Define a Hamiltonian $H: \mathbb{R}^n \rightarrow \mathbb{R}$ by $H(\boldsymbol{\theta}) = \frac{1}{2} \|\Phi_1 \boldsymbol{\theta} - \mathbf{y}_1\|^2$ and a function

$S_\epsilon: \mathbb{R}^n \times [0, \infty) \rightarrow \mathbb{R}$ by

$$S_\epsilon(\mathbf{x}, t) = \epsilon \log \int_{\mathbb{R}^n} \exp \left(\frac{1}{\epsilon} (\langle \mathbf{x}, \boldsymbol{\theta} \rangle - tH(\boldsymbol{\theta}) - g(\boldsymbol{\theta})) \right) d\boldsymbol{\theta}. \quad (3.3)$$

Then, S_ϵ satisfies the following viscous HJ PDE:

$$\begin{cases} \partial_t S_\epsilon(\mathbf{x}, t) + \frac{1}{2} \|\Phi_1 \nabla_{\mathbf{x}} S_\epsilon(\mathbf{x}, t) - \mathbf{y}_1\|^2 + \frac{\epsilon}{2} \nabla_{\mathbf{x}} \cdot (\Phi_1^T \Phi_1 \nabla_{\mathbf{x}} S_\epsilon(\mathbf{x}, t)) = 0 & \mathbf{x} \in \mathbb{R}^n, t > 0, \\ S_\epsilon(\mathbf{x}, 0) = J(\mathbf{x}) & \mathbf{x} \in \mathbb{R}^n, \end{cases} \quad (3.4)$$

where the initial condition J is defined by

$$J(\mathbf{x}) = \epsilon \log \int_{\mathbb{R}^n} \exp \left(\frac{1}{\epsilon} (\langle \mathbf{x}, \boldsymbol{\theta} \rangle - g(\boldsymbol{\theta})) \right) d\boldsymbol{\theta}. \quad (3.5)$$

Using this HJ PDE framework, the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ can be characterized by $p(\boldsymbol{\theta}|\mathcal{D}) \propto \exp \left(\frac{1}{\epsilon} (\langle \mathbf{x}, \boldsymbol{\theta} \rangle - tH(\boldsymbol{\theta}) - g(\boldsymbol{\theta})) \right)$ by setting the variance of the noise to $\sigma_1^2 = \frac{\epsilon}{t}$. By straightforward computation, the derivatives of S_ϵ are given by

$$\begin{aligned} \partial_t S_\epsilon(\mathbf{x}, t) &= - \frac{\int_{\mathbb{R}^n} H(\boldsymbol{\theta}) \exp \left(\frac{1}{\epsilon} (\langle \mathbf{x}, \boldsymbol{\theta} \rangle - tH(\boldsymbol{\theta}) - g(\boldsymbol{\theta})) \right) d\boldsymbol{\theta}}{\int_{\mathbb{R}^n} \exp \left(\frac{1}{\epsilon} (\langle \mathbf{x}, \boldsymbol{\theta} \rangle - tH(\boldsymbol{\theta}) - g(\boldsymbol{\theta})) \right) d\boldsymbol{\theta}} = -\mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[H(\boldsymbol{\theta})], \\ \nabla_{\mathbf{x}} S_\epsilon(\mathbf{x}, t) &= \frac{\int_{\mathbb{R}^n} \boldsymbol{\theta} \exp \left(\frac{1}{\epsilon} (\langle \mathbf{x}, \boldsymbol{\theta} \rangle - tH(\boldsymbol{\theta}) - g(\boldsymbol{\theta})) \right) d\boldsymbol{\theta}}{\int_{\mathbb{R}^n} \exp \left(\frac{1}{\epsilon} (\langle \mathbf{x}, \boldsymbol{\theta} \rangle - tH(\boldsymbol{\theta}) - g(\boldsymbol{\theta})) \right) d\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}], \\ \nabla_{\mathbf{x}}^2 S_\epsilon(\mathbf{x}, t) &= \frac{1}{\epsilon} \left(\mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})} [\boldsymbol{\theta} \boldsymbol{\theta}^T] - \mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}] \mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]^T \right) = \frac{1}{\epsilon} \text{Cov}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}], \end{aligned} \quad (3.6)$$

all of which are related to the posterior expectation and posterior covariance. Namely, computing the first and second derivatives of the solution to the viscous HJ PDE (3.4) is equivalent (up to a multiplicative constant) to computing the first and second moments of the posterior distribution, respectively, and the posterior mean estimator for $\boldsymbol{\theta}$ can be computed as $\nabla_{\mathbf{x}} S_\epsilon(\mathbf{x}, t)$.

These connections can be easily generalized to the case with multiple data points ($N > 1$). Denote $\Phi_i = \Phi(\mathbf{z}_i)$, and set $\frac{\epsilon}{t_i} = \sigma_i^2$ for each $i = 1, \dots, N$. The posterior distribution then becomes $p(\boldsymbol{\theta}|\mathcal{D}) \propto \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \sum_{i=1}^N \frac{t_i}{2} \|\Phi_i \boldsymbol{\theta} - \mathbf{y}_i\|^2 - g(\boldsymbol{\theta}))\right)$. This learning problem is related to a multi-time viscous HJ PDE as follows. Define the i -th Hamiltonian $H_i : \mathbb{R}^n \rightarrow \mathbb{R}$ by $H_i(\boldsymbol{\theta}) = \frac{1}{2} \|\Phi_i \boldsymbol{\theta} - \mathbf{y}_i\|^2$ and the function $S_\epsilon : \mathbb{R}^n \times [0, \infty)^N \rightarrow \mathbb{R}$ by

$$S_\epsilon(\mathbf{x}, t_1, \dots, t_N) = \epsilon \log \int_{\mathbb{R}^n} \exp\left(\frac{1}{\epsilon} \left(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \sum_{i=1}^N \frac{t_i}{2} \|\Phi_i \boldsymbol{\theta} - \mathbf{y}_i\|^2 - g(\boldsymbol{\theta}) \right)\right) d\boldsymbol{\theta}. \quad (3.7)$$

Then, S_ϵ satisfies the following multi-time viscous HJ PDE:

$$\begin{cases} \partial_{t_i} S_\epsilon(\mathbf{x}, t_1, \dots, t_N) + \frac{1}{2} \|\Phi_i \nabla_{\mathbf{x}} S_\epsilon(\mathbf{x}, t_1, \dots, t_N) - \mathbf{y}_i\|^2 + \frac{\epsilon}{2} \nabla_{\mathbf{x}} \cdot (\Phi_i^T \Phi_i \nabla_{\mathbf{x}} S_\epsilon(\mathbf{x}, t_1, \dots, t_N)) = 0 & \mathbf{x} \in \mathbb{R}^n, t_i > 0, i = 1, \dots, N, \\ S_\epsilon(\mathbf{x}, 0, \dots, 0) = J(\mathbf{x}) & \mathbf{x} \in \mathbb{R}^n, \end{cases} \quad (3.8)$$

where the initial condition J is defined in (3.5). Using the notation of the Hamiltonians H_i , the posterior distribution is also given by $p(\boldsymbol{\theta}|\mathcal{D}) \propto \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \sum_{i=1}^N t_i H_i(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right)$. With straightforward computation, the derivatives of S_ϵ are given by

$$\begin{aligned} \partial_{t_i} S_\epsilon(\mathbf{x}, t_1, \dots, t_N) &= - \frac{\int_{\mathbb{R}^n} H_i(\boldsymbol{\theta}) \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \sum_{i=1}^N t_i H_i(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}}{\int_{\mathbb{R}^n} \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \sum_{i=1}^N t_i H_i(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}} = -\mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[H_i(\boldsymbol{\theta})], \\ \nabla_{\mathbf{x}} S_\epsilon(\mathbf{x}, t_1, \dots, t_N) &= \frac{\int_{\mathbb{R}^n} \boldsymbol{\theta} \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \sum_{i=1}^N t_i H_i(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}}{\int_{\mathbb{R}^n} \exp\left(\frac{1}{\epsilon}(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \sum_{i=1}^N t_i H_i(\boldsymbol{\theta}) - g(\boldsymbol{\theta}))\right) d\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}], \\ \nabla_{\mathbf{x}}^2 S_\epsilon(\mathbf{x}, t_1, \dots, t_N) &= \frac{1}{\epsilon} (\mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta} \boldsymbol{\theta}^T] - \mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}] \mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]^T) = \frac{1}{\epsilon} \text{Cov}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]. \end{aligned} \quad (3.9)$$

As in the $N = 1$ case, $\nabla_{\mathbf{x}} S_\epsilon(\mathbf{x}, t_1, \dots, t_N)$ and $\epsilon \nabla_{\mathbf{x}}^2 S_\epsilon(\mathbf{x}, t_1, \dots, t_N)$ are equivalent to the posterior expectation and posterior covariance matrix, respectively. These connections can then be leveraged to compute the posterior mean estimator and to estimate the uncertainty in the learned model. Figure 3.2 summarizes the connec-

tions established above between Bayesian inference problems and multi-time viscous HJ PDEs.

3.3 Riccati-based methodology

In the previous section, we established a new theoretical connection between Bayesian inference problems with linear models and additive Gaussian noise and viscous HJ PDEs with quadratic Hamiltonians. In this section, we show how this connection can be leveraged to reuse existing HJ PDE solvers to create new efficient training methods for Bayesian inference. While the theoretical connection holds more generally, in this section, we focus on the case where the prior distribution is also Gaussian. In this case, the corresponding viscous HJ PDEs can be solved using Riccati ODEs, which in turn can be used to solve the original Bayesian inference problem of interest. Specifically, we develop a new Riccati-based approach for solving these Bayesian inference problems that

1. can add and remove data points from the training dataset invariant to the order of the data,
2. does not require retraining on or access to previously incorporated data to update the learned model, which may provide computational benefits, particularly when the posterior distribution must be re-computed many times, and
3. yields a continuous flow of solutions, which allows the hyperparameters in the learning problem to be tuned continuously.

3.3.1 Connection using Gaussian prior to Riccati ODEs

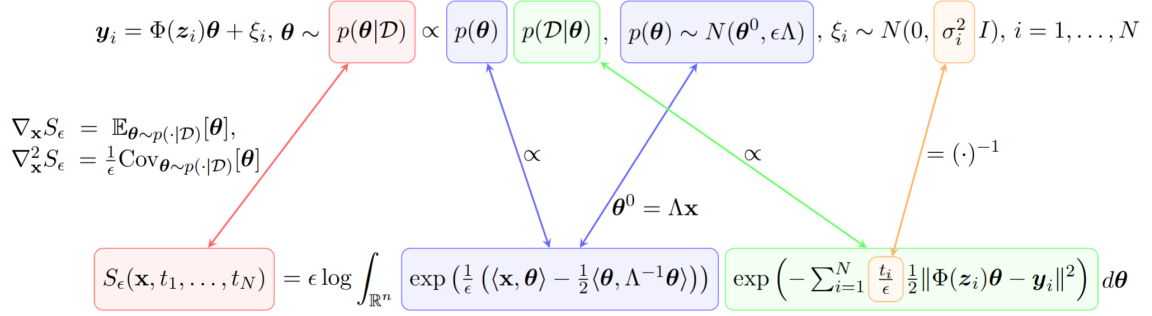


Figure 3.3: (See Section 3.3.1) Mathematical formulation of the connection between a Bayesian inference problem with linear model, Gaussian likelihood, and Gaussian prior (**top**) and the solution to a multi-time viscous HJ PDE with quadratic Hamiltonian and quadratic initial condition (**bottom**). The content of this illustration is a special case of the connection in Figure 3.2, where the prior is Gaussian (set $g(\theta) = \frac{1}{2}\langle \theta, \Lambda^{-1}\theta \rangle$ in Figure 3.2). The colors indicate the associated quantities. The arrow labels describe how the boxed quantities are related.

In this section, we discuss the special case where the prior distribution is Gaussian with mean $\theta^0 = \Lambda \mathbf{x}$ and covariance matrix $\epsilon\Lambda$, where $\Lambda \in \mathbb{R}^{n \times n}$ is symmetric positive definite. Note that this choice of mean and covariance is not as restrictive as the notation suggests since we have freedom in picking $\mathbf{x} \in \mathbb{R}^n$ (picking a different $\mathbf{x}$ corresponds to evaluating the associated viscous HJ PDE at a different point in space); as long as Λ is invertible (i.e., the prior is a non-degenerate Gaussian), θ^0 can be chosen to be any point in $\mathbb{R}^n$. This particular prior corresponds to setting $g(\theta) = \frac{1}{2}\langle \theta, \Lambda^{-1}\theta \rangle$, and the initial condition (3.5) of the associated viscous HJ PDE becomes $J(\mathbf{x}) = \frac{1}{2}\langle \mathbf{x}, \Lambda \mathbf{x} \rangle + \frac{\epsilon}{2} \log \det(\Lambda) + \frac{\epsilon n}{2} \log(2\pi\epsilon)$. Since the viscous HJ PDE has quadratic Hamiltonian and quadratic initial condition, its solution S_ϵ can be computed using Riccati ODEs as follows.

We first consider the single data point case ($N = 1$). In this case, the solution S_ϵ to the viscous HJ PDE (3.4) is given by $S_\epsilon(\mathbf{x}, t) = \frac{1}{2}\mathbf{x}^T P(t)\mathbf{x} + \mathbf{q}(t)^T \mathbf{x} + r(t)$, where $P: [0, +\infty) \rightarrow \mathbb{R}^{n \times n}$, $\mathbf{q}: [0, +\infty) \rightarrow \mathbb{R}^n$, and $r: [0, +\infty) \rightarrow \mathbb{R}$ satisfy the following

Riccati ODEs:

$$\begin{cases} \dot{P}(s) + P(s)\Phi_1^T\Phi_1P(s) = 0, \\ \dot{\mathbf{q}}(s) + P(s)\Phi_1^T(\Phi_1\mathbf{q}(s) - \mathbf{y}_1) = 0, \\ \dot{r}(s) + \frac{1}{2}\|\Phi_1\mathbf{q}(s) - \mathbf{y}_1\|^2 + \frac{\epsilon}{2}Tr(\Phi_1^T\Phi_1P(s)) = 0, \end{cases} \quad \begin{cases} P(0) = \Lambda, \\ \mathbf{q}(0) = 0, \\ r(0) = \frac{\epsilon}{2}\log\det(\Lambda) + \frac{\epsilon n}{2}\log(2\pi\epsilon). \end{cases} \quad (3.10)$$

The mean and covariance matrix of the posterior distribution are given by $\mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}] = \nabla_{\mathbf{x}}S_{\epsilon}(\mathbf{x}, t) = P(t)\mathbf{x} + \mathbf{q}(t)$ and $\text{Cov}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}] = \epsilon\nabla_{\mathbf{x}}^2S_{\epsilon}(\mathbf{x}, t) = \epsilon P(t)$, respectively. Moreover, the MAP estimator and posterior mean estimator are equivalent in this case (since the posterior is Gaussian) and can be computed as $\boldsymbol{\theta} = \nabla_{\mathbf{x}}S_{\epsilon}(\mathbf{x}, t) = P(t)\mathbf{x} + \mathbf{q}(t)$. Note that none of these formulas involve r , and hence, if we only care about solving the associated learning problem, the ODE for r may be ignored. The ODE system (3.10) also has an analytical solution given by $P(s) = (\Lambda^{-1} + s\Phi_1^T\Phi_1)^{-1}$, $\mathbf{q}(s) = sP(s)\Phi_1^T\mathbf{y}_1$, which can be computed directly when the dimension n is small. However, when n is large, inverting an $n \times n$ matrix becomes inefficient, and using other numerical methods (e.g., Runge-Kutta method, recursive least squares) to solve the Riccati ODEs (3.10) may be preferable.

Now consider the case where we have $N > 1$ data points. Then, the solution S_{ϵ} to the multi-time viscous HJ PDE (3.8) satisfies $S_{\epsilon}(\mathbf{x}, t_1, \dots, t_N) = \frac{1}{2}\mathbf{x}^TP\left(\sum_{i=1}^N t_i\right)\mathbf{x} + \mathbf{q}\left(\sum_{i=1}^N t_i\right)^T\mathbf{x} + r\left(\sum_{i=1}^N t_i\right)$. Here, $P: [0, +\infty) \rightarrow \mathbb{R}^{n \times n}$, $\mathbf{q}: [0, +\infty) \rightarrow \mathbb{R}^n$, and $r: [0, +\infty) \rightarrow \mathbb{R}$ are three continuous functions that satisfy the following ODE system over the interval $s \in \left[\sum_{i=1}^{j-1} t_i, \sum_{i=1}^j t_i\right]$ for each $j = 1, \dots, N$:

$$\begin{cases} \dot{P}(s) + P(s)\Phi_j^T\Phi_jP(s) = 0, \\ \dot{\mathbf{q}}(s) + P(s)\Phi_j^T(\Phi_j\mathbf{q}(s) - \mathbf{y}_j) = 0, \\ \dot{r}(s) + \frac{1}{2}\|\Phi_j\mathbf{q}(s) - \mathbf{y}_j\|^2 + \frac{\epsilon}{2}Tr(\Phi_j^T\Phi_jP(s)) = 0, \end{cases} \quad (3.11)$$

where the initial condition is the same as in (3.10). The posterior mean, posterior covariance, and MAP/posterior mean estimator are computed identically as in the $N = 1$ case but with $t = \sum_{i=1}^N t_i$. This ODE system also has an analytical solution: $P(s) = \left(\Lambda^{-1} + \sum_{i=1}^{j-1} t_i \Phi_i^T \Phi_i + (s - \sum_{i=1}^{j-1} t_i) \Phi_j^T \Phi_j \right)^{-1}$, $\mathbf{q}(s) = P(s) \left(\sum_{i=1}^{j-1} t_i \Phi_i^T \mathbf{y}_i + (s - \sum_{i=1}^{j-1} t_i) \Phi_j^T \mathbf{y}_j \right)$ for $s \in \left[\sum_{i=1}^{j-1} t_i, \sum_{i=1}^j t_i \right]$ and $j = 1, \dots, N$. When n is relatively small and there are no computational restrictions on accessing all N data points at once, these analytical formulas can be used to compute the posterior mean and covariance. However, when the dimension n is large or there are computational restrictions on the storage of or access to the entire dataset, using numerical methods to solve the Riccati ODEs (3.11) may be more advantageous than using the analytical formulas. In particular, our Riccati-based approach provides potential benefits when incrementally updating the learned models. Note that while incremental, online, or recursive Gaussian Process Regression (GPR) techniques [59, 246, 130, 195, 54, 38, 213, 232, 257] could also be employed to sequentially update the learned models, such techniques only provide a discrete perspective for these model updates. In contrast, our Riccati-based approach provides a continuous perspective that gives us access to a continuous flow of solutions that allows for the addition and removal of data points as well as continuous hyperparameter tuning. Detailed discussions of some potential computational advantages are presented in the remainder of Section 3.3. The theoretical connection using a Gaussian prior is summarized in Figure 3.3.

3.3.2 Updating the likelihood

In this section, we discuss how the likelihood can be updated efficiently using our Riccati-based methodology. Namely, we show how data points can be added or re-

moved from the learned model without retraining on or accessing the entire previous training set and in a manner that is invariant to the order of the data points. We also show how this Riccati-based framework naturally yields a continuous flow of solutions that allows hyperparameters to be tuned continuously.

3.3.2.1 Adding and removing data points

First, consider adding a data point $(\mathbf{z}_{N+1}, \mathbf{y}_{N+1})$ to the training set $\mathcal{D}$. Recalling that we assume each data point is independent, the posterior distribution after adding the $(N + 1)$ th data point becomes

$$p(\boldsymbol{\theta}|\mathcal{D}, (\mathbf{z}_{N+1}, \mathbf{y}_{N+1})) \propto p(\boldsymbol{\theta}|\mathcal{D}) \exp\left(-\frac{1}{2\sigma_{N+1}^2}\|\Phi_{N+1}\boldsymbol{\theta} - \mathbf{y}_{N+1}\|^2\right), \quad (3.12)$$

where $\Phi_{N+1} = \Phi(\mathbf{z}_{N+1})$. In the HJ PDE framework, this corresponds to augmenting the multi-time system (3.8) with the following HJ PDE:

$$\partial_{t_{N+1}} S_\epsilon(\mathbf{x}, t_1, \dots, t_{N+1}) + \frac{1}{2}\|\Phi_{N+1}\nabla_{\mathbf{x}} S_\epsilon(\mathbf{x}, t_1, \dots, t_{N+1}) - \mathbf{y}_{N+1}\|^2 + \frac{\epsilon}{2}\nabla_{\mathbf{x}} \cdot (\Phi_{N+1}^T \Phi_{N+1} \nabla_{\mathbf{x}} S_\epsilon(\mathbf{x}, t_1, \dots, t_{N+1})) = 0. \quad (3.13)$$

Thus, adding an additional data point can be interpreted as evolving a new time variable t_{N+1} from 0 to $\frac{\epsilon}{\sigma_{N+1}^2}$, where σ_{N+1}^2 denotes the variance of the $(N + 1)$ th data point. As such, the new posterior mean

$$\mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D}, (\mathbf{z}_{N+1}, \mathbf{y}_{N+1}))}[\boldsymbol{\theta}] = P \left(\sum_{i=1}^{N+1} t_i \right) \mathbf{x} + \mathbf{q} \left(\sum_{i=1}^{N+1} t_i \right) \quad (3.14)$$

and posterior covariance

$$\text{Cov}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D}, (\mathbf{z}_{N+1}, \mathbf{y}_{N+1}))}[\boldsymbol{\theta}] = \epsilon P \left(\sum_{i=1}^{N+1} t_i \right) \quad (3.15)$$

can be computed by solving the Riccati ODE (3.11) (with index $j = N + 1$) on $\left(\sum_{i=1}^N t_i, \sum_{i=1}^{N+1} t_i\right)$ with initial conditions $P\left(\sum_{i=1}^N t_i\right) = \frac{\Sigma_N}{\epsilon}$ and $\mathbf{q}\left(\sum_{i=1}^N t_i\right) = \mu_N - \frac{1}{\epsilon}\Sigma_N\mathbf{x}$, where $\mu_N := \mathbb{E}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]$ and $\Sigma_N := \text{Cov}_{\boldsymbol{\theta} \sim p(\cdot|\mathcal{D})}[\boldsymbol{\theta}]$ denote the original posterior mean and covariance, respectively.

Next, we discuss removing one data point $(\mathbf{z}_k, \mathbf{y}_k)$, $k \in \{1, \dots, N\}$ from $\mathcal{D}$. In the HJ PDE framework, removing the k th data point corresponds to evolving the time variable t_k from $\frac{\epsilon}{\sigma_k^2}$ to 0. Thus, the new posterior mean and covariance resulting from deleting the k th data point can be computed as $P(\sum_{i=1}^N t_i - t_k)\mathbf{x} + \mathbf{q}(\sum_{i=1}^N t_i - t_k)$ and $\epsilon P(\sum_{i=1}^N t_i - t_k)$, respectively, where $P, \mathbf{q}$ are obtained by solving the Riccati ODE (3.11) (with index $j = k$) backwards on $(\sum_{i=1}^N t_i - t_k, \sum_{i=1}^N t_i)$ with terminal condition $P(\sum_{i=1}^N t_i) = \frac{\Sigma_N}{\epsilon}$ and $\mathbf{q}(\sum_{i=1}^N t_i) = \mu_N - \frac{1}{\epsilon}\Sigma_N\mathbf{x}$, where we again denote by μ_N, Σ_N the original posterior mean and covariance using all N data points.

Note that when adding or removing a data point, we do not require access to the entire previous dataset $\mathcal{D}$. Instead, we only require access to the previous posterior mean μ_N , the previous posterior covariance Σ_N , and the data point to be added or removed; all of the information about the rest of the training data remains encoded in the solution to the HJ PDE. As such, our Riccati-based approach provides potential computational and memory advantages when adding or removing data points by allowing the posterior distribution to be updated without having to retrain on the entire dataset. Additionally, note that, due to our assumption of independent data, the posterior mean and covariance are invariant to the order of the data points. As a result, adding or removing data points using our Riccati-based approach is also invariant to the order of the data; i.e., the final learned model will be the same regardless of the order in which data points are added or removed.

3.3.2.2 Tuning the variance

Tuning the variance σ_k^2 of the k th data point can be done using a similar methodology as that for adding or removing a data point. If we decrease the hyperparameter from σ_k^2 to $\tilde{\sigma}_k^2$, the new posterior mean and covariance can be computed as $P(\frac{\epsilon}{\tilde{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2})\mathbf{x} + \mathbf{q}(\frac{\epsilon}{\tilde{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2})$ and $\epsilon P(\frac{\epsilon}{\tilde{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2})$, respectively, where $P, \mathbf{q}$ are obtained by solving the Riccati ODE (3.11) (with index $j = k$) forward from $s = 0$ to $s = \frac{\epsilon}{\tilde{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2}$ with initial condition $P(0) = \frac{\Sigma}{\epsilon}$ and $\mathbf{q}(0) = \mu - \frac{1}{\epsilon}\Sigma\mathbf{x}$, where we denote by μ, Σ the original posterior mean and covariance computed using the original value of σ_k^2 . Note that every $s \in [0, \frac{\epsilon}{\tilde{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2}]$ can be written as $\frac{\epsilon}{\hat{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2}$ for some $\hat{\sigma}_k^2 \in [\tilde{\sigma}_k^2, \sigma_k^2]$. Hence, as we evolve the Riccati ODE from $s = 0$ to $s = \frac{\epsilon}{\tilde{\sigma}_k^2} - \frac{\epsilon}{\sigma_k^2}$, we simultaneously obtain a flow of solutions corresponding to each $\hat{\sigma}_k^2 \in [\tilde{\sigma}_k^2, \sigma_k^2]$, or, in other words, we tune the variance continuously from σ_k^2 to $\tilde{\sigma}_k^2$.

Similarly, if we increase the hyperparameter from σ_k^2 to $\tilde{\sigma}_k^2$, the new posterior mean and covariance can be computed as $P(0)\mathbf{x} + \mathbf{q}(0)$ and $\epsilon P(0)$, respectively, where $P, \mathbf{q}$ are obtained by solving the Riccati ODE (3.11) (with index $j = k$) backward from $s = \frac{\epsilon}{\sigma_k^2} - \frac{\epsilon}{\tilde{\sigma}_k^2}$ to $s = 0$ with terminal condition $P(\frac{\epsilon}{\sigma_k^2} - \frac{\epsilon}{\tilde{\sigma}_k^2}) = \frac{\Sigma}{\epsilon}$ and $\mathbf{q}(\frac{\epsilon}{\sigma_k^2} - \frac{\epsilon}{\tilde{\sigma}_k^2}) = \mu - \frac{1}{\epsilon}\Sigma\mathbf{x}$. As in the previous case, evolving the Riccati ODE from $s = \frac{\epsilon}{\sigma_k^2} - \frac{\epsilon}{\tilde{\sigma}_k^2}$ to $s = 0$ again corresponds to continuously tuning the variance from σ_k^2 to $\tilde{\sigma}_k^2$, and hence, every value of $P(s)\mathbf{x} + \mathbf{q}(s)$ and $\epsilon P(s)$, $s \in [0, \frac{\epsilon}{\sigma_k^2} - \frac{\epsilon}{\tilde{\sigma}_k^2}]$ corresponds to the posterior mean and covariance using a different variance for the k th data point.

3.3.3 Updating the prior: tuning hyperparameters

In this section, we discuss how our Riccati-based approach can be used to update the mean and covariance of the Gaussian prior. Tuning these hyperparameters may improve the inference accuracy of the learned model by updating the prior as new information becomes available.

Changing the prior mean $\boldsymbol{\theta}^0$ simply requires evaluating the solution to the corresponding HJ PDE at a different point $\mathbf{x} = \Lambda^{-1}\boldsymbol{\theta}^0$ in space, which, in turn, only requires the posterior mean to be updated with the new $\mathbf{x}$ -value using some matrix-vector multiplications and addition, while the posterior covariance remains unchanged. Hence, tuning the prior mean is relatively cheap computationally.

Consider changing the prior covariance from $\epsilon\Lambda$ to $\epsilon\tilde{\Lambda}$. Originally, the prior covariance appears in the initial condition of the sequence of Riccati ODEs (3.11). However, we do not have to re-solve the entire sequence of Riccati ODEs in order to update the prior covariance. Instead, we reinterpret the prior covariance as part of a Hamiltonian of the corresponding multi-time HJ PDE, which allows it to be tuned using only the results of training with the original prior. Denote the original posterior mean and covariance (computed using the the original prior) by μ_{pos} and Σ_{pos} , respectively. Then, the new posterior mean $\tilde{\mu}_{pos}$ and covariance $\tilde{\Sigma}_{pos}$ using the new prior covariance $\epsilon\tilde{\Lambda}$ can be obtained using Riccati ODEs as follows. First, solve the Riccati ODE (3.11) forward from $s = 0$ to $s = 1$ with $\Phi_j = \tilde{\Lambda}^{-1/2}$, $\mathbf{y}_j = 0$, and initial conditions $P(0) = \frac{\Sigma_{pos}}{\epsilon}$ and $\mathbf{q}(0) = \mu_{pos} - \frac{1}{\epsilon}\Sigma_{pos}\mathbf{x}$. This step incorporates information about the new prior covariance into the model and yields a solution $P(1)$ and $\mathbf{q}(1)$. Next, solve the Riccati ODE (3.11) backward from $s = 1$ to $s = 0$ with $\Phi_j = \Lambda^{-1/2}$, $\mathbf{y}_j = 0$, and terminal conditions $\tilde{P}(1) = P(1)$ and $\tilde{\mathbf{q}}(1) = \mathbf{q}(1)$, where $P(1), \mathbf{q}(1)$ are the solutions from the previous step. This step removes information

about the old prior covariance from the model and yields the solution $\tilde{P}(0)$ and $\tilde{\mathbf{q}}(0)$. Finally, the new posterior mean and covariance are computed as $\tilde{\mu}_{pos} = \tilde{P}(0)\mathbf{x} + \tilde{\mathbf{q}}(0)$ and $\tilde{\Sigma}_{pos} = \epsilon\tilde{P}(0)$, respectively.

When the new prior covariance $\epsilon\tilde{\Lambda}$ is a scaling of the original prior covariance $\epsilon\Lambda$ (i.e., $\tilde{\Lambda} = \alpha\Lambda$ for some $\alpha > 0$), the posterior can be updated by solving only one Riccati ODE instead of two. Again denote the original posterior mean and covariance computed using the original prior by μ_{pos} and Σ_{pos} . If $\alpha > 1$, then the new posterior mean and covariance can be computed as $P(\frac{1}{\alpha}) + \mathbf{q}(\frac{1}{\alpha})\mathbf{x}$ and $\epsilon P(\frac{1}{\alpha})$, respectively, where $P, \mathbf{q}$ are obtained by solving the Riccati ODE (3.11) backward from $s = 1$ to $s = \frac{1}{\alpha}$ with $\Phi_j = \Lambda^{-1/2}$, $\mathbf{y}_j = 0$, and terminal conditions $P(1) = \frac{\Sigma_{pos}}{\epsilon}$ and $\mathbf{q}(1) = \mu_{pos} - \frac{1}{\epsilon}\Sigma_{pos}\mathbf{x}$. Similarly, if $\alpha \in (0, 1)$, then the new posterior mean and covariance can be computed as $P(\frac{1}{\alpha}) + \mathbf{q}(\frac{1}{\alpha})\mathbf{x}$ and $\epsilon P(\frac{1}{\alpha})$, respectively, where $P, \mathbf{q}$ are obtained by solving the Riccati ODE (3.11) forward from $s = 1$ to $s = \frac{1}{\alpha}$ with $\Phi_j = \Lambda^{-1/2}$, $\mathbf{y}_j = 0$, and initial conditions $P(1) = \frac{\Sigma_{pos}}{\epsilon}$ and $\mathbf{q}(1) = \mu_{pos} - \frac{1}{\epsilon}\Sigma_{pos}\mathbf{x}$. In both cases, evolving the Riccati ODE yields a continuous flow of solutions corresponding to tuning the hyperparameter α continuously; namely, every s between 1 and $1/\alpha$ corresponds to another solution with a different prior covariance $\frac{\epsilon}{s}\Lambda$.

3.4 Examples in scientific machine learning

In this section, we apply our methodology to three examples from SciML, in which ML tools are employed to solve ODEs or PDEs [244, 142]. In each example, we learn

the solution $u : \mathbb{R}^\ell \rightarrow \mathbb{R}^m$ and the source term $f : \mathbb{R}^\ell \rightarrow \mathbb{R}^m$ of a differential equation

$$\begin{cases} \mathcal{F}[u] = f & \text{in } \Omega, \\ \mathcal{B}[u] = b & \text{on } \partial\Omega, \end{cases} \quad (3.16)$$

using noisy data and quantify corresponding epistemic uncertainties [131, 234, 348]. In the differential equation above, $\mathcal{F}$ and $\mathcal{B}$ represent linear differential and boundary operators, respectively. Following the framework in Section 3.2.1, we estimate u using the linear model $u_\theta = \Phi(\cdot)\theta$, where $\Phi : \mathbb{R}^\ell \rightarrow \mathbb{R}^{m \times n}$ is the matrix whose columns are the basis functions $\phi_i : \mathbb{R}^\ell \rightarrow \mathbb{R}^m$, $i = 1, \dots, n$. As a result, we also estimate f using a linear model $f_\theta = \mathcal{F}[u_\theta](\cdot) = \mathcal{F}[\Phi](\cdot)\theta$.

We use the predicted mean associated with our learned models as our prediction of the quantities of interest; i.e., our prediction for u is $\mathbb{E}[u_\theta|\mathcal{D}] = \Phi\mathbb{E}[\theta|\mathcal{D}]$ and our prediction for f is $\mathbb{E}[f_\theta|\mathcal{D}] = \mathcal{F}[\Phi]\mathbb{E}[\theta|\mathcal{D}]$. Note that considering the predicted mean of our learned models is equivalent to taking θ to be the posterior mean estimate. We compute the predicted uncertainty of our learned models as their covariances: $\text{Cov}[u_\theta|\mathcal{D}] = \Phi\text{Cov}[\theta|\mathcal{D}]\Phi^T$ and $\text{Cov}[f_\theta|\mathcal{D}] = \mathcal{F}[\Phi]\text{Cov}[\theta|\mathcal{D}]\mathcal{F}[\Phi]^T$. The predicted uncertainty can then be used to help inform the learning process (e.g., by providing a metric for determining when to stop learning or where more data may be needed). When $m = 1$, the covariance is scalar-valued and we can take twice its square root (i.e., twice the posterior standard deviation) as the predicted uncertainty. In each example, we assume a Gaussian likelihood and Gaussian prior and apply the Riccati-based methodology from Section 3.3 to compute the posterior mean and covariance. For demonstration purposes, we use the 4th-order Runge-Kutta (RK4) method to numerically solve the corresponding Riccati ODEs, although any other appropriate numerical method could be used instead. Details for the choice of hyperparameters in each example can be found in Appendix B.1. Code for all of the examples is

publicly available at <https://github.com/ZongrenZou/HJPDE4UQSciML>.

3.4.1 Solving a boundary-value ODE problem

Consider the following boundary value problem:

$$\begin{cases} \kappa \frac{d^4 u}{dt^4}(\tau) + \beta \frac{d^2 u}{dt^2}(\tau) + u(\tau) = f(\tau), \tau \in (0, T), \\ u(0) = u_0, u(T) = u_T, u'(0) = u'_0, u'(T) = u'_T, \end{cases} \quad (3.17)$$

where $\kappa = 0.0001, \beta = 0.01$ are known constants. In this example, we solve (3.17) using noisy measurements of u_0, u_T, u'_0, u'_T and noisy measurements of the source term f at different times τ , which we denote by $\{(\tau_i, f_i)\}_{i=1}^{N_f}$. To demonstrate the capabilities and potential computational advantages of our Riccati-based approach, we consider the following three learning scenarios:

- A. Continual learning: stream data of f sequentially in τ (Section 3.4.1.1),
- B. Hyperparameter tuning: continuously tune the standard deviation of the prior (Section 3.4.1.2),
- C. Outlier removal: remove excessively noisy data from the learned model (Appendix B.2).

In each scenario, we use the truncated Karhunen-Loève (KL) expansion of a Gaussian process as our linear model. Specifically, we use the leading $n = 30$ terms of the KL expansion:

$$u_{\boldsymbol{\theta}}(\tau) = \sum_{k=1}^n \theta_k \sqrt{\alpha_k} \tilde{\phi}_k(\tau), \quad (3.18)$$

where $\tilde{\phi}_k, \alpha_k$ are the eigenfunctions and eigenvalues of the kernel

$$k(x_1, x_2) = \exp\left(-\frac{|x_1 - x_2|}{0.05}\right), \quad \forall x_1, x_2 \in [-10, 10]. \quad (3.19)$$

In other words, we use the basis functions $\{\phi_k(\tau) = \sqrt{\alpha_k}\tilde{\phi}_k(\tau), k = 1, \dots, n\}$ [315]. For more details, the analytic expansion can be found in [98, 312]. Unless otherwise specified, we also assume that the prior on $\boldsymbol{\theta}$ is standard independent Gaussian, i.e., $\boldsymbol{\theta} \sim N(0, I)$ where I is the $n \times n$ identity matrix. The exact solution is assumed to be $u(\tau) = \exp(-2\tau)\sin(15\tau)$ (but treated as unknown a priori), and we set $T = 1$. The data of u_0, u_T (u'_0, u'_T , respectively) are corrupted by additive Gaussian noise with mean zero and standard deviation 0.01 (0.001, respectively).

3.4.1.1 Case A: continual learning

	After the 101th data point	After the 151th data point	After the 201th data point
u	41.77%	6.79%	3.88%
f	217.27%	24.48%	3.39%

Table 3.1: Relative L_2 errors of the predicted means of u and f when using continual learning and our Riccati-based approach to solve (3.17) after different amounts of noisy measurements of f are incorporated into the learned models. We achieve high accuracy inferences after all the data is incorporated.

In this scenario, we learn u and f using continual learning [222, 288] to demonstrate the potential of our Riccati-based approach for real-time inferences. Continual learning refers to learning scenarios in which data is accessed in a stream and learned models are updated incrementally as new data becomes available. In some cases, the historical data is also assumed to become inaccessible after it is incorporated into the learned model, which often leads to catastrophic forgetting ([155, 222]), i.e., the abrupt degradation in the performance of learned models on previous tasks upon training on new tasks. Here, we assume that measurements

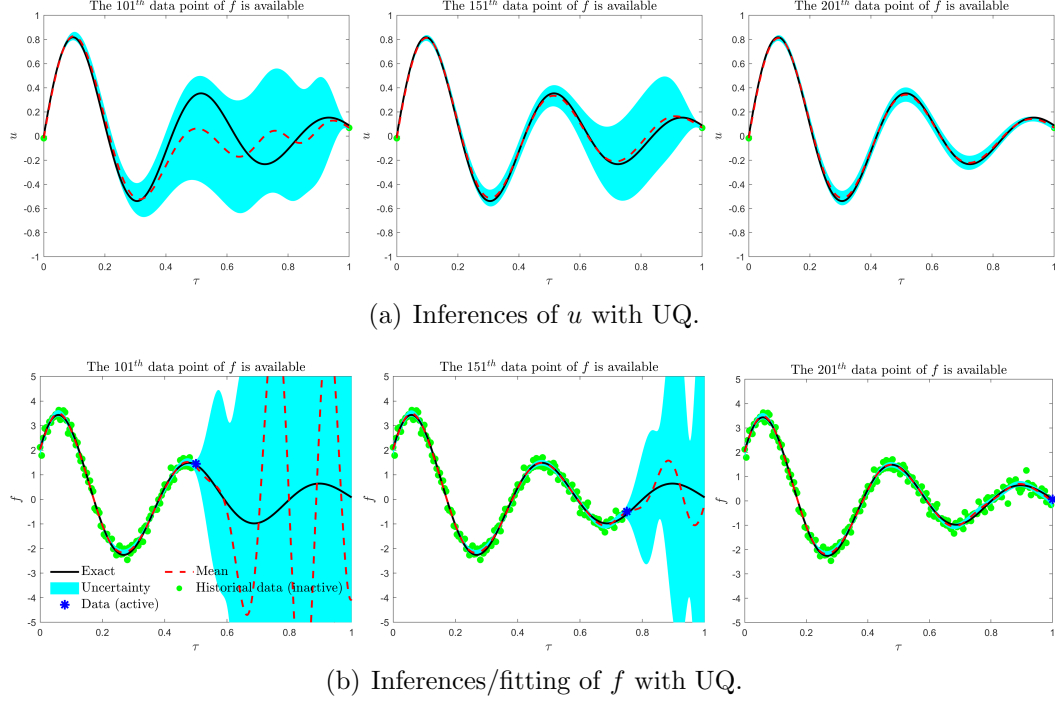


Figure 3.4: Results of solving (3.17) using continual learning and our Riccati-based approach. (a) and (b) show the predicted mean and uncertainty of u and f , respectively, after the 101th, 151th, and 201th noisy data point of f becomes available. Our Riccati-based approach naturally coincides with the continual learning framework while inherently avoiding catastrophic forgetting. Our approach allows us to incrementally update the learned models without accessing the historical data ($\bullet$), while also providing a quantitative metric for our confidence in the learned models in the form of the predicted uncertainties ($\blacksquare$). We observe that regions of low predicted uncertainty generally coincide with regions of high inference accuracy, which implies that this confidence metric is a good indicator of the reliability of the model.

of f are streamed sequentially in τ (assume that one measurement of f becomes available every $\Delta\tau = \frac{T}{200}$) and that the previous data cannot be accessed again after a new measurement arrives. As such, we want to update our learned models as soon as new data becomes available. We also assume that the measurements of f are corrupted by additive Gaussian noise with mean zero and standard deviation 0.2, so that a Bayesian approach may be useful. We apply the Riccati-based methodology from Section 3.3.2.1, which is naturally amenable to data streaming. In particular, this Riccati-based approach provides computational advantages over more standard SciML and UQ techniques in continual learning settings by using only information

of the newly available data to update the learned model instead of retraining on the entire dataset. These computational advantages become more significant in long-term integration problems, in which retraining on and storage of the historical data becomes considerably more expensive. Furthermore, although our Riccati-based approach does not rely on historical data, it does not suffer from catastrophic forgetting as all of the information from the previous data remains intrinsically encoded in the solution to the corresponding HJ PDE.

Figure 3.4 displays the predicted means and uncertainties of u and f after the 101th, 151th, and 201th noisy measurement of f becomes available. We see that as more data points of f are incorporated into the learned models, the predicted means of both u and f more closely match their exact values and the predicted uncertainties shrink. Moreover, we see that regions of high predicted uncertainty generally coincide with low inference accuracy, which shows that our confidence in the learned models is correlated with their reliability. We also observe that the predicted uncertainty of u does not develop in the same way as that of f . While the predicted uncertainty of f is generally uniformly small on $(0, \tau^*)$ and large on $[\tau^*, T)$, where τ^* denotes the τ -coordinate of the currently available data point, u can often still have relatively large predicted uncertainty on $(0, \tau^*)$ since we do not learn u directly. Note that u always has low predicted uncertainty on the boundaries as the boundary data were incorporated into the learned models at the beginning of the training. Table 3.1 presents the relative L_2 errors of the predicted means, which show that we achieve high inference accuracy after all the data has been incorporated. We compute the L_2 errors using trapezoidal rule with a uniform grid of size 1001 over the whole domain $[0, T]$.

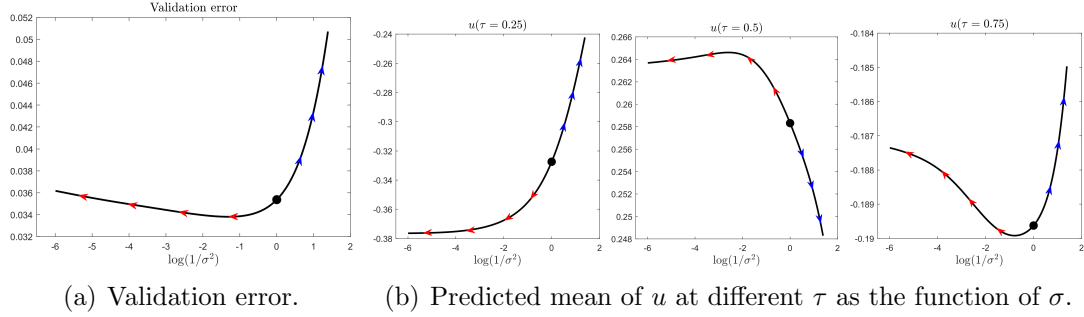


Figure 3.5: Results of tuning the standard deviation σ of the prior using our Riccati-based approach to solve (3.17). (a) shows how the validation error changes as σ is tuned. (b) shows the flow with respect to σ of the predicted mean of u at different times ($\tau = 0.25, 0.5, 0.75$). The black dot represents the value using the original choice of $\sigma = 1$, and the arrows represent the direction of the flow of solutions as we increase σ from 1 (**red**; i.e., we evolve the corresponding Riccati ODE backward) or decrease σ from 1 (**blue**; i.e., we evolve the corresponding Riccati ODE forward). In contrast to conventional ML approaches, our Riccati-based approach allows the hyperparameter to be tuned continuously without retraining on or access to the training data.

3.4.1.2 Case B: tuning the standard deviation of the prior

In the previous case, the prior distribution was set to be independent standard Gaussian, i.e., $\boldsymbol{\theta} \sim N(0, \sigma^2 I)$, which is the usual choice for KL expansions of Gaussian processes [306]. However, this choice may not be appropriate for every problem, and tuning the hyperparameters of the prior may yield better results. In this section, we demonstrate how the Riccati-based approach from Section 3.3.3 can be used to tune the standard deviation σ of the prior continuously. In this case, the training data consists of the same measurements of u and u' used in Case A and 41 measurements of f sampled equidistantly on $[0, T]$. The validation data consists of 10 measurements of f sampled randomly on $[0, T]$ following a uniform distribution. Both the training and validation data are corrupted by additive Gaussian noise with mean zero and standard deviation 0.2.

Figure 3.5 shows how the validation error and predicted mean of u develop as σ

is tuned. Figure 3.5(b) shows that the predicted mean of u varies as σ changes, and Figure 3.5(a) shows that the lowest validation error is not achieved with the original choice of $\sigma = 1$, which highlights the importance of hyperparameter tuning. Recall that tuning σ corresponds to evolving a Riccati ODE in time. Hence, tuning σ can be done continuously and results in a continuous flow of solutions, where each solution corresponds to a different choice of σ . This flow of solutions is obtained numerically via the intermediary steps of RK4 and is represented by the arrows in Figure 3.5. In Figure 3.5(a), tuning σ results in a continuous flow of validation errors, and in Figure 3.5(b), it results in a continuous flow of inferences of u . Our Riccati-based approach allows hyperparameter values to be explored flexibly and continuously and, in contrast to more standard ML techniques, does not require retraining on or access to the training dataset during the tuning process.

3.4.2 Solving the 1D steady-state advection-diffusion equation

Consider the following steady-state advection-diffusion equation with Dirichlet boundary conditions:

$$\begin{cases} D \frac{\partial^2 u}{\partial x^2}(x) + \kappa \frac{\partial u}{\partial x}(x) = f(x), x \in (0, 1), \\ u(0) = u(1) = 0, \end{cases} \quad (3.20)$$

where $D = 0.001$ and $\kappa = 1$. We solve (3.20) using noisy measurements of f at different x , which we denote by $\{(x_i, f_i)\}_{i=1}^N$. To demonstrate the versatility of our Riccati-based approach in handling different learning settings, we consider the following two cases:

- A. The data of f are corrupted by large-scale noise but are cheap to obtain (Section 3.4.2.1),
- B. The data of f are corrupted by small-scale noise but are expensive to obtain (Section 3.4.2.2).

Case A uses a large-scale problem with a huge amount of data to show that we can achieve high accuracy despite a large amount of noise, while Case B uses active learning [248] to search for where new measurements are needed. In both cases, we use the truncated KL expansion of the Brownian bridge to model u . Specifically, we use the leading $n = 50$ terms of the KL expansion [98]:

$$u_{\boldsymbol{\theta}}(x) = \sum_{k=1}^n \theta_k \frac{\sqrt{2} \sin(k\pi x)}{k\pi}. \quad (3.21)$$

Note that this choice of basis functions automatically enforces the Dirichlet boundary conditions. We take the prior to be independent standard Gaussian, i.e., $\boldsymbol{\theta} \sim N(0, I)$.

3.4.2.1 Case A: large-scale data with high noise level

	1,000 data points	5,000 data points	100,000 data points
Error of u	44.44%	17.19%	9.49%
Error of f	41.89%	16.72%	3.81%

Table 3.2: Relative L_2 errors of the predicted means of u and f using different amounts of noisy measurements of f to solve (3.20). Our Riccati-based approach updates the learned models without requiring retraining when new data becomes available or storing and processing large datasets in their entirety.

In this case, we demonstrate the computational efficiency of our Riccati-based approach on a large-scale problem. Specifically, we assume that measurements of f are corrupted by a large amount of noise but are cheap to acquire. Thus, to obtain accurate predictions and compensate for the high noise level, we need a lot of

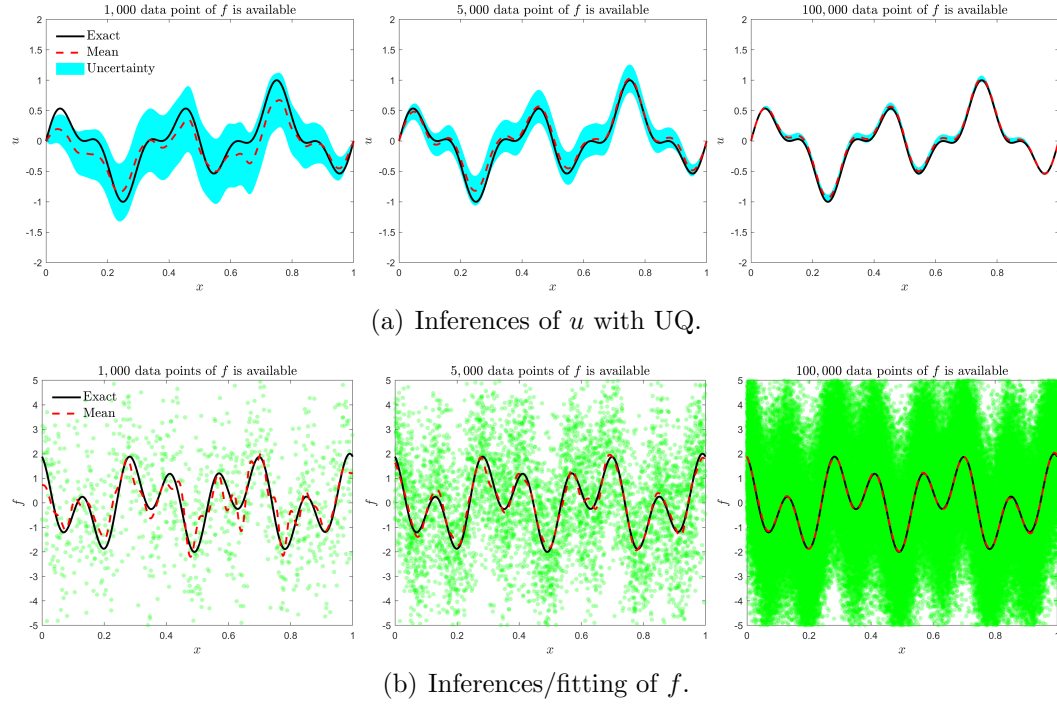


Figure 3.6: Results of solving (3.20) using large-scale data with high noise level and our Riccati-based approach. (a) shows the corresponding predicted mean and uncertainty (■) of u . (b) shows the inference/fitting of f using 1,000, 5,000, and 100,000 noisy data points. Note that we do not show the predicted uncertainty of f for clarity of presentation. Our Riccati-based approach updates the learned models incrementally using information from one data point ($\cdot$) at a time, which provides significant computational and memory savings in big data regimes.

data. Working with such a large dataset in its entirety would be extremely expensive in terms of both computations and memory. Instead, we apply our Riccati-based approach from Section 3.3.2.1 to continuously learn our model without have to train on or store the entire dataset. In this case, the measurements of f are sampled uniformly randomly on $[0, 1]$ and corrupted by additive Gaussian noise with mean zero and standard deviation 2 (e.g., see in Figure 3.6(b)). We keep sampling f until a satisfactory result is obtained (i.e., until the predicted uncertainty is sufficiently low).

Figure 3.6 shows inferences of u and f after 1,000, 5,000, and 100,000 data points of f are incorporated into the learned models. Note that the predicted uncertainty

at the boundary points is always zero because our choice of basis functions automatically enforces the boundary conditions. Due to the high noise level, the inferences of u and f are inaccurate when only 1,000 data points are used. Additionally, the predicted uncertainty of u is relatively large, which indicates that the learned models are low confidence. As we incorporate more measurements of f into the learning process, the inferences become more accurate and the predicted uncertainty of u decreases. Once the predicted uncertainty is small enough, we determine that we are confident in our learned model and can stop collecting data. For example, in Figure 3.6(a), we see that the predicted uncertainty of u is significantly and sufficiently reduced after using 100,000 data points, and we stop learning. In Table 3.2, we observe that the relative L_2 errors of our inferences are also significantly reduced, confirming that our confidence in the model is a reasonable indicator of its accuracy. The L_2 errors are computed using trapezoidal rule with a uniform grid of size 1001 over the domain $[0, 1]$. We again note that by using the Riccati-based methodology from Section 3.3.2.1, we avoid having to retrain every time we acquire new data points and never have to work with all 100,000 data points at once. Instead, our Riccati-based approach only requires information about one data point at a time, which provides significant computational and memory advantages in these big data regimes.

3.4.2.2 Case B: expensive data with low noise level

In this case, we demonstrate the versatility of our Riccati-based methodology by showing how it naturally complements an active learning approach. Active learning refers to the ML paradigm where the training data are sampled dynamically until a desired result is achieved (e.g., low generalization error) [265, 248, 183]. In this section, we focus on uncertainty-based active learning [183, 108, 230], in which we

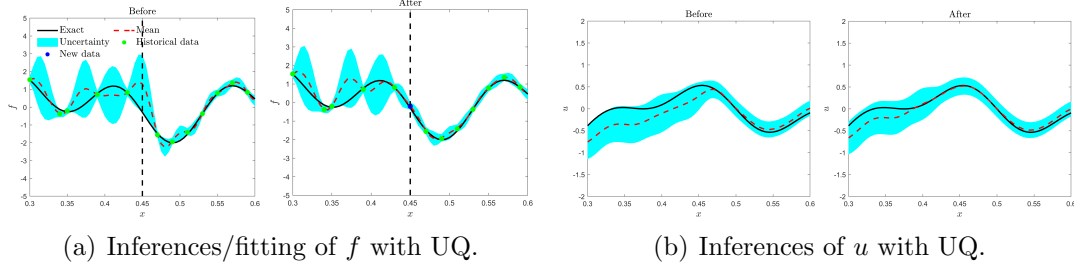


Figure 3.7: Intermediate result for adding the 46th data point when using active learning and our Riccati-based approach to solve (3.20). The **left** side of (a) and (b) shows the inferences of f and u , respectively, before the placement of the new sensor and incorporation of the 46th data point. The **right** side of (a) and (b) shows the updates to f and u after these actions. The vertical black dashed line represents the location where the 46th sensor will be placed. Note that we zoom in on the region around the new sensor instead of showing the full domain for clarity of comparison. Following an active learning framework, we repeat the following steps: compute the predicted mean and uncertainty of u and f , place a new sensor at the location of the largest predicted uncertainty (■) of f , and obtain a new measurement of f (*) at the new sensor location. Our Riccati-based approach naturally complements this framework as it can efficiently perform repeated updates of the predicted mean and uncertainty.

choose the new data point to be sampled from the location where the predicted uncertainty is largest. Assume that we have access to high quality data with low-level noise (here, we use additive Gaussian noise with mean zero and standard deviation 0.1) but that the data are expensive to collect. For example, consider the scenario where high precision but expensive sensors are deployed to measure the value of f at particular locations x . As a result, we have limited access to data and must carefully select where sensors should be placed in order to obtain reliable inferences. Assume the potential locations for sensors form a 101-point uniform grid on $[0, 1]$. We use the predicted uncertainty of f to determine where the next sensor should be placed. Note that although we could have instead used the predicted uncertainty of u to choose the sensor location, using the predicted uncertainty of f is more consistent with the fact that we collect measurements of f . Following the active learning framework [248, 236, 230], we start with an empty dataset $\mathcal{D}$ and index $i = 0$ and then do the following:

1. Compute the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ based on the current dataset $\mathcal{D}$ as well as the predicted mean and uncertainty of u and f at all potential sensor locations. If the predicted uncertainty is sufficiently low, then terminate the procedure.
2. Identify the location where the predicted uncertainty of f is highest out of all remaining potential locations. Label this location as x_i , and place a sensor there to measure f (denote these measurements by f_i).
3. Update the current dataset $\mathcal{D}$ to $\mathcal{D} \cup \{(x_i, f_i)\}$ and the index i to $i + 1$. Go back to step 1.

To perform steps 1 and 2, we employ the Riccati-based approach from Section 3.3.2.1, in which the models and their predicted uncertainty are updated incrementally as new data becomes available. Note that the Riccati-based approach performs these updates without using historical data.

Figure 3.7 displays a zoom in of the intermediate result when adding the 46th measurement of f . The vertical black dashed line indicates where the predicted uncertainty of f is the highest (i.e., where the new sensor is placed). While the addition of the sensor significantly reduces the predicted uncertainty of f around that point, it only mildly reduces the predicted uncertainty of u since we do not learn u directly. Figure B.2 in Appendix B.3 displays intermediate results when adding the 47th sensor, which more dramatically reduces the predicted uncertainty of u . These results demonstrate how our Riccati-based approach can be seamlessly integrated into an active learning framework by allowing for continual updates of the models and uncertainty metrics that can be leveraged to interactively improve the learning process.

3.4.3 Solving the 2D Helmholtz equation

In this example, we solve the 2D Helmholtz equation with Dirichlet boundary conditions:

$$\begin{cases} (\kappa^2 - \Delta)u(x, y) = f(x, y), & x, y \in [0, 2\pi], \\ u(x, 0) = u(x, 2\pi) = u(0, y) = u(2\pi, y) = 0, & x, y \in [0, 2\pi], \end{cases} \quad (3.22)$$

where $\kappa^2 = 1$ is the Helmholtz constant and Δ is the Laplacian operator. We consider the scenario where we have large-scale, noisy data of f but assume that computational limitations prevent us from being able to process or store all of the data at once. We use this example to show how our Riccati-based approach naturally overcomes these computational limitations. Specifically, we decompose the domain into multiple smaller subdomains and update our learned model on each piece one-by-one using continual learning and the Riccati-based methodology from Section 3.3.2. Note that since the Riccati-based approach is invariant to the order of the data points, it does not require a specific method for decomposing the domain. To illustrate the flexibility of our approach, we decompose the domain $[0, 2\pi]^2$ uniformly into 7×7 equal subdomains and consider the following two patterns for traversing the subdomains:

- A. A sequential pattern (Section 3.4.3.1; e.g., see Figure 3.8(a)),
- B. A multi-level order of traversal (Section 3.4.3.2; e.g., see Figure 3.10).

In both cases, we employ a linear model $u_{\theta}(x, y) = \sum_{k=1}^n \theta_k \phi_k(x, y)$, where $\{(x, y) \mapsto \phi_k(x, y)\}_{k=1}^n = \left\{ (x, y) \mapsto \frac{\sqrt{2L} \sin(\frac{j\pi x}{L})}{j\pi} \frac{\sqrt{2L} \sin(\frac{k\pi y}{L})}{k\pi} \right\}_{j,k=1}^{\sqrt{n}}$, $n = 225$, and $L = 2\pi$. Note that these basis functions automatically enforce the boundary conditions. We learn

the model using measurements of f that are corrupted by additive Gaussian noise with mean 0 and standard deviation 0.5. We define f by

$$f(x, y) = \sin(6x) \sin(4y) - 0.8 \sin(5x) \sin(7y), \quad (3.23)$$

and the solution to (3.22) can be analytically derived accordingly. To compute the accuracy of learned models, we evaluate our inferences of u and f on a uniform grid of size 450×450 over the domain. To learn θ , we take noisy measurements of f at each of the 200,704 interior points of that grid. Hence, each subdomain contains 4,096 measurements of f .

3.4.3.1 Case A: sequential domain decomposition

	25 subdomains	40 subdomains	44 subdomains	49 subdomains (all)
Error of u	897.75%	231.56%	93.19%	9.73%
Error of f	178.51%	62.74%	20.38%	2.32%

Table 3.3: Relative L_2 errors of the predicted means of u and f when solving (3.22) using our Riccati-based approach and the sequential order of integration shown in Figure 3.8(a). The errors significantly reduce as more subdomains are visited but remain slightly high after all of the data is incorporated due to the high level of noise in the data.

In this case, we traverse the subdomains sequentially using the snaking pattern shown in Figure 3.8(a). This decomposition pattern naturally extends the sequential data streaming pattern used for the 1D problem in Section 3.4.2.1 to the 2D problem considered here. As such, this traversal pattern intuitively coincides with the Riccati-based methodology for adding data points in Section 3.3.2.1.

In Figure 3.8, we see that we learn u and f fairly well, but the accuracy of the predicted means of u and f remains compromised by the large amount of noise in the data. In particular, Figures 3.8(c)-(d) show that the exact values are still within

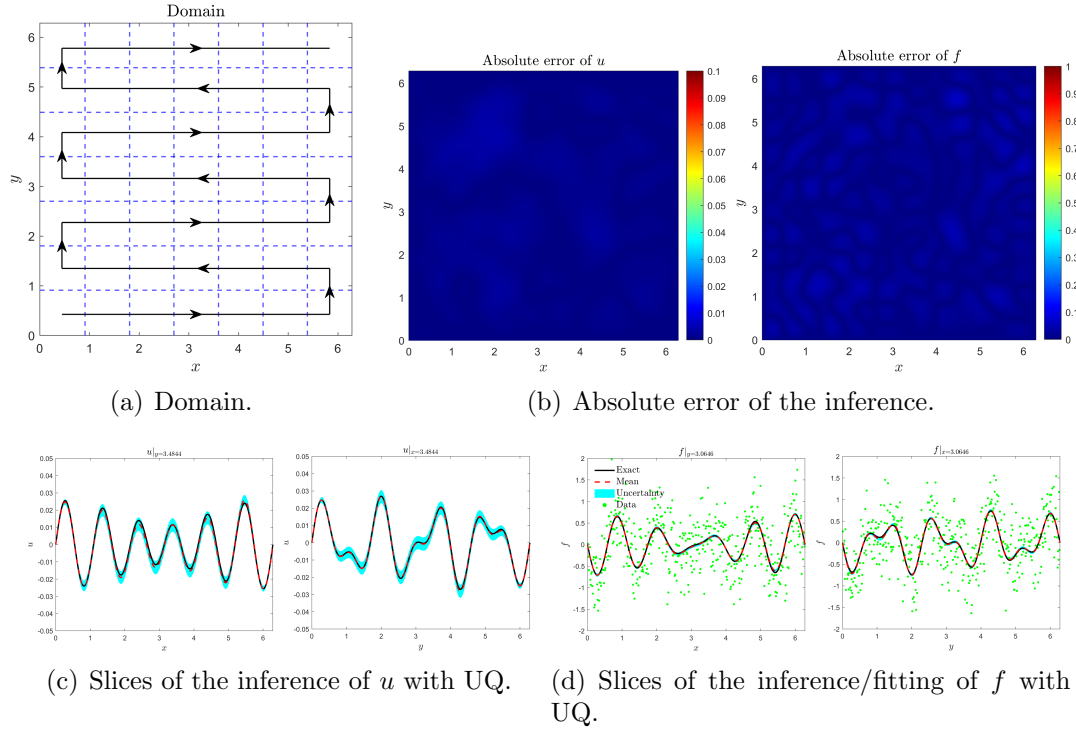


Figure 3.8: Results of solving (3.22) with noisy data of f using our Riccati-based approach while traversing the domain sequentially. (a) displays the 7×7 uniform domain decomposition and the sequential order of integration of the domain. (b) shows the absolute errors of the predicted means of u and f after all 49 subdomains have been visited. (c) and (d) show 1D slices of the inferences and predicted uncertainties (■) of u and f , respectively. These results show how our Riccati-based approach can naturally be extended to solve higher-dimensional problems. Namely, by leveraging data streaming, our Riccati-based approach is able to overcome the computational and memory challenges related to the higher dimension of the PDE and subsequently increased dataset size.

the predicted uncertainty intervals around our predicted means, which indicates that our inferences are still reliable. In Figure 3.9 and Table 3.3, we present intermediate results for the errors of our inferences after the domain has been partially traversed. In Figure 3.9, we observe that the absolute error of the inference of f vanishes on each of the visited subdomains, whereas the absolute error of the inference of u is reduced on the visited subdomains but not necessarily eliminated. This behavior is due to the fact that we do not learn u directly and instead learn u from data of f and information from the PDE. In Table 3.3, we again see that the errors

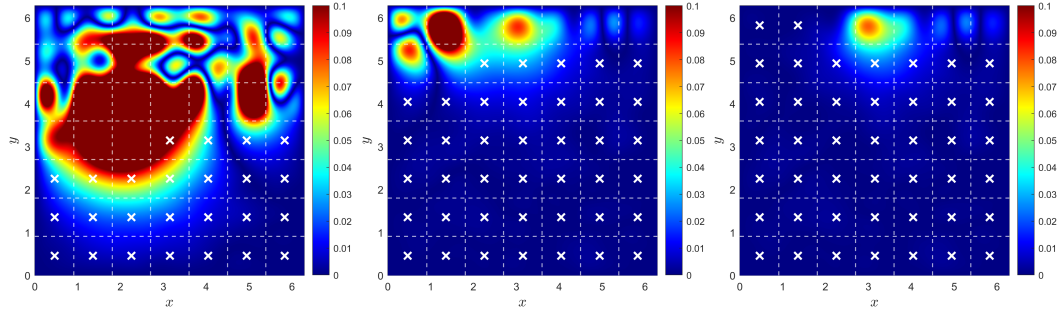
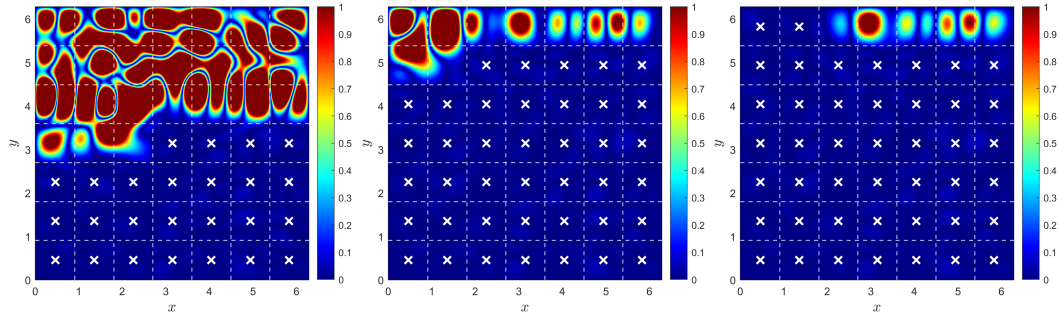
(a) Absolute error of the inference of u .(b) Absolute error of the inference/fitting of f .

Figure 3.9: Intermediate results of solving (3.22) using our Riccati-based approach while traversing the domain sequentially. We collect data from each subdomain sequentially following the pattern in Figure 3.8(a). (a) and (b) shows the absolute errors of the inference of u and f , respectively. The white crosses indicate the subdomains that have already been visited. Our Riccati-based approach incrementally updates our predictions as we traverse the domain, which provides computational and memory advantages over having to process or store all of the data from the entire domain at once.

significantly reduce as we traverse domain, but the final error of our inference of u remains somewhat high due to the high noise level of the data. Since our Riccati-based approach incrementally updates our inferences with one data point at a time, it does not suffer from any computational or memory scaling issues related to the large dataset size but does still depend on the numerical precision of the ODE solver.

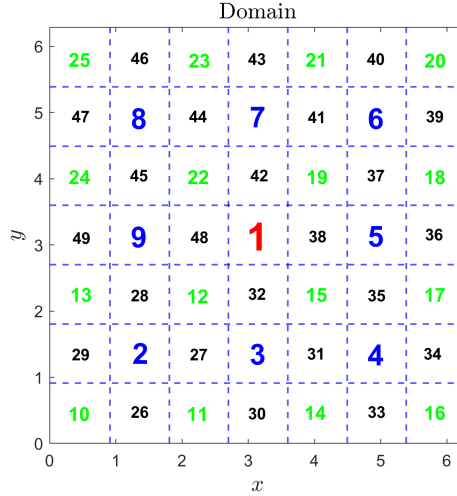


Figure 3.10: Domain decomposition with multi-level subdomain traversal. The level one subdomain is marked in **red**, the level two subdomains are marked in **blue** numbers, the level three subdomains are marked in **green** numbers, and the level four subdomains are marked in **black**.

3.4.3.2 Case B: multi-level domain decomposition

In this section, we follow a multi-level order of subdomain traversal inspired by multi-grid methods [113, 192] to highlight the invariance of our Riccati-based approach to the order of the data. In the previous case, we incorporated the data sequentially. However, this order is not required, as our Riccati-based approach can incrementally incorporate the data into the learned models in any order. Specifically, we consider the multi-level traversal order shown in Figure 3.10. Level one consists of the center subdomain. Level two consists of the eight subdomains roughly centered around the level one subdomain. Level three consists of 16 subdomains, each corresponding to one corner of each of the level two subdomains. Level four consists of the remaining 24 subdomains. Using this type of multi-level order has the potential to allow the learning process to stop earlier (and hence, use less data) because each level captures information across the entire domain. As a result, the error of the inferences is more likely to reduce uniformly across the domain. In contrast, the sequential traversal pattern used in Section 3.4.3.1 is more likely to learn accurate predictions on the

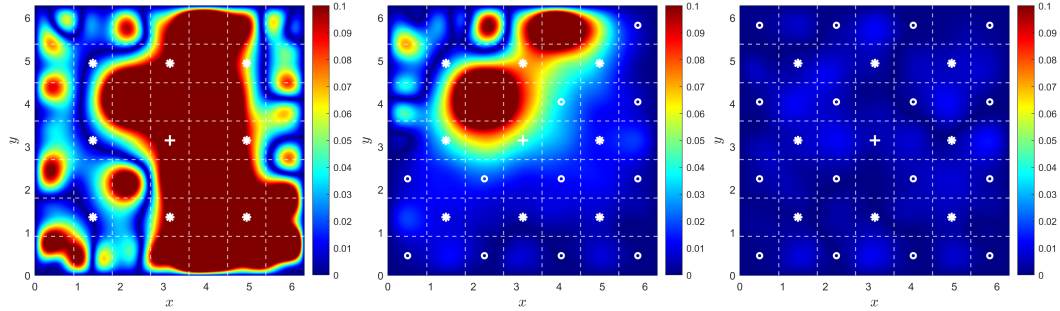
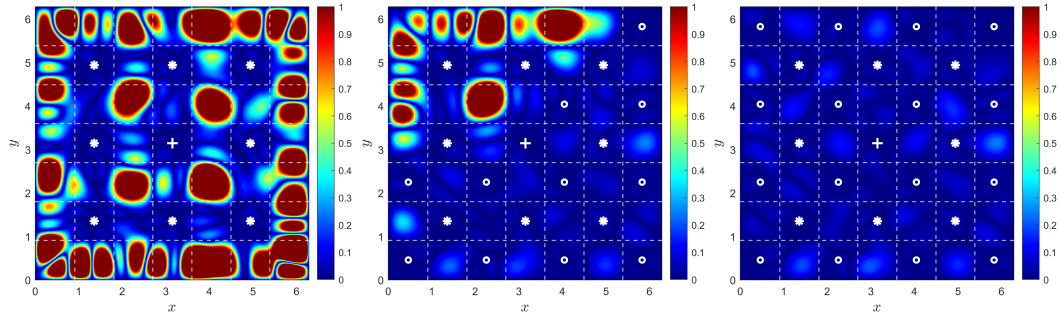
(a) Absolute error of the inference of u .(b) Absolute error of the inference/fitting of f .

Figure 3.11: Intermediate results of solving (3.22) using our Riccati-based approach while traversing the domain using the multi-level order in Figure 3.10. (a) and (b) show the absolute errors of the inference of u and f , respectively. The level one subdomain is marked by the white cross, the level two subdomains are marked by white stars, and the level three subdomains are marked by white circles. Our Riccati-based approach incrementally incorporates data points into the predictions invariant to the order of the data. As such, the end result after all the data is incorporated is identical to Figure 3.8(b)-(d), despite the intermediate results differing.

bottom half of the domain before learning any reasonably reliable prediction on the top half of the domain.

Figure 3.10 shows the intermediate results using this multi-level approach. As in the previous case, we see that the errors generally improve in/near the subdomains we have visited. The end result (not shown) after all of the subdomains have been visited is identical to Figure 3.8(b)-(d) with 9.73% relative L_2 error for u and 2.32% relative L_2 error for f , which demonstrates the invariance of our Riccati-based approach to the order of the data points.

3.5 Summary

In this chapter, we established a new theoretical connection between Bayesian inference problems with linear models and Gaussian likelihoods and viscous HJ PDEs with quadratic Hamiltonian. As a first exploration of this connection, we specialized to Gaussian priors and leveraged the result to develop a new Riccati-based methodology for efficiently updating the predicted models. We then demonstrated some potential computational advantages of this Riccati-based approach by applying it to several UQ-based examples from SciML [142, 234, 348]. In particular, these examples illustrate that this approach naturally coincides with data streaming applications [222, 288, 265, 248]. As such, this Riccati-based approach is amenable to memory-limited hardware architectures, such as FPGAs [146], which opens possibilities for real-time applications requiring embedded implementations.

Some other possible future directions for this work are as follows. Currently, our theoretical connection requires a linear model, Gaussian likelihood, and particular assumptions on the prior. In the Gaussian setting, it would be interesting to explore how incremental or online GPR techniques [59, 246, 130, 195, 54, 38, 213, 232, 257] could be integrated into our Riccati-based framework to approximate the solutions when the Riccati ODEs are difficult to solve directly using numerical ODE solvers, particularly in big data regimes. To consider a wider range of applications, more research is needed to extend this connection to nonlinear models, non-Gaussian likelihoods, and more general priors, including the development of new and efficient numerical algorithms for these settings in high dimensions. Specifically, the link between Bayesian inference and viscous HJ PDEs no longer holds when the Hamiltonian is non-quadratic. Moreover, the Riccati-based methodology in Section 3.3 further restricts the prior to be Gaussian. As such, more work is needed to ex-

plore how this connection could be leveraged to create efficient numerical methods for more general learning settings. Another natural extension of this work would be to investigate how this connection could be leveraged to reuse existing efficient ML algorithms to solve high-dimensional HJ PDEs. So far, this work only considers the opposite direction but opens opportunities for new ML-based solvers for fields related to HJ PDEs (e.g., stochastic optimal control [326, 81]).

CHAPTER FOUR

**Quantifying uncertainty arising
from noisy inputs-outputs**

4.1 Introduction

Recent years have witnessed great progress in scientific machine learning (SciML) in which the data and physics are combined seamlessly under modern machine learning (ML) frameworks to address complex scientific modeling, computation and model discovery [142]. As SciML emerges as a revolutionary paradigm, its uncertainty quantification (UQ) becomes increasingly critical for trustworthy and reliable predictions. Numerous approaches have been proposed for UQ in SciML. To name a few, Bayesian neural networks (BNNs) have been utilized to quantify uncertainty arising from noisy (outputs) data [315, 175], physical models [346] as well as operator learning [173, 206] in the context of solving ordinary/partial differential equations (ODEs/PDEs), Adversarial training has been applied in [321] for UQ in solving PDEs with physics-informed neural networks (PINNs). The deep ensemble method has been employed for solving ODEs/PDEs [234, 348, 334], inferring thermal boundary conditions from noisy measurements [270], as well as for the discovery and forecasting of extreme events [230]. The randomized prior method has found applications in operator learning [320]. Uncertainty stemming from solving stochastic differential equations was investigated in [329, 317, 11]. Further, researchers in [201, 344, 325, 309, 292, 137] employed the deep generative models to quantify the uncertainties in solving ODEs/PDEs and solving Bayesian inverse problems efficiently. A comprehensive review and a Python library for UQ in SciML can be found in [234] and [348], respectively.

In the present study, our specific focus is to explore the uncertainties in the predictions of two popular SciML models, e.g. the physics-informed neural networks (PINNs) [244] and neural operators (NOs) [178, 171]. The former encodes the physics in the form of differential equations via automatic differentiation, while

the latter is capable of learning the solution operator or hidden physics given training data. Generally, PINNs take as input the spatial-temporal coordinates and outputs functions that approximate the solutions to PDEs, while NOs take the boundary conditions and/or source terms, etc. as inputs and outputs functions, which approximate the solutions to differential equations. Despite the plurality of aforementioned UQ methods in SciML, we note that most of the existing works assume that the inputs, e.g. spatial-temporal coordinates in PINNs, are clean and only take into consideration the uncertainties associated with the noisy measurements of outputs. However, in real-world applications the measurements on both the coordinates and the corresponding responses may be noisy if they are measured from sensors, e.g., particle image velocimetry (PIV) [31, 35], estimating biomechanical properties of the aorta [325], hydrological modeling [147, 148]. As reported in [70, 82, 100, 310, 284, 289, 306, 13, 261, 99, 194, 68, 225], the measurement error in the inputs of models can present considerably greater challenges compared to noise in the outputs, primarily attributed to the inherent model nonlinearity. To the best of our knowledge, quantifying uncertainty originating from noisy inputs of PINNs as well as NOs has been largely ignored. Specifically, we describe the problem of interest as follows [70]:

$$\tilde{y} = \mathcal{H}(\chi) + \epsilon_o, \quad (4.1a)$$

$$\tilde{\chi} = \chi + \epsilon_{in}, \quad (4.1b)$$

where $\mathcal{H}$ denotes an NN surrogate model, χ denotes the input, and $\tilde{y}$ and $\tilde{\chi}$ denote the observed target value of the output and input of $\mathcal{H}$, respectively. We denote the measurement error or noise as ϵ_o and ϵ_{in} for the output and input, respectively, making the regression of $\chi, \mathcal{H}$ a noisy input-output problem.

Studies on uncertainty arising from both noisy inputs and outputs can be traced

back to the 1980s and 1990s within statistical research [70, 82, 100] and ML communities [310, 284, 289], respectively. Regression models developed to address this challenge are commonly referred to as *errors-in-variables* or *measurement error* models [306, 13, 261]. For example, Gaussian processes (GPs), which are a popular machine learning algorithm for regression problems, have been used in modeling uncertainty arising from noisy inputs as well as the outputs [99, 194]. However, the aforementioned work on UQ for noisy inputs-outputs has primarily focused on function approximation problems using neural networks (NNs) and GPs, without the involvement of physics information, e.g. ODEs/PDEs. More recently, the effect of noisy inputs in operator learning has been studied, e.g., in [68] convergence rates were analyzed for learning linear operators with noisy inputs-outputs data and in [225] an errors-in-variables modeling was adopted for operator learning. Nevertheless, they focused on analyzing and/or mitigating the effects of noisy inputs in NOs. How to quantify the uncertainties caused by both noisy inputs and outputs on the SciML models, especially PINNs and NOs, remains unclear.

In this work, we present a Bayesian approach for quantifying uncertainty induced by noisy inputs-outputs in PINNs as well as NOs. Specifically, we assume that we have measurement errors or noise for (1) the spatial-temporal coordinates (i.e., inputs) as well as the values of the function (i.e., outputs) in PINNs, and (2) the input and output functions in NOs, as illustrated in Figure 4.1. We point out that in the current study we treat NOs as pretrained surrogates models and use them to reconstruct the complete input and output functions from noisy and partial measurements (see Section 4.2 for details). Hence, the NOs are pretrained and in fact deterministic models, which is essentially different from training NOs with UQ as in [320, 173, 92]: in this work, the unknowns are the input and output functions instead of the NOs. This employment is one of the major functionalities

of NOs in the literature. To name a few examples, in [171, 172], Bayesian inverse problems were solved with pretrained FNOs where data of the solution are available; in [201, 348, 234], a combination of NNs and pretrained DeepONets was employed for mixed problems with partial and noisy data. While in PINNs, the unknowns are the NNs as in the literature of PINNs, which are stochastic models within the Bayesian framework because the NN parameters are treated as random variables.

The rest of this chapter is organized as follows. In Section 4.2, we introduce the problem formulation, PINNs and NOs, and discuss how PINNs and NOs are employed to encode the physics information. In Section 4.3, we present the Bayesian framework for quantifying uncertainty arising from noisy inputs-outputs under the frameworks of PINNs and NOs. In Section 4.4, we conduct four computational examples to demonstrate the consequences of ignoring the noise in the inputs and the effectiveness of our approach in addressing noisy inputs-outputs with UQ. We summarize our work in Section 4.5.

4.2 Encoding physics in scientific machine learning

Consider the following ODE/PDE:

$$\mathcal{F}_\lambda[u](x) = f(x), x \in \Omega, \quad (4.2a)$$

$$\mathcal{B}_\lambda[u](x) = b(x), x \in \partial\Omega, \quad (4.2b)$$

where Ω denotes the domain, $\mathcal{F}$ the general differential operator, $\mathcal{B}$ the general boundary operator, λ the parameter of the ODE/PDE, u , f and b the sought solution,

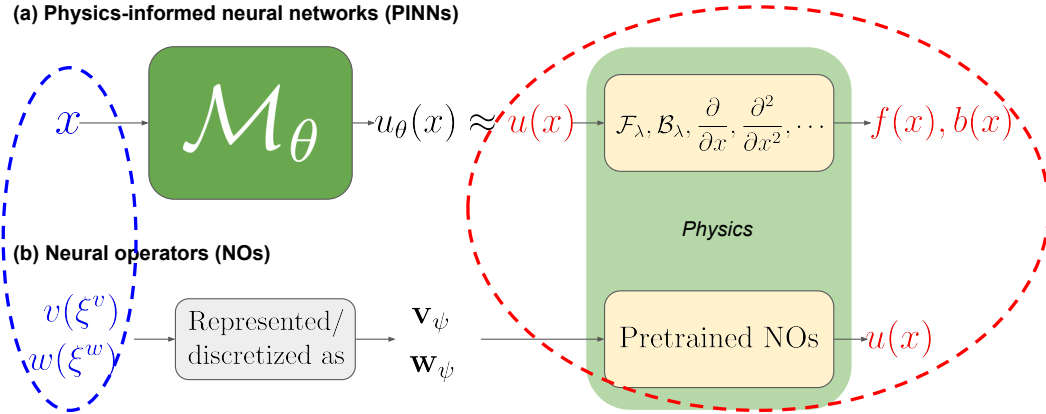


Figure 4.1: A schematic view of two prevalent ways of encoding physics in SciML. In **(a)**, we present the framework of physics-informed neural networks (PINNs) [244], where the physics is encoded by modeling $u(x)$ using a NN, parameterized by θ and denoted by $\mathcal{M}_\theta$, and approximating f, b via automatic differentiation and the ODE/PDE. In **(b)**, we represent the physics by pretrained neural operators (NOs), which are trained offline to learn the solution operators of (4.2), often based on clean and simulated data. The pretrained NOs effectively encode the underlying physics within the NN architecture and parameters [160], while in the downstream tasks such as solving ODEs/PDEs and tackling hybrid problems, they remain fixed and serve as equation-free surrogates. Here $\mathbf{v}_\psi$ and $\mathbf{w}_\psi$ denote discretizations of the input functions $v(\xi^v)$ and $w(\xi^w)$, respectively, which are parameterized by ψ . Note that in these two methods, the inputs and outputs (color-coded by blue and red, respectively) differ, making it crucial to develop uncertainty quantification (UQ) approaches to handle noisy inputs-outputs in both scenarios.

source term and initial/boundary term, respectively, and x the spatial-temporal coordinate. In this work we are interested in solving a class of problems using SciML, which is referred to as the *hybrid* problem, i.e., inferring u, f, b , and/or λ given data (denoted as $\mathcal{D}$) and the underlying physics. Note that if λ is known and u is not, the hybrid problem is equivalent to solving a forward ODE/PDE problem.

There are in general two common ways to encode physics in solving ODEs/PDEs problem in SciML: taking into consideration the physics-informed losses as in PINNs [244, 315, 277, 276] and other SciML models [269, 52, 345, 47, 46, 42, 271, 267], which are often realized via automatic differentiation, and using pretrained NOs

that represent the physics, e.g. DeepONets [178] and FNOs [171]. A schematic view of these two prevalent approaches is presented in Figure 4.1. As shown, the inputs of PINNs are spatial-temporal coordinates while they are functions in NOs. We briefly review the PINNs as well as NOs in what follows, and interested readers are directed to [244, 180] and [178, 171, 298, 172] for more details of PINNs and neural operators, respectively.

4.2.1 Physics-informed neural networks

The PINN method, originally proposed in [244], addresses the hybrid problem by modeling $u(x)$ (and λ if λ is unknown) with a NN parameterized by θ , denoted as $u_\theta(x)$ (and $\lambda_\theta(x)$ if λ is unknown), and then modeling $f(x)$ and $b(x)$ with $\mathcal{F}_\lambda[u_\theta](x)$ and $\mathcal{B}_\lambda[u_\theta](x)$ via automatic differentiation, respectively. The physics is then explicitly encoded in the training process by adding physics-informed terms involving $\mathcal{F}_\lambda[u_\theta](x)$ and $\mathcal{B}_\lambda[u_\theta](x)$. The optimal θ can be obtained by minimizing the following loss function:

$$\begin{aligned} \mathcal{L}(\theta) = & \frac{w_u}{N_u} \sum_{i=1}^{N_u} \|u_\theta(x_i^u) - u_i\|^2 + \frac{w_f}{N_f} \sum_{i=1}^{N_u} \|\mathcal{F}[u_\theta](x_i^f) - f_i\|^2 \\ & + \frac{w_b}{N_b} \sum_{i=1}^{N_b} \|\mathcal{B}[u_\theta](x_i^b) - b_i\|^2 + \frac{w_\lambda}{N_\lambda} \sum_{i=1}^{N_\lambda} \|\lambda_\theta(x_i^\lambda) - b_i\|^2, \end{aligned} \quad (4.3)$$

where w_u, w_f, w_b, w_λ are belief weights for balancing different terms, and $\|\cdot\|$ is the ℓ^2 -norm for finite-dimensional vectors, and $\{x_i^u, u_i\}_{i=1}^{N_u}$, $\{x_i^f, f_i\}_{i=1}^{N_f}$, $\{x_i^b, b_i\}_{i=1}^{N_b}$, $\{x_i^\lambda, \lambda_i\}_{i=1}^{N_\lambda}$ are observational data.

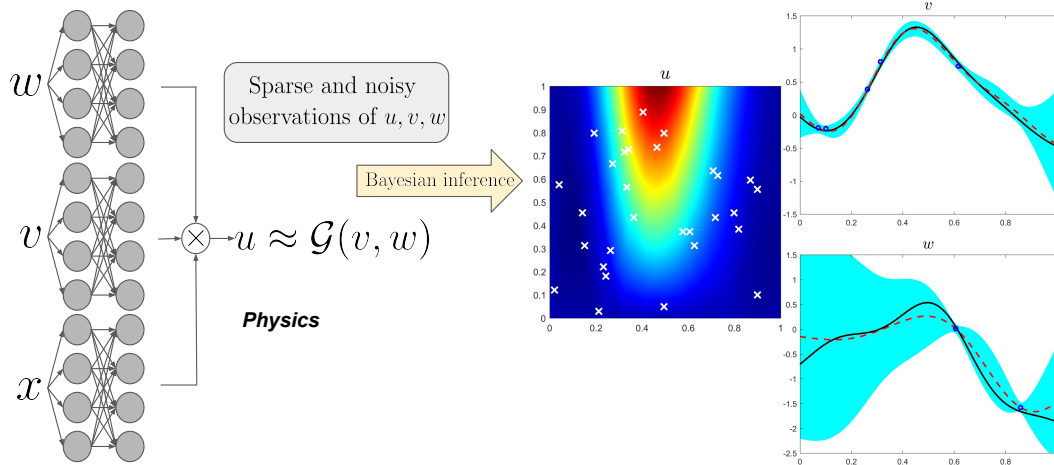


Figure 4.2: An example of using pretrained NOs to encode physics in the hybrid problem. In particular, we use a multi-input DeepONet [138] to represent the physics which connects u, v and w . Here, the multi-input DeepONet, denoted by $\mathcal{G}$, takes as input w and v and outputs u , while $\otimes$ denotes elementwise multiplication. In the hybrid problem, u, v, w are reconstructed with UQ from their noisy measurements and the physics, represented by the pretrained multi-input DeepONet, enabling synergistic learning, as shown in the predicted mean and uncertainty. See Section 4.4.5 for details.

4.2.2 Neural operators

The NOs are developed as fast solvers to families of ODEs/PDEs. Moreover, the pretrained NOs can also be employed as surrogates to represent the physics involved in addressing hybrid problems. This capability is regarded as one of the major functions of NOs in the literature [171, 172, 201, 234, 348, 343]. In particular, the solution operator of (4.2) is first approximated with NNs, denoted by $\mathcal{G}$. The corresponding training procedure is often considered as offline and is done with simulated and clean data to ensure the physics is accurately encoded. Then the pretrained NO $\mathcal{G}$ is used as a differentiable equation-agnostic surrogate, which encodes the underlying physical knowledge between the input and output functions (see Section 4.3.2 for details).

In this study, we choose Deep operator networks (DeepONets) [178] and Fourier

neural operators (FNOs) [171] as two representative methods for operator learning. Although both of them have been further developed to physics-informed versions, which achieve higher accuracy with fewer data [298, 172], we intentionally omit the physics-informed learning in this chapter for the sake of simplicity. DeepONet typically consists of two sub-networks: one that takes as input the spatial-temporal coordinate x , termed the *trunk net*, and another that takes as input the discretized function v , termed the *branch net*. On the other hand, FNO utilizes the fast Fourier transform over the discretization of the input function. For a comprehensive comparison between DeepONets and FNOs, we refer the readers to [179]. In addition to the vanilla version, we also consider the multi-input DeepONet [138], which maps multiple functions to a single function. We employ it as an equation-agnostic surrogate for hybrid problem involving more than two functions. An illustration of the architecture of a multi-input DeepONet and its associated hybrid problem is presented in Figure 4.2; see Section 4.4.5 for an example.

4.3 Uncertainty quantification for noisy inputs-outputs

In this section, we adopt the noisy input-output model formulated in (4.1) in the hybrid problem, and present the methodology of addressing it with UQ. Recall that we have the following model:

$$\begin{aligned}\tilde{y} &= \mathcal{H}(\chi) + \epsilon_o, \\ \tilde{\chi} &= \chi + \epsilon_{in}.\end{aligned}$$

Note that in the present study, $\mathcal{H}$ is (1) an NN in PINNs whose parameters are unknown, or (2) a pretrained NO whose parameters are known and fixed. Interpretations of inputs, outputs and NN model $\mathcal{H}$ are provided in Table 4.1. As shown, when the physics is encoded through PINNs and automatic differentiation, the input is the spatial-temporal coordinate and the outputs are the approximations of the solution u , the source term f , and/or the boundary term b . In such cases, PINNs are trained with the noisy inputs-outputs data. When the pretrained NOs, often trained with clean and simulated data, are used to represent the physics implicitly, the inputs are functions, e.g. model parameter and source/boundary terms, and the output is the approximation of u . We note that the physics could be encoded by employing both approaches simultaneously, i.e. using pretrained NOs as surrogates and adding physics-informed terms in the downstream training, as in [172, 298]. However, for the sake of simplicity in this study, we do not employ the physics-informed terms in NOs.

Methods of encoding physics	Inputs	Outputs	Model ($\mathcal{H}$)	Data
PINNs	x, t	$u/f/b$	$u_\theta/f_\theta/b_\theta$	Measurements of x, u, f, b
Pretrained NOs	v	u	$\mathcal{G}$	Measurements of v, u

Table 4.1: Interpretations of inputs, outputs and NN models $\mathcal{H}$ in PINNs and NOs. We note that when solving hybrid problems with NOs, $\mathcal{G}$ is pretrained, often offline based on clean and simulated data, and therefore remain fixed in reconstructing v and u from their noisy measurements. In contrast, PINNs are directly trained with the noisy inputs-outputs data.

We then adopt the Bayesian framework [70, 310] to address the noisy inputs-outputs in PINNs and pretrained NOs for solving hybrid problems. Specifically, we model the inputs χ as well as the parameter θ of the NN model $\mathcal{H}$ and establish likelihoods for measurements of both the inputs χ and outputs $\tilde{y}$ of $\mathcal{H}$ following (4.1). With the Bayes' theorem, we have the following:

$$p(\theta, \chi | \mathcal{D}) \propto p(\mathcal{D} | \theta, \chi) p(\theta, \chi), \quad (4.4)$$

where $p(\theta, \chi|\mathcal{D})$ is the posterior, $p(\mathcal{D}|\theta, \chi)$ is the likelihood, and $p(\theta, \chi)$ is the prior. Note that in [315, 201, 234, 348], only the noise in the outputs is considered. Consequently, the posterior is exclusively concerned with the NN parameter θ , i.e. $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$, and hence does not take into consideration the uncertainty induced by noisy inputs.

Similarly, we assume independent and identically distributed (i.i.d.) distribution for data ($\mathcal{D} = \{\tilde{\chi}_i, \tilde{y}_i\}_{i=1}^N$, where N denotes the number of data) and further assume ϵ_i and ϵ_o follow Gaussian distributions with mean zero and standard deviations σ_i and σ_o , respectively. Then, by the independence between $\tilde{y}_i$ and $\tilde{\chi}_i$ conditional χ_i , the likelihood can be expressed as follows:

$$\begin{aligned} p(\mathcal{D}|\theta, \chi) &= \prod_{i=1}^N p(\tilde{y}_i|\theta, \chi_i)p(\tilde{\chi}_i|\chi_i) \\ &= \prod_{i=1}^N \frac{1}{\sqrt{2\pi}\sigma_o} \exp\left(-\frac{\|\tilde{y}_i - \mathcal{H}(\chi_i)\|^2}{2\sigma_o^2}\right) \frac{1}{\sqrt{2\pi}\sigma_{in}} \exp\left(-\frac{\|\chi_i - \tilde{\chi}_i\|^2}{2\sigma_{in}^2}\right), \end{aligned} \quad (4.5)$$

where $\chi_i, i = 1, \dots, N$ denote the actual inputs where $\tilde{y}_i$ are observed, i.e. the exact values of $\tilde{\chi}_i$, and $\|\cdot\|$ denotes the Euclidean norm. We note that, although not explicitly expressed, here $\mathcal{H}$ depends on θ ; see also Table 4.1. By the independence between the NN parameter θ and the actual input $\tilde{\chi}$, the prior can be expressed as:

$$p(\theta, \chi) = p(\theta)p(\chi). \quad (4.6)$$

When $\mathcal{H}$ is an NN and no physics or neural operator is applied, the problem degenerates to a function approximation problem. A pedagogical example in addressing noisy inputs-outputs in function approximations can be found in the appendix of [347].

We note that although in this study we assume ϵ_{in} and ϵ_o follow Gaussian distri-

butions, the formulation of the noise model (defined in (4.1)) makes no assumption on the distributions of ϵ_{in} nor ϵ_o . Depending on the specific data measurements at hand, ϵ_{in} and ϵ_o could follow distributions with bounded support as well, e.g. uniform distribution or truncated normal distribution. We further remark that for certain problems in which in-domain computation is required, proper prior for χ , i.e., $p(\chi)$ in (4.6), can be specified to ensure that χ remains inside the computational domain even if the observed target value $\tilde{\chi}$ is out of the domain due to the noise. A relevant example is presented for demonstration in Section 4.4.3.

4.3.1 Physics-informed neural networks

Solving hybrid problems with noisy inputs-outputs under the PINN framework allows us to consider scenarios where both the spatial-temporal coordinate (x) and the value of functions (u, f, b, λ) are measured with noises. In this case, the posterior (4.4) can be directly integrated into the B-PINN framework to handle problems involving ODEs/PDEs with noisy inputs-outputs. We note that while diverse UQ methods exist for PINNs, our focus in this study is directed toward the Bayesian framework, which is widely employed for UQ in SciML due to their effectiveness and interpretability [234]. Given i.i.d. data of the sought solution u ($\{\tilde{x}_i^u, \tilde{u}_i\}_{i=1}^{N_u}$), the source term f ($\{\tilde{x}_i^f, \tilde{f}_i\}_{i=1}^{N_f}$), and/or the boundary term b ($\{\tilde{x}_i^b, \tilde{b}_i\}_{i=1}^{N_b}$), the likelihood is expressed as follows:

$$p(\mathcal{D}|\theta, \lambda, \{x_i^u\}_{i=1}^{N_u}, \{x_i^f\}_{i=1}^{N_f}, \{x_i^b\}_{i=1}^{N_b}) = \prod_{i=1}^{N_u} p(\tilde{u}_i|\theta, x_i^u) p(\tilde{x}_i^u|x_i^u) \prod_{i=1}^{N_f} p(\tilde{f}_i|\theta, \lambda, x_i^f) p(\tilde{x}_i^f|x_i^f) \prod_{i=1}^{N_b} p(\tilde{b}_i|\theta, \lambda, x_i^b) p(\tilde{x}_i^b|x_i^b), \quad (4.7)$$

where x_i^u, x_i^f, x_i^b are the actual values of $\tilde{x}_i^u, \tilde{x}_i^f, \tilde{x}_i^b$, respectively, and

$$p(\tilde{u}_i|\theta, x_i^u) = \frac{1}{\sqrt{2\pi}\sigma_o^u} \exp\left(-\frac{\|u_\theta(x_i^u) - \tilde{u}_i\|_2^2}{2(\sigma_o^u)^2}\right), \quad (4.8a)$$

$$p(\tilde{f}_i|\theta, \lambda, x_i^f) = \frac{1}{\sqrt{2\pi}\sigma_o^f} \exp\left(-\frac{\|\mathcal{F}_\lambda[u_\theta](x_i^f) - \tilde{f}_i\|_2^2}{2(\sigma_o^f)^2}\right), \quad (4.8b)$$

$$p(\tilde{b}_i|\theta, \lambda, x_i^b) = \frac{1}{\sqrt{2\pi}\sigma_o^b} \exp\left(-\frac{\|\mathcal{B}_\lambda[u_\theta](x_i^b) - \tilde{b}_i\|_2^2}{2(\sigma_o^b)^2}\right), \quad (4.8c)$$

in which $\sigma_o^u, \sigma_o^f, \sigma_o^b$ are the scales of the additive Gaussian noise in observing u, f, b , respectively. We note that in PINNs the value of u_θ at a specific coordinate x does not depend on the value λ , and hence we omit it. The likelihood for observing the inputs can be expressed as follows:

$$p(\tilde{x}_u^i|x_u^i) = \frac{1}{\sqrt{2\pi}\sigma_{in}^u} \exp\left(-\frac{\|x_u^i - \tilde{x}_u^i\|^2}{2(\sigma_{in}^u)^2}\right), \quad (4.9a)$$

$$p(\tilde{x}_f^i|x_f^i) = \frac{1}{\sqrt{2\pi}\sigma_{in}^f} \exp\left(-\frac{\|x_f^i - \tilde{x}_f^i\|^2}{2(\sigma_{in}^f)^2}\right), \quad (4.9b)$$

where σ_{in}^u and σ_{in}^f denote the scales of the additive Gaussian noise in observing the inputs of u and f , respectively. Here, we omit the likelihood for the input of b because, in practice, the exact location of the boundary is often known and hence the measured location should correspond to the actual one, i.e., $\tilde{x}_i^b = x_i^b$. The prior reads as $p(\theta, x) = p(\theta)p(x)$ and $p(\theta, \lambda, x) = p(\theta)p(\lambda)p(x)$ for the forward problem (the parameter of the ODE/PDE λ is known) and the inverse problem (λ is unknown), respectively. In summary, the posterior in PINNs can be written as

$$p(\theta, \lambda, \{x_i^u\}_{i=1}^{N_u}, \{x_i^f\}_{i=1}^{N_f}|\mathcal{D}) \propto p(\theta)p(\lambda) \prod_{i=1}^{N_u} p(\tilde{u}_i|\theta, x_i^u)p(\tilde{x}_i^u|x_i^u)p(x_i^u) \quad (4.10)$$

$$\prod_{i=1}^{N_f} p(\tilde{f}_i|\theta, \lambda, x_i^f)p(\tilde{x}_i^f|x_i^f)p(x_i^f) \prod_{i=1}^{N_b} p(\tilde{b}_i|\theta, \lambda, \tilde{x}_i^b).$$

As previously explained, here we set $\tilde{x}_i^b = x_i^b$ and therefore do not model x_i^b as unknown variables. We note that for inverse problems in PINNs, λ as well as the actual values of $\tilde{x}_i^u, \tilde{x}_i^f$ are treated the same as the NN parameter θ . We further remark that this (extended B-PINNs) framework is also capable of handling a combination of clean and noisy inputs.

An alternative strategy for addressing noisy inputs-outputs is recasting the input noise as a heteroscedastic output noise, as discussed in [306, 234]. For example, [194] explored the use of Gaussian processes to handle noisy inputs-outputs through a two-step procedure: (1) Gaussian processes regression is employed to fit the data and (2) then the input noise is substituted by an output noise via first-order Taylor expansion. Recall that our model for noisy inputs-outputs can be expressed as $\tilde{y} = \mathcal{H}(\tilde{\chi} - \sigma_{in}) + \sigma_o$ and hence the two-step approach proposed in [194] can in fact be integrated into a one-step approach via automatic differentiation [244]. We obtain the following approximation of the original model for one-dimensional $\tilde{\chi}$:

$$\tilde{y} \approx \mathcal{H}(\tilde{\chi}) - \mathcal{H}'(\tilde{\chi})\epsilon_{in} + \epsilon_o. \quad (4.11)$$

By adopting this approach, we do not need to model the actual input or the input noise. Instead, the original B-PINNs method can be directly applied with a heteroscedastic output noise, which follows a Gaussian distribution with mean zero and variance $(\mathcal{H}'(\tilde{\chi}))^2\sigma_{in}^2 + \sigma_o^2$. Note that this approach essentially integrates the delta method [214] into PINNs, ignores the Hessian of $\mathcal{H}$, and serves as an approximation of the original model for noisy inputs-outputs. It may not work as well as the original model, especially in solving ODEs/PDEs. An empirical comparison is conducted in Section 4.4.1.

4.3.2 Neural operators

When pretrained NOs are used to encode physics in hybrid problems, as discussed in Section 4.2, their inputs and outputs become functions. For the sake of simplicity in analyzing uncertainties, in this work we consider scenarios where only values of functions are noisy and corresponding coordinates are clean. The consideration of noisy coordinates in NOs requires the establishment of additional NNs to handle the input functions, which is beyond the scope of this work. As shown in Figure 4.1, solving a hybrid problem with a pretrained NO $\mathcal{G}$ only requires regression of the input function, v , since the regressed output function can be obtained immediately by $\mathcal{G}(\mathbf{v})$ where $\mathbf{v}$ denotes its discretization. In this regard, we directly model the discretization of v on a mesh, denoted by $\mathbf{v}_\psi$:

$$\mathbf{v}_\psi := [v_1^\psi, \dots, v_n^\psi]^T, \quad (4.12)$$

where n denotes the mesh size. The data are $\mathcal{D}_u = \{x_i^u, \tilde{u}_i\}_{i=1}^{N_u}$ and $\mathcal{D}_v = \{\tilde{v}_i\}_{i \in I_v}$, where I_v is the index set of the grid point on which v is measured. That is, in the hybrid problem we assume that v is observed on the subset of the same mesh used in training $\mathcal{G}$. Then, the posterior can be written as

$$p(\mathbf{v}_\psi | \mathcal{D}) \propto p(\mathcal{D}_u, \mathcal{D}_v | \mathbf{v}_\psi) p(\mathbf{v}_\psi) = p(\mathcal{D}_u | \mathbf{v}_\psi) p(\mathcal{D}_v | \mathbf{v}_\psi) p(\mathbf{v}_\psi), \quad (4.13)$$

where the last equality comes from the independence of observing $\mathcal{D}_u$ and $\mathcal{D}_v$ conditional on $\mathbf{v}_\psi$. Here, $p(\tilde{\mathbf{v}})$ is the prior and is either analytical known, e.g. a multivariate Gaussian, or can be estimated from the data used to train the NO [178, 171]. The likelihood is formulated as follows, assuming additive Gaussian noises with scales σ_o

and σ_{in} in observing u and v , respectively:

$$p(\mathcal{D}_u|\mathbf{v}_\psi) = \prod_{i=1}^{N_u} \frac{1}{\sqrt{2\pi}\sigma_o} \exp\left(-\frac{\|\mathcal{G}(\mathbf{v}_\psi)(x_i^u) - \tilde{u}_i\|^2}{2\sigma_o^2}\right), \quad (4.14a)$$

$$p(\mathcal{D}_v|\mathbf{v}_\psi) = \prod_{i \in I_v} \frac{1}{\sqrt{2\pi}\sigma_{in}} \exp\left(-\frac{\|v_i^\psi - \tilde{v}_i\|^2}{2\sigma_{in}^2}\right). \quad (4.14b)$$

4.3.3 Posterior estimate

In this study, we employ the Hamiltonian Monte Carlo (HMC) [211] to obtain samples from the posterior. For fast and stable implementations, a Python library termed NeuralUQ [348] is used. When using PINNs to encode the physics, we primarily focus on the function represented by the NN, and therefore at the testing stage clean input x is used as the inputs of the NN. In this regard, the predicted mean $\hat{u}(x)$ estimated by $\frac{1}{M} \sum_{i=1}^M u_{\theta_i}(x)$ is used as the prediction and the predicted variance defined as $\frac{1}{M} \sum_{i=1}^M (u_{\theta_i}(x) - \hat{u})^2$ is used as the quantified uncertainty. Here, $\{\theta_i\}_{i=1}^M$ are the posterior samples of the NN parameter θ (M denotes the number of posterior samples). In NOs, we are interested in the reconstruction of the input and output functions. The predicted means estimated by $\hat{\mathbf{v}} = \frac{1}{M} \sum_{i=1}^M \mathbf{v}_\psi^i$ and $\hat{u}(x) = \frac{1}{M} \sum_{i=1}^M \mathcal{G}(\mathbf{v}_\psi^i)(x)$ for $\mathbf{v}$ and u , respectively, are used as the predictions, where $\{\mathbf{v}_\psi^i\}_{i=1}^M$ are the posterior samples of $\mathbf{v}_\psi = [v_1^\psi, \dots, v_n^\psi]$. Similarly, the predicted variances $\frac{1}{M} \sum_{i=1}^M (\mathbf{v}_\psi^i - \hat{\mathbf{v}})^2$, $\frac{1}{M} \sum_{i=1}^M (\mathcal{G}(\mathbf{v}_\psi^i)(x) - \hat{u}(x))^2$ are used as the quantified uncertainties.

4.4 Numerical examples

In this section we present six computational examples: (1) a 1D nonlinear Poisson equation, (2) an ODE system describing a diesel engine, (3) a nonlinear 2D steady-state Allen-Cahn equation, (4) a 1D Burgers equation, (5) a 1D reaction-diffusion equation with heteroscedastic diffusion coefficient, and (6) a 120-dimensional parametric Darcy problem. In the first three examples, we employ the present method to quantify uncertainties in PINNs with noisy inputs and outputs. In the following three examples, we use the proposed approach to quantify uncertainties when the physics is encoded in two different pretrained neural operators, i.e., DeepONet and FNO.

We note that the NO is pretrained offline with clean and sufficient data such that it approximates the solution operator accurately and therefore is able to represent the physics. Furthermore, in the downstream hybrid problem, the test functions are in-distribution and therefore follow the physics encoded in the NOs beforehand. In the literature of NOs [178, 171, 160], data of the input functions for training and testing are often sampled from general distributions whose probabilistic density functions are analytically known, e.g. Gaussian random fields. We follow this setup in this work, and in hybrid problems with pretrained NOs, the distribution used to generate training data is employed as the prior of the discretization of the input functions, i.e. $p(\mathbf{v}_\psi)$ in (4.13). Details of data generation, the training of NOs, and hyperparameters in numerical experiments can be found in Appendix C.1.

4.4.1 A nonlinear 1D Poisson equation with PINNs

Consider the following 1D nonlinear Poisson equation:

$$\kappa \frac{\partial^2 u}{\partial x^2} - \lambda u^3 = f(x), \quad x \in [0, 1], \quad (4.15)$$

with the Dirichlet boundary condition $u(0) = u(1) = 1$ and κ and λ being non-negative constants. The data are generated by assuming $u(x) = \cos^3(2\pi x)$, $\kappa = 0.01$ and $\lambda = 0.1$. In this example, we employ the B-PINN approach and consider noisy measurements of u , f and x with the present approach.

4.4.1.1 Forward problem

	Error of f (%)	Error of u (%)
clean data	0.03	0.00
clean input and noisy output	7.16	3.57
noisy input and clean output	13.50	11.73

Table 4.2: PINN for Poisson equation: Inference of u and f with different training data for f . The error is defined as the relative L_2 error between the exact and the prediction.

To demonstrate the importance and necessity of respecting the noise in inputs, we start with a *forward problem* where κ and λ are known and we solve (4.15) given data on (x, f) . We first employ vanilla PINNs for the following three different scenarios: clean input and output data, clean input but noisy output data, and noisy input but clean output data. In all scenarios, the boundary condition is hard-encoded in the model, and we assume that 51 available measurements for (x, f) , which are equidistantly distributed on $[0, 1]$, are used to train PINNs. The measurement noise for both x and f is additive Gaussian noise with a scale 0.01. Results are shown in Table 4.2 and Figure C.1, from which we can see noisy data impair the performance

of PINNs significantly. In particular, even with the same scale, the noise in x has larger damage upon the accuracy of PINNs in solving the equation than the noise in f .

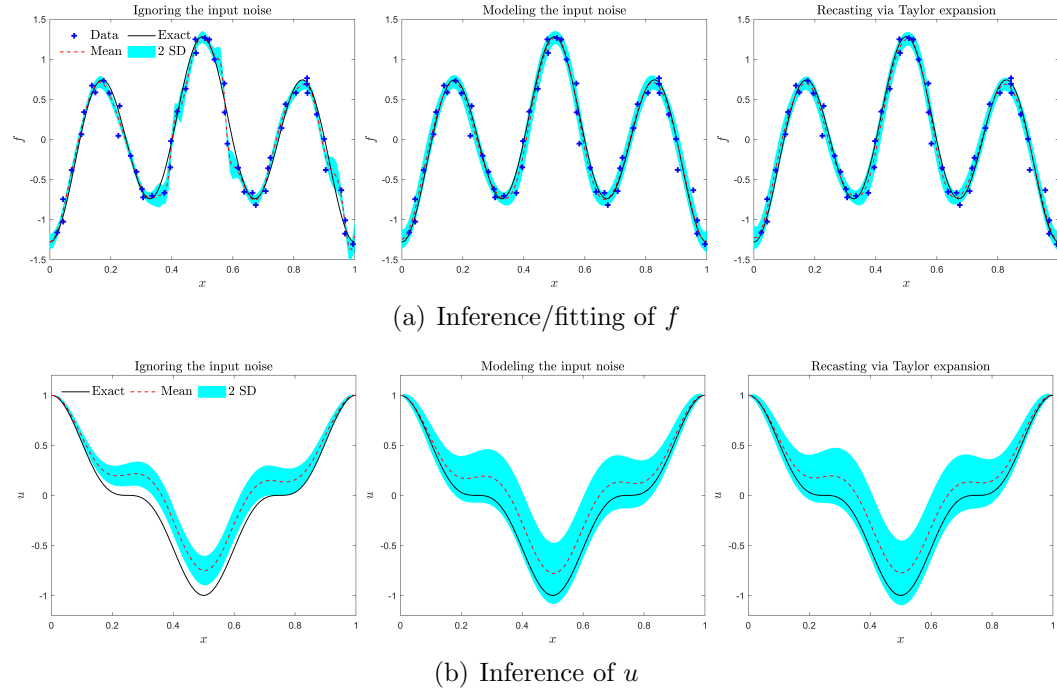


Figure 4.3: Results from solving (4.15) in the B-PINN framework based on noisy inputs-outputs data of f using different approaches. The boundary condition is clean and hard-encoded in the modeling. From left to right are results from the original B-PINN method which ignores the noise in x , our approach which models the noise in x , and the recasting method which recasts the noise in x as a heteroscedastic noise in f via first-order Taylor expansion.

	Ignoring the input noise	Modeling the input noise	Recasting
Error of f	16.18%	8.16%	9.13%
Error of u	32.17%	28.17%	28.65%

Table 4.3: Errors of solving (4.15) with noisy inputs-outputs data of f using different methods. The error is defined as the relative L_2 error between the exact and the predicted mean.

Now we solve (4.15) with noisy inputs-outputs data of f , i.e., (x, f) . In this case, 51 data points of f are sampled equidistantly on $[0, 1]$ and their inputs and outputs are corrupted by additive Gaussian noise with scales 0.01 and 0.05, respectively. The boundary condition is clean and hard-encoded in the modeling. We employ and

compare three methods: the original B-PINN method and two of our approaches proposed in Section 4.3.1. Results are presented in Figure 4.3 and Table 4.3. We can see that the original B-PINN method, which ignores the noise in x , is unable to provide accurate predictions (as shown in Table 4.3). The predicted uncertainties of u and f (as shown in Figure 4.3) from the original B-PINN method also fail to bound the errors between the exact and the predictions. Slight overfitting in predicting f is observed from results of B-PINNs shown in Figure 4.3(a), showing the consequence of the unawareness of the input noise. Meanwhile, our approaches present much better results in terms of the predicted means and uncertainties: errors are small and bounded by the predicted uncertainties. We show in Figure 4.3(b) that the consideration of the noise in the input with our approaches yields significantly larger quantified uncertainty in inferring u , indicating successful capturing of uncertainty induced by noisy inputs and demonstrating the effect of uncertain inputs in solving (4.15) with noisy data. Besides, as presented in Figure 4.3(a), the uncertainty band of our approach, which is obtained by $[\mu - 2\sigma, \mu + 2\sigma]$ where μ is the predicted mean and σ is the predicted standard deviation, is horizontally (along the x -axis) larger than the original B-PINNs, showing the effect of considering the input noise. As reported in Table 4.2, modeling noises in both the coordinate and the value f with our approaches results in significantly higher accuracy in fitting f as well as inferring u , proving the effectiveness of the proposed approach in solving (4.15) given noisy inputs-outputs data.

4.4.1.2 Inverse problem

We then move to an *inverse problem* where $\kappa = 0.01$ is known but λ is not. In this case, we aim to identify λ as well as to reconstruct u from noisy data of u and f . We note that the unknown parameter in the inverse problem is treated the same as the

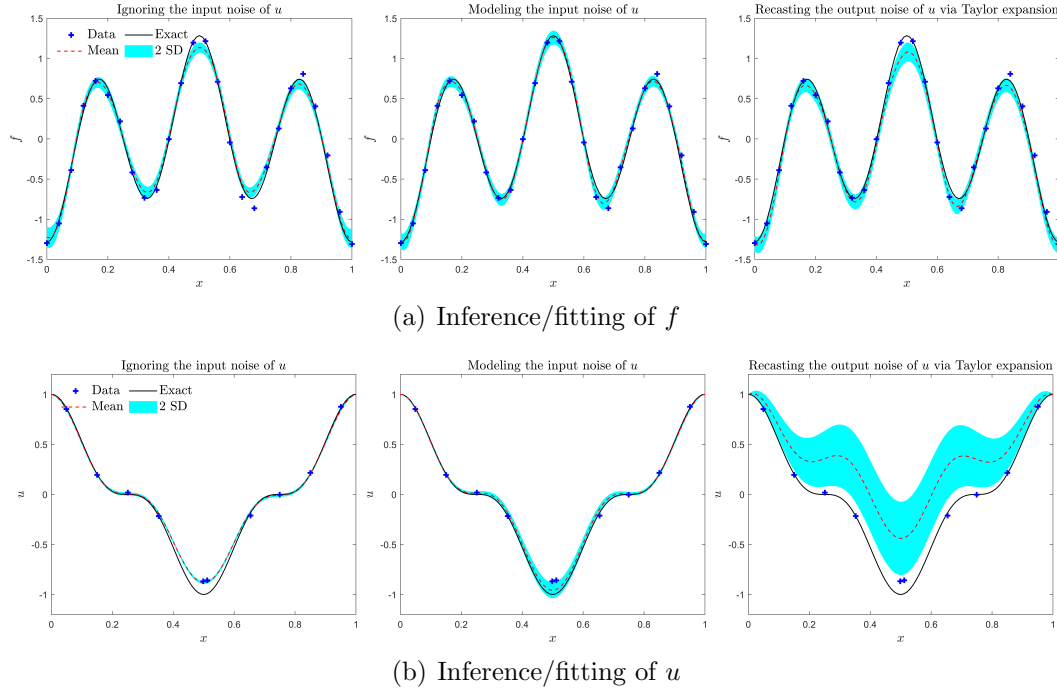


Figure 4.4: Results from solving (4.15) with λ unknown based on noisy inputs-outputs data of u and noisy outputs data of f . The boundary condition is clean and hard-encoded in the modeling. We note that in this case data of u are noisy in both the input (heteroscedastic noise; defined in (4.16)) and the output while data of f are noisy in the output but clean in the input, demonstrating the flexibility of our approach. From left to right are results from the original B-PINN method, our approach, and the recasting method.

NN parameter [244, 315]. Therefore, the complexity increases with the number of unknown parameters. In this specific case, 26 data points of f and 10 data points of u , equidistantly sampled on $x \in [0, 1]$, are available. To demonstrate the flexibility and compatibility of the proposed approach in handling a hybrid of clean and noisy inputs, in this case we corrupt the data of u in both their inputs and outputs but corrupt data of f only in their outputs. All the noise are assumed to be Gaussian. The noise scales for the outputs of u and f are 0.01 and 0.05, respectively. Further, we consider a more complex scenario by assuming a heteroscedastic noise model [234]

	Ignoring the input noise	Modeling the input noise	Recasting
Error of u	7.83%	4.23%	66.04%
Inference of λ	0.2793 ± 0.0440	0.1333 ± 0.0537	0.2108 ± 0.0459

Table 4.4: Inferences of λ (mean $\pm$ standard deviation) and relative L_2 errors of the predicted mean for u using different methods to solve the inverse problem. The reference solution for λ is 0.1.

for the input of u , which is a function of x expressed as follows:

$$\sigma_{in}(x) = 0.001 + \begin{cases} 0.4(x - 1/3), & x \in (\frac{1}{3}, \frac{1}{2}], \\ -0.4(x - 2/3), & x \in (\frac{1}{2}, \frac{2}{3}], \\ 0, & x \in (0, \frac{1}{3}] \cup (\frac{2}{3}, 1], \end{cases} \quad (4.16)$$

where σ_{in} denotes the scale of the input noise of u . Specifically, the noise scale increases as the measurement moves away from the boundary (see the visualization of the noise model in Figure C.2). Results are shown in Figure 4.4 and Table 4.4, from which we can see that the original B-PINN method fails to yield trustworthy inferences of u and λ (errors are large while predicted uncertainties are small) because it does not take into consideration the noise in the input of u . As a comparison, results of our approach, as displayed in Table 4.4, show significantly smaller error in inferring λ and reconstructing u , demonstrating the effectiveness of the proposed approach in solving the inverse problem with noisy inputs-outputs data. Moreover, Figure 4.4(b) exhibits that our approach is capable of producing reliable predictions, in which the predicted uncertainty bounds the error between the predicted mean and the exact of u . Besides, we demonstrate that recasting the output noise via Taylor expansion is ineffective in addressing the noisy inputs-outputs for the inverse problem in this scenario.

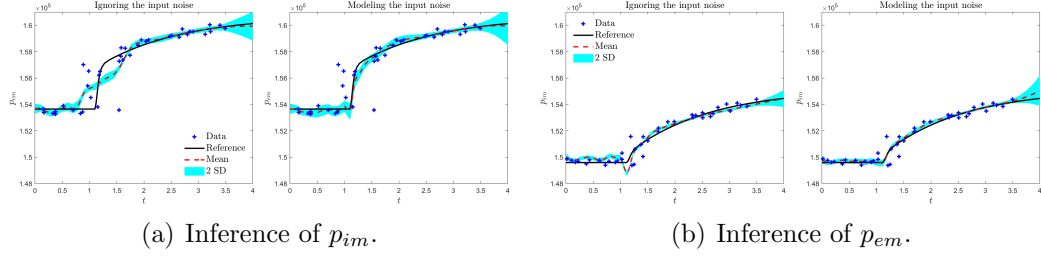


Figure 4.5: Predictions for (a) p_{im} and (b) p_{em} for solving the inverse ODE problem in the diesel engine with noisy inputs-outputs data. Here we employ the original B-PINN method which ignores the input noise (**left**) and our approach which models the input noise (**right**). 40 noisy inputs-outputs data of p_{im} and p_{em} and 40 noisy outputs data of f_{im} and f_{em} are used as the training data.

	Inference of c_1 ($\times 10^{-4}$)	Inference of c_2 ($\times 10^{-3}$)
Ignoring the input noise	3.2593 ± 0.0573	-8.8294 ± 0.0577
Modeling the input noise	-2.2835 ± 0.1008	-3.2543 ± 0.1012

Table 4.5: Inferences of c_1, c_2 (mean $\pm$ standard deviation) for the inverse ODE problem given noisy inputs-outputs. The reference values for c_1 ($\times 10^{-4}$) and c_2 ($\times 10^{-3}$) are -2.0817 and -3.4465 , respectively [290].

4.4.2 Inverse ODE problem in diesel engine system

To further test the performance of the present approach in real problems, we now employ it to solve an inverse ODE problem in a diesel engine. The problem considered here can be viewed as a submodel of the one developed in [290] to describe the gas flow dynamics in a diesel engine, which is expressed as:

$$\frac{dp_{im}}{dt} = \frac{R_a T_{im}}{V_{im}} (W_c + W_{egr} - W_{ei}) + f_{im}, \quad (4.17a)$$

$$\frac{dp_{em}}{dt} = \frac{R_e T_{em}}{V_{em}} (W_{eo} - W_t - W_{egr}) + f_{em}, \quad (4.17b)$$

where the ideal gas constant of air R_a , ideal gas constant of exhaust gas R_e , intake manifold volume V_{im} , exhaust manifold volume V_{em} , and the intake manifold temperature T_{im} , are known constants. In addition, the intake manifold pressure p_{im} , exhaust manifold pressure p_{em} , exhaust manifold temperature T_{em} , compressor mass

flow W_c , exhaust gas recirculation mass flow W_{egr} , total mass flow W_{ei} , and the mass flow out from the cylinder W_{eo} are time-dependent functions [290]. The total mass flow W_{ei} and the mass flow out from the cylinder W_{eo} are modeled as follows [290]:

$$W_{ei} = \frac{\eta_{vol} p_{im} n_e V_d}{120 R_a T_{im}}, \quad (4.18)$$

$$W_{eo} = W_{ei} + W_f, \quad (4.19)$$

where n_e, V_d, W_f are constants and the volumetric efficiency, denoted as η_{vol} , is modeled as:

$$\eta_{vol} = c_1 \sqrt{p_{im}} + c_2 \sqrt{n_e} + c_3, \quad (4.20)$$

where c_1, c_2, c_3 are referred to as the volumetric efficiency constants in the literature [290, 208]. The description of the aforementioned constants and the corresponding values employed in our computations are the same as in [208]. The source terms f_{im} and f_{em} are assumed to be zero.

In this example, we infer the values of the volumetric efficiency constants (c_1 and c_2) given the noisy measurements of p_{im}, p_{em}, f_{im} and f_{em} . We note that since n_e is a constant, c_2 and c_3 are linearly dependent and hence we only model and infer the values of c_1 and c_2 . Also, we assume that T_{em}, W_c, W_{egr} and all other constants can be measured and are treated as known for simplicity. Specifically, we assume that we have access to 40 measurements of p_{im} and p_{em} , corrupted by additive Gaussian noises in both the input and output, and 40 the measurements of f_{im} and f_{em} , corrupted by additive Gaussian noises only in the output. The data are generated using the *MATLAB Simulink* simulation (available at <http://www.fs.isy.liu.se/Software>) [290] for $t \in [0, 4]$ and the training data are assumed to distribute at $t \in [0, 3.5]$. In the implementation, we rescale p_{im}, p_{em} such that their values, as well as the right-hand sides of (4.17), are on the scale of one. We assume

that the noise scales for the (rescaled) outputs of p_{im}, p_{em} and f_{im}, f_{em} are 0.02, 0.02, 0.2, 0.2, respectively, and the following heteroscedastic noise model for the inputs of p_{im} and p_{em} is considered:

$$\sigma_{in}(t) = 0.2 + \begin{cases} 0.19(t - 1), & t \in [0, 1], \\ -0.19(t - 1)/4, & t \in (1, 4]. \end{cases} \quad (4.21)$$

This noise model represents that the scale of the input noises of p_{im} and p_{em} peak at $t = 1$ and decay as t moves away from $t = 1$ (the visualization of the noise model is presented in Figure C.2(b)). Results are presented in Figure 4.5 for p_{im}, p_{em} ($t \in [0, 4]$) and Table 4.5 for c_1, c_2 , from which we can see that modeling the input noise is necessary for accurate and trustworthy inferences when solving the inverse problem with noisy inputs-outputs.

We would also like to discuss the computational cost for the vanilla B-PINNs and the present method for this specific problem. The computations are performed on a standard laptop (13th Gen Intel(R) Core(TM) i9-13900HX with 2.20 GHz processor). In average of five independent trials, the vanilla B-PINN method took 89.1 seconds, and the proposed approach took 92.8 seconds due to modeling and inferring the actual inputs. As shown, the computational cost for the present approach is similar as the vanilla B-PINNs, but is able to provide more accurate results for this specific problem. The details of the HMC setup can be found in Appendix C.1.

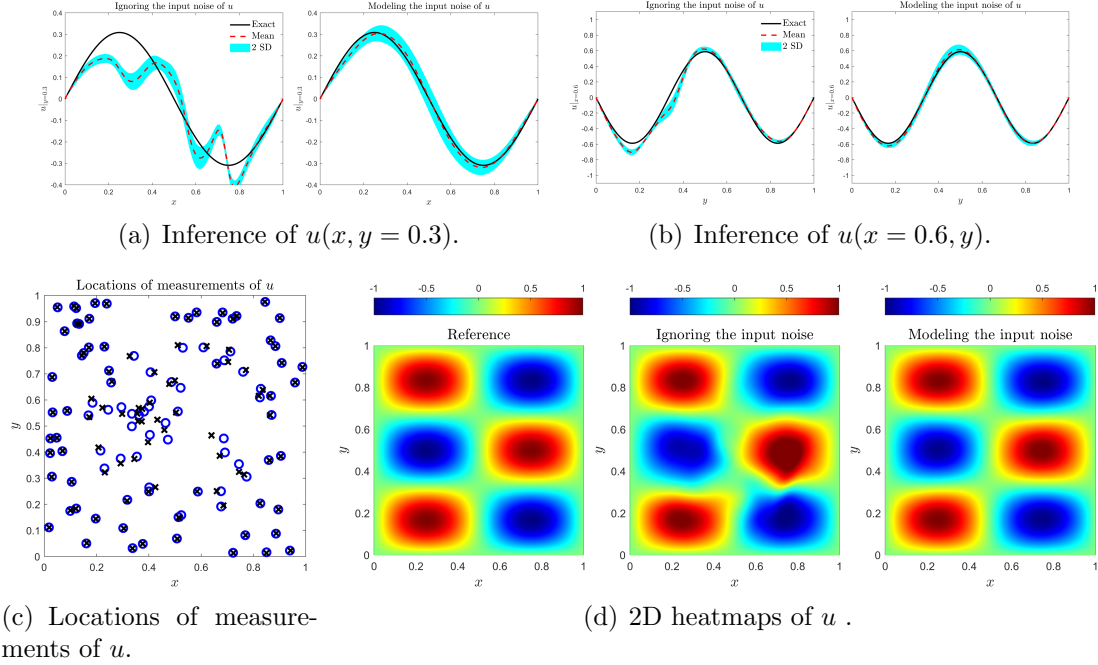


Figure 4.6: Results from solving the inverse problem of (4.22) based on noisy inputs-outputs data with D and λ unknown. In (c) we show the locations of the measurements of u , in which **blue** circles represent the actual locations and **black** crosses represent the measured locations.

	Error of u	Inference of $D (\times 10^2)$	Inference of λ
Ignoring the input noise of u	12.17%	0.8201 ± 0.0293	0.8451 ± 0.0259
Modeling the input noise of u	4.23%	0.9522 ± 0.0233	0.9499 ± 0.0236

Table 4.6: Inferences of D and λ (mean $\pm$ standard deviation) and relative L_2 errors of the predicted mean for u . The reference values for $D (\times 10^2)$ and λ are 1.

4.4.3 A nonlinear 2D steady-state Allen-Cahn equation with PINNs

We consider the following 2D steady-state Allen-Cahn equation [315]:

$$D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \lambda u(1 - u) = f(x, y), x, y \in [0, 1], \quad (4.22a)$$

$$u(x, 0) = u(x, 1) = 0, \quad x \in [0, 1], \quad (4.22b)$$

$$u(0, y) = u(1, y) = 0, \quad y \in [0, 1], \quad (4.22c)$$

where $D > 0$ and λ are constants. The exact expressions of u and f are obtained from $u(x, y) = \sin(2\pi x) \sin(3\pi y)$, $x, y \in [0, 1]$ and $D = 0.01, \lambda = 1$. In this example, we assume that we have 200 and 100 noisy measurements for f and u , respectively. All the measurements are randomly sampled (subject to the 2D uniform distribution) from the domain. The boundary conditions are assumed to be known and are hard-encoded in the NN model. In addition, D and λ are assumed to be unknown. We now solve the inverse problem and employ our approach to infer the D/λ , reconstruct u , and quantify the uncertainty induced by noisy inputs-outputs. A specific prior on the real location of the measurement, i.e., $p(\chi)$, where χ denotes the exact input, is employed to ensure that the real location remains within the computational domain $([0, 1]^2)$ regardless of the measured location (see the details for the prior in Appendix C.1). Similar to the previous example, the data of u are corrupted in both their inputs and outputs while the data of f are corrupted only in the input. We assume that the scales of noise for the outputs of u and f are 0.01 and 0.1, respectively, and we consider the following heteroscedastic noise model for the input of u :

$$\sigma_{in}(x, y) = 0.001 + \begin{cases} 0.1(b - \|(x - x_c, y - y_c)\|_2), & \|(x - x_c, y - y_c)\|_2 \leq b, \\ 0, & \|(x - x_c, y - y_c)\|_2 > b, \end{cases} \quad (4.23)$$

where $b = 0.4$, σ_{in} denotes the scale of the input noise of u , and x_c, y_c are the coordinates of the center of the domain, i.e., $x_c = 0.5, y_c = 0.5$. This noise model indicates that the noise scale is determined by the distance between the exact location of the measurement and the center of the domain and increases as the exact location moves closer to the center (the visualization of the noise model is shown in Figure C.2).

We present results of the inverse problem with noisy inputs-outputs in Figure 4.6 and Table 4.6, from which we observe that our approach, which models the

input noise, is able to provide accurate inferences in u , D , and λ , compared to the original B-PINN method which ignores the input noise. Moreover, the predicted uncertainty of our approach successfully bounds the error between the reference and the predicted mean. Similar to previous examples, ignoring the input noise of the data of u affects negatively the identification of model parameters in the inverse problem. Similarly, the computational cost was evaluated on the same machine in average of five independent trials: the original B-PINN method took 97.6 seconds while the proposed approach took 101.2 seconds (see Appendix C.1 for the details of the HMC setup).

4.4.4 1D Burgers' equation with FNOs

We now employ the present method to quantify the uncertainty in the predictions of NOs with noisy inputs and outputs. Specifically, we consider the following 1-D Burgers' equation with the periodic boundary condition:

$$\begin{aligned} \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} &= \nu \frac{\partial^2 u}{\partial x^2}, \quad x \in [0, 1], \quad t \in (0, 1], \\ u(x, t = 0) &= u_0, \quad x \in [0, 1], \end{aligned} \tag{4.24}$$

where $\nu = 0.01$ is the viscosity and u_0 is the initial condition. Here, we first train an FNO to learn the solution operator from the initial condition $u|_{t=0}$ to $u|_{t=1}$ using clean and sufficient data. With the pretrained FNO which encodes the physics, we reconstruct $u|_{t=0}$ and $u|_{t=1}$ from their sparse and noisy measurements with uncertainties. In particular, we assume that eight noisy measurements of $u|_{t=0}$ (0.05 noise scale) and three of $u|_{t=1}$ (0.01 noise scale) are available. We note that in this example, the discretization of $u|_{t=0}$ is the input of the FNO, which is $\mathbf{v}_\psi$ as defined in 4.12 and displayed in Figure 4.1, while $u|_{t=1}$ is the output.

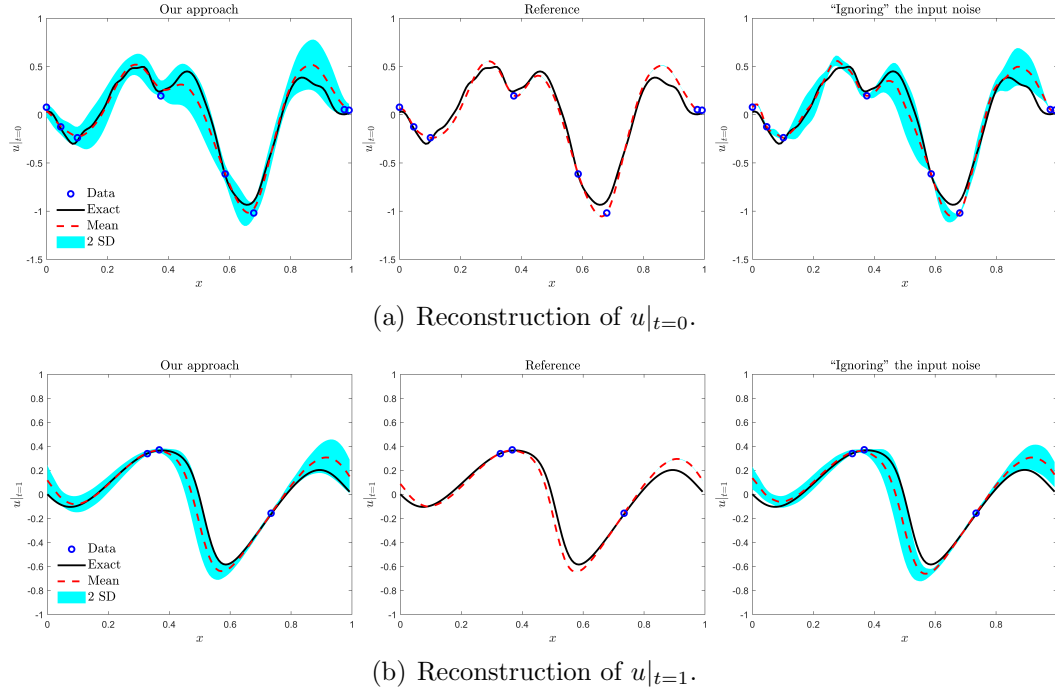


Figure 4.7: Burgers' equation: Reconstructing $u|_{t=0}$ and $u|_{t=1}$ from their noisy measurements and a pretrained FNO, which represents the physics and serves as an equation-free surrogate. From left to right are results from our approach with correctly specified input noise, the reference method, and our approach but misspecifying the input noise to a small value (0.001) in constructing the likelihood. Here, the reference method refers to the deterministic inference in which a maximum a posteriori (MAP) estimate is performed without UQ.

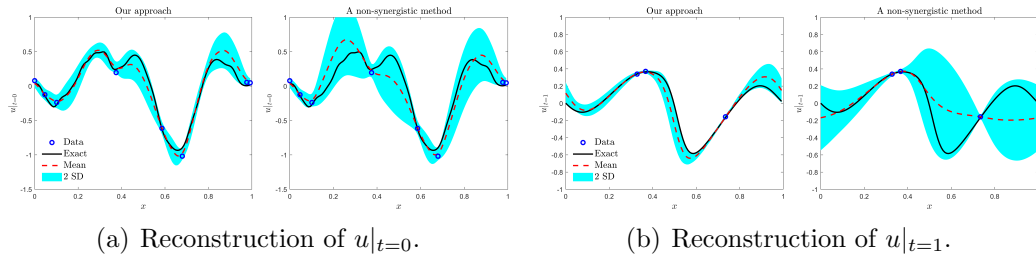


Figure 4.8: Burgers' equation: Comparison between our approach which reconstructs $u|_{t=0}$ and $u|_{t=1}$ synergistically (**left**) and a non-synergistic method (**right**), in which these two functions are reconstructed independently from their respective sparse and noisy measurements without access to data of the other function.

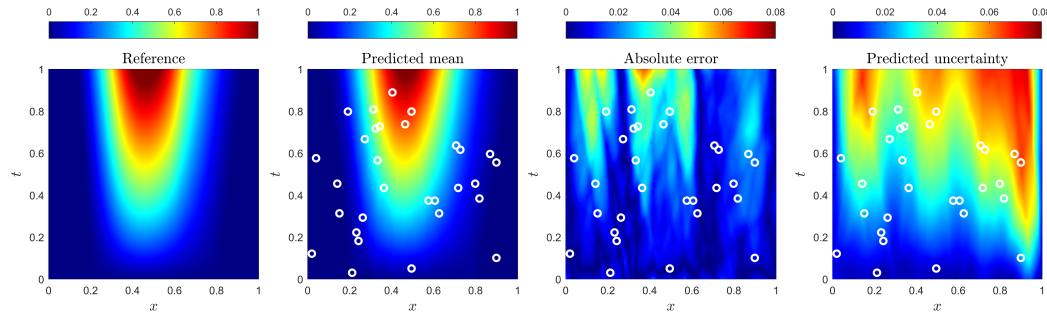
Results of our approach are presented in the first column of Figure 4.7, from which we observe that the reconstructions are generally accurate even with limited and noisy data and the errors between the references and the predicted means are

bounded by the predicted uncertainties. In the second column of Figure 4.7, we present results from reconstructing the input and output functions deterministically, which is achieved by performing a maximum a posteriori (MAP) estimate of the posterior in (4.13) with gradient descent. The comparison indicates the importance of UQ in such hybrid problems for trustworthy and reliable predictions. Another comparison is also made between ignoring and respecting the noise in the measurements of the input function. As shown in the third column of Figure 4.7, misspecifying the scale of the input noise to a very small value (0.001) not only produces worse inference of the input function $u|_{t=0}$ but also impairs the inference of the output function $u|_{t=1}$, demonstrating the consequence of ignoring/misspecifying the noise in the input when solving the hybrid problem with pretrained NOs.

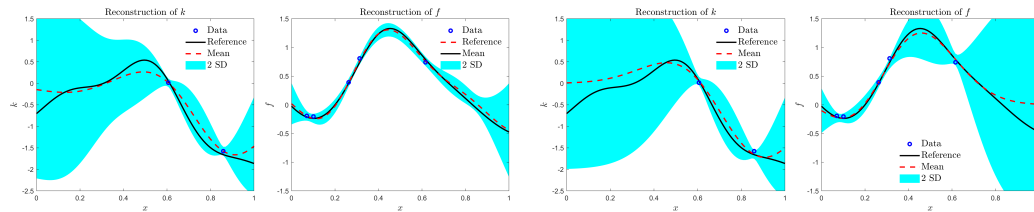
In this example, we further compare our approach with a non-synergistic learning method for reconstructing $u|_{t=0}$ and $u|_{t=1}$, in which these two functions are reconstructed independently from their respective sparse and noisy measurements without access to information of the other function. Specifically, the posterior for the discretization of the input function becomes $p(\mathbf{v}_\psi|\mathcal{D}_v) \propto p(\mathbf{v}_\psi)p(\mathcal{D}_v|\mathbf{v}_\psi)$ and no pretrained FNO is employed, indicating that the physics is not utilized; see (4.14a). Meanwhile the posterior for inferring the output function reads $p(\mathbf{v}_\psi|\mathcal{D}_u) \propto p(\mathbf{v}_\psi)p(\mathcal{D}_u|\mathbf{v}_\psi)$ where the pretrained FNO is employed in the likelihood $p(\mathcal{D}_u|\mathbf{v}_\psi)$; see (4.14b). Results of this comparison are presented in Figure 4.8, from which we observe significant increase in both the errors and the predicted uncertainties in reconstructing the input and output functions of the FNO. Although the non-synergistic method also provides reliable predictions in which the predicted uncertainties bound the errors between the predicted means and the references, its decoupling of the learning of $u|_{t=0}$ and $u|_{t=1}$ hinders the potential for synergistic learning. Conversely, in our approach a pretrained FNO is employed to represent the physics, bridging

these two functions with the underlying physics. As a result, the reconstructions benefit from the synergy of information from the physics and data of both functions, leading to more accurate and confident predictions. The differences in predicted uncertainties between these two approaches showcase synergistic learning realized by our approach. In this section, we follow the same setup from [171, 179] and set the viscosity $\nu = 0.1$ to demonstrate our approach in tackling noisy inputs in NOs. Other interesting topics such as Burgers' equation with smaller or zero viscosity will be investigated in future work.

4.4.5 1D time-dependent reaction-diffusion equation with multi-input DeepONets



(a) Reconstruction of u .



(b) Reconstructions of k (left) and f (right). (c) A non-synergistic learning method.

Figure 4.9: 1D time-dependent reaction-diffusion equation with heteroscedastic diffusion term: Reconstructions with uncertainties of u, k, f , as displayed in (a) and (b), from their sparse and noisy measurements and a pretrained multi-input DeepONet, which represents the physics in (4.25). White circles represent locations of measurements of u . In (c), a non-synergistic method is employed to reconstruct k and f independently from their respective measurements without access to data of the other function or data of u .

In this example, we consider the following reaction-diffusion PDE (adapted from [138]):

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}(0.01(|k(x)| + 1)\frac{\partial u}{\partial x}) + \kappa u^2 + f(x), x \in [0, 1], t \in [0, 1], \quad (4.25)$$

with zero initial and boundary conditions, where $\kappa = 0.01$, and $k(x)$ and $f(x)$ are space-dependent terms, representing the diffusion and the source term, respectively. Similar as in Section 4.4.4, we train a NO with clean and sufficient data to learn the solution operator from $k(x)$ and $f(x)$ to $u(x, t)$. We then would like to reconstruct k, f, u from their sparse and noisy measurements. In particular, a multi-input DeepONet [138] (see Figure 4.2 for the architecture of the NO) is employed to learn the solution operator, represent the physics and therefore bridge k, f and u . We then assume we have two random measurements of k , five random measurements of f , and 30 random measurements of u , and measurements are corrupted by the additive Gaussian noise with scale 0.05. By the independence in observing data and the independence between priors of k and f , the posterior in this case reads:

$$p(\mathbf{k}_\psi, \mathbf{f}_\psi | \mathcal{D}_f, \mathcal{D}_k, \mathcal{D}_u) \propto p(\mathcal{D}_f | \mathbf{f}_\psi) p(\mathcal{D}_k | \mathbf{k}_\psi) p(\mathcal{D}_u | \mathbf{k}_\psi, \mathbf{f}_\psi) p(\mathbf{k}_\psi) p(\mathbf{f}_\psi), \quad (4.26)$$

where $\mathbf{k}_\psi$ and $\mathbf{f}_\psi$ denote the discretization of k and f , respectively. A simpler case can be found in Appendix C.2, where the diffusion term is a known constant and the task becomes reconstructing f and $u|_{t=1}$ from their sparse and noisy measurements.

Results are displayed in Figure 4.9(a-b), from which we can see all three functions are reconstructed accurately and the predicted uncertainties are reasonable: they bound the errors between the references and the predicted means. Both the errors and the predicted uncertainties grow as one moves away from the region where data are available. In Figure 4.9(c) we present results from a non-synergistic

method in inferring k and f . Similar as in Section 4.4.4, the non-synergistic method reconstructs these two functions independently from their respective measurements without access to data of the other function, data of u , or the pretrained multi-input DeepONet. Specifically, the posterior for the discretization of k becomes $p(\mathbf{k}_\psi|\mathcal{D}_k) \propto p(\mathbf{k}_\psi)p(\mathcal{D}_k|\mathbf{k}_\psi)$ and the posterior for the discretization of f is $p(\mathbf{f}_\psi|\mathcal{D}_f) \propto p(\mathbf{f}_\psi)p(\mathcal{D}_f|\mathbf{f}_\psi)$. By comparison, we observe significant decrease in the errors and predicted uncertainties when our approach is employed, demonstrating the effectiveness of the proposed approach and the pretrained multi-input DeepONet in representing the underlying physics. The errors and predicted uncertainties from our approach do not develop as fast as the non-synergistic method as one moves away from the region where data are available, because information of k , f , and u are coupled by the pretrained NO.

4.4.6 120-dimensional Darcy problem with DeepONets

We consider the Darcy's law describing a steady flow through porous media in two dimensions [234, 348, 285]:

$$\nabla \cdot (\lambda(x, y) \nabla u(x, y)) = f, (x, y) \in (0, 1)^2, \quad (4.27)$$

where λ denotes the hydraulic conductivity field, u denotes the hydraulic head, and $f = -40$ is a constant. The boundary conditions are:

$$u(0, y) = 1, \quad u(1, y) = 0, \quad y \in (0, 1) \quad (4.28a)$$

$$\partial_{\mathbf{n}} u(x, 0) = \partial_{\mathbf{n}} u(x, 1) = 0, \quad x \in (0, 1), \quad (4.28b)$$

where $\mathbf{n}$ denotes the unit normal vector of the boundary.

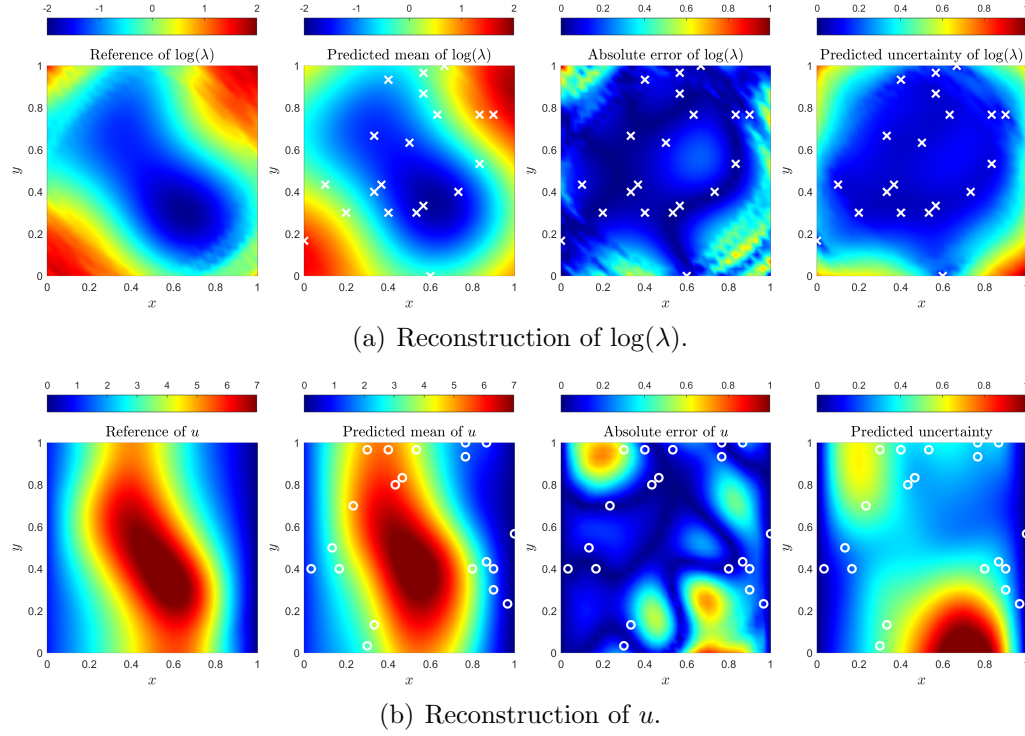


Figure 4.10: 120-dimensional Darcy problem: Reconstructions with uncertainties of $\log(\lambda)$ (shown in (a)) and u (shown in (b)) from their sparse and noisy measurements as well as a pretrained DeepONet representing the physics in (4.27). The white cross stands for the location where λ is observed while the white circle stands for the location where u is observed.

In this study, we use the following model to describe the hydraulic conductivity field, i.e. $\lambda(x, y) = \exp(k(x, y))$, where $k(x, y)$ is a sample of a truncated Karhunen-Loève expansion of a Gaussian process with zero mean and the following exponential squared kernel:

$$k(x_1, y_1, x_2, y_2) = \exp\left(-\frac{(x_1 - x_2)^2}{2l^2} - \frac{(y_1 - y_2)^2}{2l^2}\right), \quad x_1, x_2, y_1, y_2 \in [0, 1], \quad (4.29)$$

where $l = 0.25$ is the correlation length. We keep the first 120 leading terms of the expansion, which leads to a 120-dimensional parametric problem.

Similar as previous examples, a DeepONet is pretrained with clean data to represent the physical relation between λ and u : it approximates the solution operator

which maps $\log(\lambda)$ to u . We assume that we have noisy and gappy data of u and $\log(\lambda)$ and we reconstruct these two functions synergistically based on the pretrained NO. In our numerical experiments, 20 measurements of u and 20 measurements of $\log(\lambda)$ are randomly sampled and corrupted by additive Gaussian noises with scales 0.5 and 0.05, respectively. Results are presented in Figure 4.10. As shown, we observe that the predicted means are accurate compared to the references and the errors between the predicted means and the reference are mostly bounded by the predicted uncertainties.

4.5 Summary

In this chapter, we proposed a Bayesian approach to quantify uncertainties arising from the noisy inputs-outputs in two prevalent scientific machine learning (SciML) models, the physics-informed neural networks (PINNs), which encode the physics via automatic differentiation [244], and neural operators (NOs) [160], which learn the hidden physics from data and serve as equation-free surrogates [171, 201, 234, 348]. In particular, we targeted solving hybrid problems using PINNs and NOs. In contrast to most existing work on uncertainty quantification (UQ) in SciML in which only noisy outputs are considered [234], our approach encompasses the scenarios, where the inputs of SciML models (e.g., the spatial-temporal coordinate in PINNs and the input functions in NOs) are corrupted by noises as well. The consideration of the noise in the inputs provides two immediate benefits: (1) the deployment of SciML models in tackling problems where uncertainties are associated with spatial-temporal coordinates, e.g. particle image velocimetry (PIV) [35, 31] and hydrological modeling [147, 148], is facilitated; (2) accounting for input noise allows for the compatibility of pretrained Neural Operators (NOs) with noisy input function measurements. This

versatility makes the NOs more robust and adaptable in practical applications.

The proposed approach can seamlessly be combined with the PINNs, vanilla Fourier neural operators (FNOs) [171], vanilla DeepONets [178], and multi-input DeepONets [138] in addressing the *hybrid* problem with noisy inputs and outputs. We considered six numerical examples to demonstrate the effectiveness of the present method. Specifically, we considered three examples for PINNs, in which both the coordinates and the corresponding responses were measured with noises, i.e., (1) a 1D Poisson equation with PINNs, (2) an inverse ODE problem in the diesel engine, and (3) a 2D steady-state Allen-Cahn equation. We further considered three examples for NOs, i.e., (4) a 1D Burgers equation, where the solution was reconstructed from the noisy and sparse measurements on the input/output functions with pretrained FNOs; (5) a 1D time-dependent reaction-diffusion equation with heterogeneous diffusion term in which the solution, the source term and the diffusion term were reconstructed from noisy measurements with pretrained multi-input DeepONet; and (6) a 120-dimensional Darcy problem, where the hydraulic head and the hydraulic conductivity were regressed synergistically from sparse and noisy data. The results show that the proposed method is able to provide accurate inferences and quantify uncertainties given noisy inputs-outputs data.

In the current study, the noise in the inputs of the test cases is generally smaller than the one in the outputs. In numerical experiments, it has been observed that the present approaches failed to provide accurate inferences in terms of the predicted mean and uncertainties when the noise in the inputs is significantly larger than the noise in the outputs. Further study needs to be conducted to develop more robust approaches for handling inputs with large noises. Further, in this work we have employed Hamiltonian Monte Carlo (HMC) [211] to estimate the posterior distribution for the purpose of accurate posterior inference and therefore have only considered

problems with datasets of small sizes. Efficient Bayesian approaches [234, 348], e.g. variational inference and Laplace approximation, can be utilized to tackle more complicated problems involving noisy inputs-outputs. In addition, for operator learning problems, we only consider the case in which we have sufficient clean simulated data to pretrain the NOs. However, in real-world applications it is possible that we only have noisy measurements for both the input and output functions to train the NOs. Besides, in the present study we assume that the underlying physics is accurately encoded through either correctly specified differential equation in PINNs or sufficiently trained NOs, while in practice physics/model misspecification in SciML will lead to inaccurate prediction and uncertainty quantification, as discussed in [346]. Further study is needed to extend the proposed approach to address noisy inputs-outputs and model misspecification at the same time. Finally, we note that the priors in PINNs/NOs in each test case of this study is prescribed, and it will affect the predicted accuracy especially for cases with few training data. Although several ways to optimize the prior, e.g., maximizing the marginal likelihood using variational inference and Laplace, have been discussed in [234], how to obtain the best prior of BNNs remains an open question [234]. All the limitations discussed above lead to interesting future work on extending the presented approach to a broader range of problems. We expect our approach to be effectively and reliably applied to various practical tasks arising from computational science and engineering.

CHAPTER FIVE

Physical model misspecification and uncertainty

5.1 Introduction

Accurate governing equations or physical models are essential for understanding and quantifying physical processes across diverse scientific disciplines, such as the Navier-Stokes equations in fluid dynamics and the transport equations in geosciences. Generally, the governing equations are derived based on established theories such as the conservation of mass and momentum, thermodynamic laws, and so on. However, there are also complex systems in which the governing equations are difficult to obtain theoretically due to the lack of understanding for certain physical processes, e.g., the constitutive relation in highly nonequilibrium flows [330], the forcing caused by solar and volcanic variability in climate problems [87], non-equilibrium reactions, etc. How to obtain analytical expressions that represent physical phenomena in nature and engineering remains an open question.

With the rapid growth of available data and computing power, data-driven discovery of governing equations based on machine learning algorithms has emerged as a new paradigm for obtaining accurate physical models as the alternative to theoretical derivations [262, 29, 256, 52, 333, 168, 103, 278, 250, 332, 337, 116]. To name a few examples, Schmidt *et al.* presented a symbolic regression approach to discover equations from experimental data [262]; Brunton *et al.* proposed the sparse identification of nonlinear dynamical systems (SINDy) to learn both ordinary and partial differential equations from data [29, 256]. In addition, Chen *et al.* employed deep neural networks (DNNs), more specifically the physics-informed neural networks (PINNs), which encode the physical laws into DNNs via automatic differentiation [244], to discover governing equations from scarce data [52]. In most of the aforementioned approaches, a library of candidate terms that describe all the possible physical processes is required as prior knowledge [262, 29, 256, 52]. However,

mathematical expressions or models for certain physical processes are quite challenging to obtain in some complex real-world applications, e.g., chemical reactions in combustion, the constitutive relation in highly partially ionized plasmas, etc. It is probable that we miss some details of the physical process, hence misspecifying the physical models in such scenarios [22]. An incomplete or non-comprehensive library leads to the discrepancy between the observational data and the assumed governing equations. In other words, the discovered governing equations may not be accurate enough to describe the physical processes represented by the observational data.

Several recent approaches have been developed to reduce the discrepancy between the observational data and the assumed physical models. For instance, Ebers *et al.* [77] proposed to learn the *missing physics* or the discrepancy between the imperfect models and the measurements using a plurality of methods, e.g. Gaussian process regression (GPR), DNNs, etc.; Chen *et al.* also employed DNNs to recover the *missing physics* in prior models [53]; in [334, 76], combinations of PINNs and symbolic regression were employed to identify the missing physics in biological problems, e.g. Alzheimer’s disease; Saurabh *et al.* [187] and Zhu *et al.* [338] discovered source terms of dynamic systems in a “gray-box” fashion with NN-based numerical integrators. However, most of the existing work focuses on resolving the missing physics. Investigations on scenarios that we *misspecify the physical models* are rare to the best of our knowledge. Further, the observational data from sensors in real-world applications are generally noisy, and may also be incomplete due to the limitation of data collection techniques. Such noisy and incomplete or gappy data lead to uncertainties in the predictions of machine learning algorithms or more specifically the scientific machine learning algorithms [234, 348]. In addition, as widely discussed in the climate change modeling [223, 106, 22], the incomplete understanding of physical processes will also result in uncertainties in simulations. We point out that quantifying uncer-

tainties arising from noisy/gappy data as well as the incomplete understanding of the physical processes is an important issue but has been largely ignored in the existing literature on data-driven discovery of governing equations. In the current study, we refer to the uncertainties in the discovered physical models as *model uncertainty*.

In the present work, we aim to address the issue of misspecifying physical models as well as quantifying model uncertainty in the data-driven discovery of governing equations. Specifically, PINNs are employed as the backbone to discover the governing equations or physical models from data due to their effectiveness as well as easy implementations. In particular, we first encode the assumed physical models that may be misspecified in PINNs, and add another DNN as the correction to the misspecified model to alleviate the discrepancy between the imperfect models and the observational data. We note that the library of candidate terms used in [262, 29, 256, 52] is not required here since the added DNN can serve as a universal approximator to model the discrepancy. The Bayesian physics-informed neural networks (B-PINNs) [315] and/or ensemble PINNs [234, 348] are used to quantify the uncertainties in the discovered governing equations or physical models arising from the noisy and/or gappy data. Further, the symbolic regression is utilized to obtain the analytical expression for the unknown physics based on the trained DNNs that are used to correct the model misspecifications in PINNs.

The rest of this chapter is organized as follows. In Section 5.2, we introduce the model misspecification in PINNs as well as our approach to correct it, and also the methods for quantifying uncertainties in PINN predictions. In Section 5.3, we conduct four numerical experiments to demonstrate the effectiveness of the proposed approach in addressing the model misspecification issue in PINNs. We summarize the present work in Section 5.4.

5.2 Model misspecification in PINNs for learning governing equations

5.2.1 Learning governing equations from data using PINNs

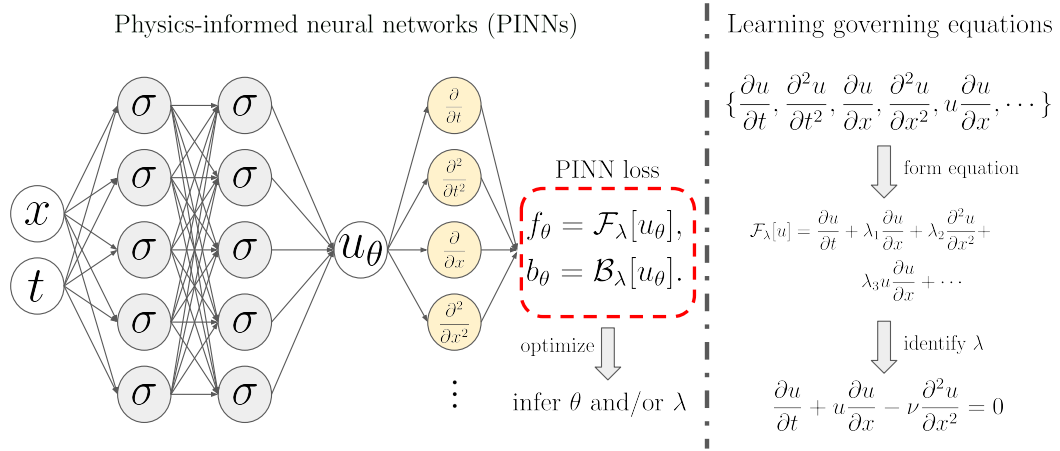


Figure 5.1: A schematic view of PINNs for learning governing equations from data. On the left, a general framework of PINNs is displayed where σ denotes the activation function, θ denotes the DNN parameters, u_θ denotes the output of the DNN that is used to approximate u , and f_θ/b_θ are computed via automatic differentiation [244]. An example on learning governing equations with PINNs is illustrated on the right panel. Here a 1D Burgers equation is learned ($\lambda_1 = 0, \lambda_2 = 1, \lambda_3 = -\nu$ where ν denotes the viscosity) given data on u .

Consider the following nonlinear ODE/PDE describing a physical system:

$$\mathcal{F}_\lambda[u](x) = f(x), x \in \Omega, \quad (5.1a)$$

$$\mathcal{B}_\lambda[u](x) = b(x), x \in \partial\Omega, \quad (5.1b)$$

where Ω is the domain, λ is the model parameter, u is the sought solution, f is the source term, b is the boundary term, $\mathcal{F}_\lambda$ and $\mathcal{B}_\lambda$ are operators parameterized by λ , defining the equation and the boundary condition. The PINN method is capable of solving ODEs/PDEs given initial/boundary conditions as well as learning governing

equations and identifying model parameters from data [244, 315, 348, 180, 32, 219, 151, 134, 135, 344, 347, 175, 174, 169, 276, 277, 198, 259, 111]. A schematic view of the PINN method is presented in Figure 5.1, in which a DNN with σ as the activation function and θ as the parameter is built to approximate u with u_θ and f, b with f_θ, b_θ , respectively, via automatic differentiation [244].

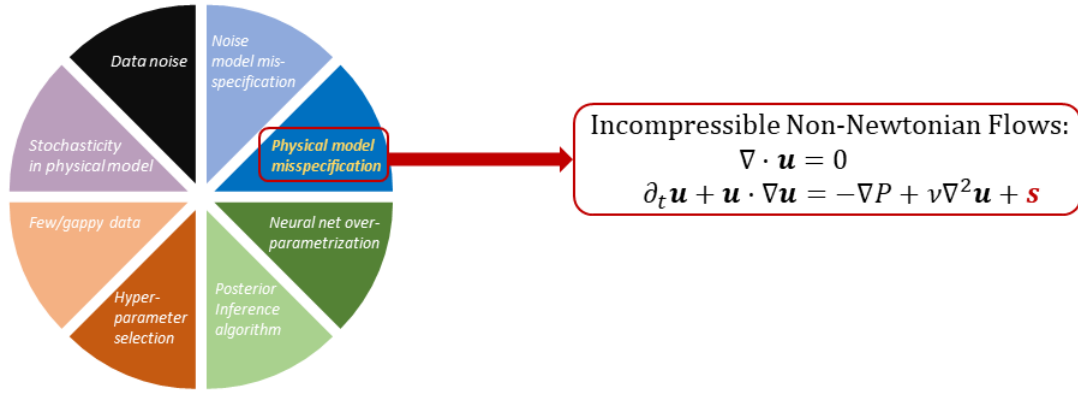
In the present study, we focus on using PINNs to learn governing equations from data. The dataset for learning governing equations with PINNs is denoted by $\mathcal{D}$, which can be expressed as $\mathcal{D} = \mathcal{D}_u \cup \mathcal{D}_f \cup \mathcal{D}_b$, where $\mathcal{D}_u = \{x_i^u, u_i\}_{i=1}^{N_u}$ is the set of data for u , and $\mathcal{D}_f = \{x_i^f, f_i\}_{i=1}^{N_f}$ and $\mathcal{D}_b = \{x_i^b, b_i\}_{i=1}^{N_b}$ are sets of data for the physics. The model parameter λ , which defines the equation, is then obtained by optimizing the following PINN loss function with the gradient-descent method and its variants [255]:

$$\mathcal{L}(\theta) = w_u \mathcal{L}_{\mathcal{D}_u}(\theta) + \mathcal{L}_{PDE}(\theta), \quad (5.2)$$

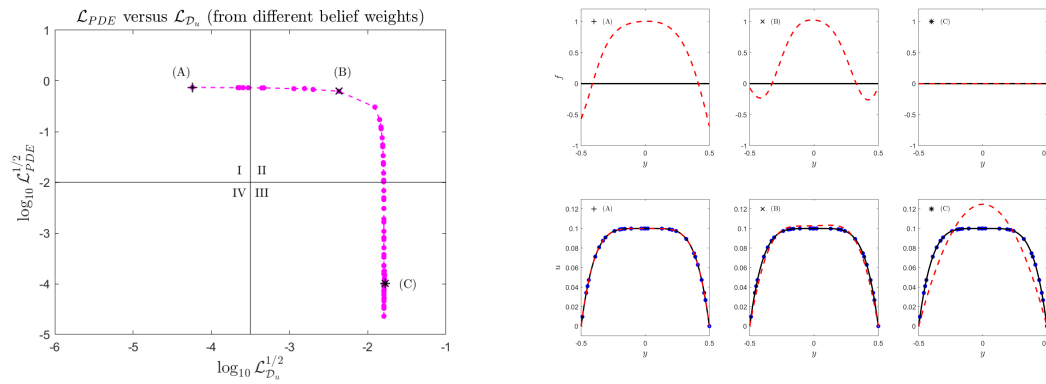
where

$$\begin{aligned} \mathcal{L}_{\mathcal{D}_u}(\theta) &= \frac{1}{N_u} \sum_{i=1}^{N_u} \|u_\theta(x_i^u) - u_i\|_2^2, \\ \mathcal{L}_{PDE}(\theta) &= \frac{w_f}{N_f} \sum_{i=1}^{N_u} \|\mathcal{F}_\lambda[u_\theta](x_i^f) - f_i\|_2^2 + \frac{w_b}{N_b} \sum_{i=1}^{N_b} \|\mathcal{B}_\lambda[u_\theta](x_i^b) - b_i\|_2^2, \end{aligned}$$

w_u, w_f, w_b are belief weights for balancing different terms, and $\|\cdot\|_2$ is the ℓ^2 -norm for finite-dimensional vector. We note that in certain cases we may not have data corresponding to the boundary/initial conditions, i.e., the last term in $\mathcal{L}_{PDE}(\theta)$, and we can directly drop this term in the computations.



(a) Uncertainty from physical model misspecifications.



(b) When PINNs encounter model misspecification.

Figure 5.2: In (a), a breakdown of total uncertainty in PINNs is presented. In (b), results from PINNs encountering model misspecifications are displayed, where a physical model of Newtonian fluid is assumed but the real physics, which generates the data, is non-Newtonian. The PINN method is employed to identify the homogeneous viscosity with different belief weights in the loss function, i.e. pairs of w_{PDE} and w_{D_u} in (5.2), resulting in different pairs of PDE loss $\mathcal{L}_{PDE}$ and data loss $\mathcal{L}_{D_u}$, shown in the left of (b). Due to the model misspecification, the PDE loss $\mathcal{L}_{PDE}$ and the data loss $\mathcal{L}_{D_u}$ cannot be minimized simultaneously. Three results are presented in the right of (b), corresponding to three cases in the left, + (data are fitted well but the physics is not satisfied), * (the physics is satisfied but data are not fitted), and x (neither data nor physics are fitted). We note that the boundary condition is hard-encoded in the modeling. Details can be found in Section 5.3.3.1.

5.2.2 Correcting the model misspecification in PINNs

The computational accuracy of PINNs in learning governing equations strongly depends on the encoded physical models, i.e. $\mathcal{F}_\lambda$ and/or $\mathcal{B}_\lambda$ in (5.1). However, the

correct specification of the physics is not always guaranteed, leading to significant discrepancy between the observational data and the assumed governing equations. An example is demonstrated in Figure 5.2 where the flow is non-Newtonian but the physical model is misspecified as the one for Newtonian flows (see Section 5.3.3.1 for details). As a result, the physics and the data cannot be fitted well at the same time: the training does not land on the region “IV”, where both the losses for the data as well as the residual of the equations are close to zero, regardless of the choice of belief weights in the loss function (5.2), i.e. w_f for the physics and w_u for the data. Three results from PINNs are sampled and displayed in Figure 5.2(b): when u_θ fits the data (which come from a non-Newtonian flow), it does not satisfy the governing equations for Newtonian flows; when u_θ is forced to follow the Newtonian flow by setting large w_{PDE} , it presents Newtonian behavior and thus cannot fit the data from the non-Newtonian flow well.

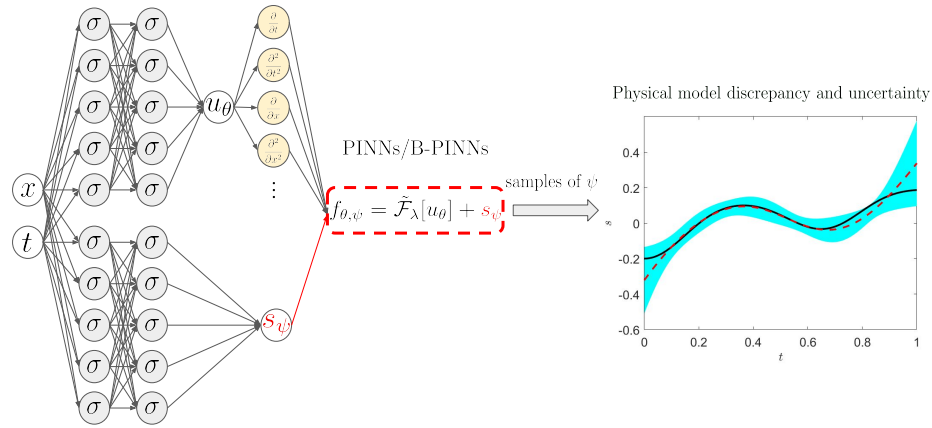


Figure 5.3: An overview of the proposed approach for correcting the misspecified physics. Here we assume that the nonlinear differential operator $\mathcal{F}$ is misspecified as $\tilde{\mathcal{F}}_\lambda$ such that $\tilde{\mathcal{F}}_\lambda[u] \neq f$, which leads to poor performance of PINNs in learning governing equations. Compared to the conventional PINN method shown in Figure 5.1, an additional DNN, parameterized by ψ , is used to model the discrepancy. We note that s_ψ serves two roles: (1) correcting the equation such that data of u can be explained, and (2) modeling the discrepancy of the physics. With posterior samples of ψ , we obtain the prediction as well as the model uncertainty of the discrepancy. We also note that DNNs in this figure, namely u_θ and s_ψ , are deterministic NNs in ensemble PINNs and Bayesian NNs (BNNs) in B-PINNs.

The discrepancy between observational data and the assumed governing equations is caused by the physics/model misspecification. In this regard, we propose a general approach developed upon the original PINN framework to quantify the discrepancy as well as to correct the model misspecification. In this work, we focus on the misspecification of the differential operator $\mathcal{F}_\lambda$, but the proposed approach can be easily generalized to the boundary/initial operator. In addition, prior information is required regarding which operator is misspecified, the differential or the boundary one. For example, if we have data on the boundaries, then we are able to detect whether or not there is misspecification on the boundary condition and will assign a DNN to correct it if there is. Here, we denote the misspecified differential operator by $\tilde{\mathcal{F}}_\lambda$ and then the discrepancy can be quantified as follows:

$$s(x) = f(x) - \tilde{\mathcal{F}}_\lambda[u](x). \quad (5.3)$$

Inspired by the PINN method, in which the sought solution u is modeled with a DNN parameterized by θ , we employ another DNN parameterized by ψ to model $s(x)$, denoted as $s_\psi(x)$. The source term $f(x)$ is approximated by $\tilde{\mathcal{F}}_\lambda[u_\theta](x) + s_\psi(x)$, and hence the term penalizing the violation of the physics in the loss function (5.2) is reformulated as:

$$\mathcal{L}_{PDE}(\theta, \psi) = \frac{w_f}{N_f} \sum_{i=1}^{N_u} \|\tilde{\mathcal{F}}_\lambda[u_\theta](x_i^f) + s_\psi(x_i^f) - f_i\|_2^2 + \frac{w_b}{N_b} \sum_{i=1}^{N_b} \|\mathcal{B}_\lambda[u_\theta](x_i^b) - b_i\|_2^2. \quad (5.4)$$

We note that s_ψ plays two important roles in our approach handling model misspecification: (1) it approximates the function quantifying the model discrepancy, i.e. s , and (2) it allows both the data term $\mathcal{L}_{\mathcal{D}_u}(\theta)$ and the PDE term $\mathcal{L}_{PDE}(\theta, \psi)$ to be minimized to very small values simultaneously in the training of PINNs such that data are fitted by u_θ and the (corrected) physics are satisfied. Furthermore, s in general is a function of the solution u or possibly the derivatives of u . Here we

treat s as a function of x and/or t , which will not affect its generality since u is also a function of x/t . However, this setup makes the present approach a general framework and is able to reduce the prior knowledge on the formulation of s .

5.2.3 Uncertainty quantification

As mentioned in Section 5.2.2, a key step in the present work is to learn the correction term, i.e., s_ψ , that is used to fix the issue of model misspecifications in PINNs. Generally, the measurements in real-world applications are noisy and can also be incomplete, leading to the uncertainties in predictions from PINNs [234, 348, 347]. We refer to the uncertainties in s_ψ as *model uncertainty*. Here, two typical approaches are employed for quantifying the uncertainties in s_ψ , i.e., ensemble PINNs [234, 348] and B-PINNs [315]. The former is computationally efficient but has difficulties preventing overfitting when the data are noisy, while the latter is able to handle the noisy data well but is computationally more expensive. We therefore utilize the ensemble PINNs and B-PINNs for cases with clean and noisy data, respectively.

We now briefly review the ensemble PINNs and B-PINNs. The ensemble PINN method follows the deep ensemble method, proposed in [163], and employs multiple standard training of PINNs independently. That is, PINNs with the PDE loss term in (5.4) are trained multiple times with random initialization of the DNN parameter. Each random initialization corresponds to one initial guess for minimizing the PINN loss function (5.2) and each training is finding one global minimum. As a result, ensemble PINNs can be considered as identifying different maximum a posteriori (MAP) estimates of the PINN loss function [234]. On the other hand, the B-PINN method focuses on estimating the posteriors for θ and ψ based on the Bayes' rule as

follows:

$$p(\theta, \psi | \mathcal{D}) \propto p(\mathcal{D} | \theta) p(\theta). \quad (5.5)$$

Here $p(\theta | \mathcal{D})$ is the density function for the posterior distribution, $p(\mathcal{D} | \theta)$ for the likelihood, and $p(\theta)$ for the prior. Given independent and identically distributed (i.i.d.) data, the density function for the likelihood distribution can be written as follows:

$$p(\mathcal{D} | \theta) = p(\mathcal{D}_u | \theta) p(\mathcal{D}_f | \theta) p(\mathcal{D}_b | \theta) = \prod_{i=1}^{N_u} p(u_i | x_i^u, \theta) \prod_{i=1}^{N_f} p(f_i | x_i^f, \theta) \prod_{i=1}^{N_b} p(b_i | x_i^b, \theta). \quad (5.6)$$

If we further assume that the data for u , f and b follow Gaussian distribution with mean zero and standard deviation, σ_u , σ_f and σ_b , respectively, then correcting it follows that:

$$p(u_i | x_i^u, \theta) = \frac{1}{\sqrt{2\pi}\sigma_u} \exp\left(-\frac{\|u_\theta(x_i) - u_i\|_2^2}{2\sigma_u^2}\right), \quad (5.7a)$$

$$p(f_i | x_i^f, \theta) = \frac{1}{\sqrt{2\pi}\sigma_f} \exp\left(-\frac{\|\mathcal{F}_\lambda[u_\theta](x_i^f) + s_\psi(x_i^f) - f_i\|_2^2}{2\sigma_f^2}\right), \quad (5.7b)$$

$$p(b_i | x_i^b, \theta) = \frac{1}{\sqrt{2\pi}\sigma_b} \exp\left(-\frac{\|\mathcal{B}_\lambda[u_\theta](x_i^b) - b_i\|_2^2}{2\sigma_b^2}\right), \quad (5.7c)$$

We note that σ_u , σ_f and σ_b are often treated as scales of the additive Gaussian noise in measuring data of u , f and b , respectively. Various methods, e.g. Markov Chain Monte Carlo (MCMC) and variational inference, have been proposed to tackle the posterior distribution (5.5). In this work, we choose to use Hamiltonian Monte Carlo (HMC) method to estimate the posterior distribution by sampling θ and ψ from it. We also note that here we only consider the homogeneous noise, while more complicated noise models, e.g. the heteroscedastic noise [234], are also compatible with the framework.

In both the ensemble PINN and B-PINN methods, we obtain samples of the NN

parameters, from which statistics of the prediction are estimated. In general, the mean and the standard deviation are often chosen to represent the prediction and the uncertainty [234], respectively. Specifically in this work, we obtain samples of θ and ψ from ensemble PINNs and/or B-PINNs, denoted by $\{\theta_i\}_{i=1}^M$ and $\{\psi_i\}_{i=1}^M$, respectively, where M denotes the number of samples. Then, the predicted means of the sought solution u and the discrepancy s are estimated as follows:

$$\hat{u}(x) \approx \frac{1}{M} \sum_{i=1}^M u_{\theta_i}(x), \hat{s}(x) \approx \frac{1}{M} \sum_{i=1}^M s_{\psi_i}(x), \quad (5.8)$$

where $\hat{u}$ and $\hat{s}$ denote the mean of u and s , respectively. Similarly, the predicted uncertainties are:

$$Var[u](x) \approx \frac{1}{M} \sum_{i=1}^M (u_{\theta_i}(x) - \hat{u}(x))^2, \quad Var[s](x) \approx \frac{1}{M} \sum_{i=1}^M (s_{\psi_i}(x) - \hat{s}(x))^2, \quad (5.9)$$

where $Var[u]$ and $Var[s]$ denote the variance of u and s , respectively. Interested readers are directed to [234] for a review and [348] for a comprehensive Python library termed NeuralUQ.

We note that upon the computation of $s_{\psi}(x)$, it can be used in various way: (1) its value represents the model discrepancy and its uncertainty represents the uncertainty in the physical model which we refer to as *model uncertainty*; (2) it corrects the misspecified physical models and identifies a system which explains the data; and (3) it could also be used to obtain mathematical expressions for the missing details of certain physical process via the symbolic regression [334, 76, 156]. Specifically in the present work, we employ the symbolic regression method developed in [58] and the associated Python library PySR to regress analytic mathematical expressions for the unknown physical processes. At this step, we utilize the symbolic regression to obtain the relation between s and u (and/or its derivatives depending on specific

problems) in order to obtain a general expression of the governing equations, see Section 5.3.1 for a demonstration example as well as the details of the procedure.

5.3 Results and discussion

In this section, we conduct four numerical experiments, i.e., (1) an ODE system, (2) a one-dimensional (1D) reaction-diffusion equation, (3) a two-dimensional (2D) channel flow, and (4) a 2D cavity flow. In the first two examples, reaction models are possibly misspecified, and in the last two examples, we assume that we misspecify the constitutive relation in non-Newtonian flows as Newtonian ones. In all the test cases, we employ DNNs to correct the misspecified models as discussed in Section 5.2. Details for the computations (e.g., training steps, NN architectures, etc.) can be found in D.1.

5.3.1 Pedagogical example: An ODE system

We first consider the following one-dimensional ODE:

$$\begin{aligned} \frac{du}{dt} &= f(t) + \lambda u(1 - u), \quad t \in [0, 1], \\ u(t = 0) &= u_0, \end{aligned} \tag{5.10}$$

where λ is a non-negative constant, u_0 is the initial condition, and f is the source term. In certain real-world applications, the reaction term, i.e., $\lambda u(1 - u)$ in the right hand side of (5.10), may not be exactly known and herein it is misspecified as an empirical model. In this regard, we study the effect of different prior knowledge about the reaction model in the discovery of governing equation. In particular, the

following cases are considered:

1. Case (A): The reaction term is correctly specified as $\lambda u(1 - u)$, where $\lambda \geq 0$ is unknown.
2. Case (B): The reaction term is misspecified as $\lambda \cos(u)$, where $\lambda \geq 0$ is unknown.
3. Case (C): The reaction term is misspecified as $\lambda \cos(u)$ (with known $\lambda = 0.2$) but we add $s(t)$ to correct it, i.e., $\lambda \cos(u) + s(t)$.

We denote the reaction term as ϕ , and the target here is to identify ϕ given data on u/f as well as differently specified models. In Case (A) and (B), the problem degenerates to identifying the model parameter λ , and ϕ in these two cases can then be computed by $\tilde{\lambda} u_\theta(1 - u_\theta)$ and $\tilde{\lambda} \cos(u_\theta)$, respectively, where $\tilde{\lambda}$ is the PINN estimate of λ . In Case(C), ϕ is computed by $\tilde{\lambda} \cos(u_\theta) + s_\psi(t)$ where s_ψ is the estimation of the discrepancy induced by the model misspecification. We note that in Case (C) we assume λ is known and set $\lambda = 0.2$ for simplicity. This is because the discrepancy is already modeled by $s(t)$ and hence treating λ as unknown does not make any difference in correcting the misspecified model.

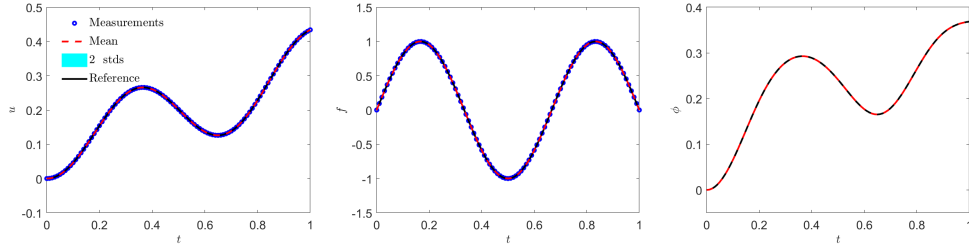
To reduce the effect of the noisy/gappy data on the predicted accuracy, we first test the case with clean and sufficient data. The training data are generated by solving (5.10) with $f(t) = \sin(3\pi t)$ and $u_0 = 0$ using *MATLAB ode45* [132]. We assume that we have 101 data for u and f , uniformly sampled from $t \in [0, 1]$, and we employ ensemble PINNs [348, 234] to learn the governing equation and also quantify the uncertainties in predictions. In all the test cases here, we choose a DNN with two hidden layers, each of which has 50 neurons and is equipped with hyperbolic tangent as the activation function, to approximate u . The equations are

then encoded in PINNs via the automatic differentiation [244]. Further, we utilize an additional DNN to correct the model misspecification in Case (C), which has the same architecture as the one used for approximating u . The Adam optimizer [152] is applied to train the ensemble PINNs.

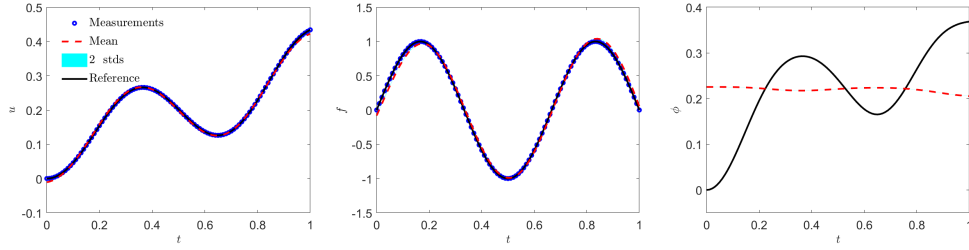
	$\tilde{\lambda}$	Error of ϕ	Error of u	Error of f	Error of $\tilde{u}$
Case (A)	1.5000 ± 0.0001	0.00%	0.00%	0.06%	0.01%
Case (B)	0.2254 ± 0.0001	41.46%	1.73%	6.87%	4.58%
Case (C)	0.2	0.55%	0.01%	0.10%	0.02%

Table 5.1: ODE system: Predictions for ϕ , u and f and the reconstruction $\tilde{u}$. The metric is relative L_2 error, and errors are computed between the reference and the mean of 20 PINNs trained independently. The exact value of λ is 1.5 and $\tilde{\lambda}$ denotes the PINN estimate of λ ($\tilde{\lambda} = 0.2$ is fixed in Case (C)). We note that $\tilde{u}$ is reconstructed by solving the identified ODE system with the identified ϕ , the inferred f and the initial condition u_0 . Cubic spline data interpolation is employed to approximate the function f from the prediction of f on a uniform grid.

As shown in Figure 5.4(a), PINNs with correct physical model (Case (A)) enable accurate inference of ϕ and the surrogate model u_θ fits the data and satisfies the ODE simultaneously. In contrast, PINNs with misspecified physical model (Case (B)) yield completely wrong estimate of ϕ as displayed in Figure 5.4(b). In Case (C), we add a DNN to correct the misspecified reaction model used in Case (B). As illustrated in Figure 5.4(c) as well as Table 5.1, the predicted accuracy for ϕ improves significantly as compared to the result in Case (B). In addition, the computational accuracy for u , f and ϕ in Case (C) is quite similar as in the Case (A), where the physical model is correctly specified. We also conduct the experiment in which λ is fixed at a different value (0.1 rather than 0.2) and the discrepancy is modeled with a DNN. As presented in Figure 5.4(d), the predicted means and uncertainties of u , f and ϕ are similar to the ones from $\lambda = 0.2$. The above results demonstrate the effectiveness of the present approach. We note that the predicted uncertainties in all cases are quite small, which is reasonable since we use sufficient data for both u and f to determine the unknown reaction term here. More results and the related



(a) PINN with the correct physical model.



(b) PINN with a misspecified physical model.

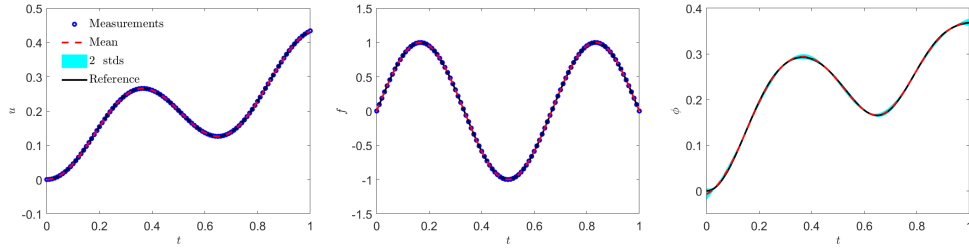
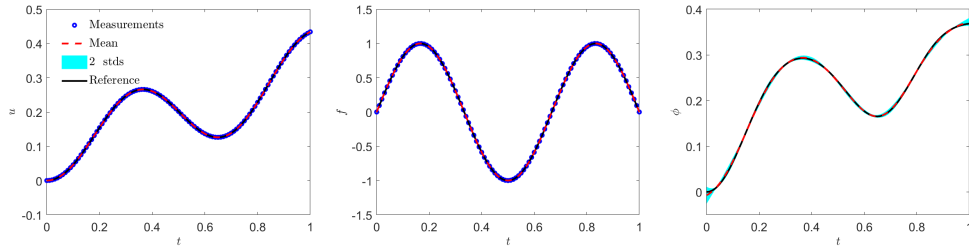
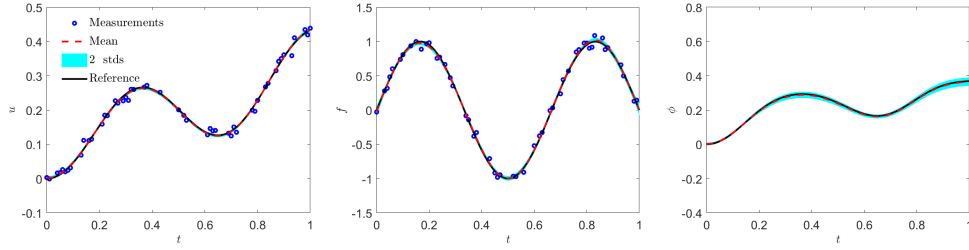
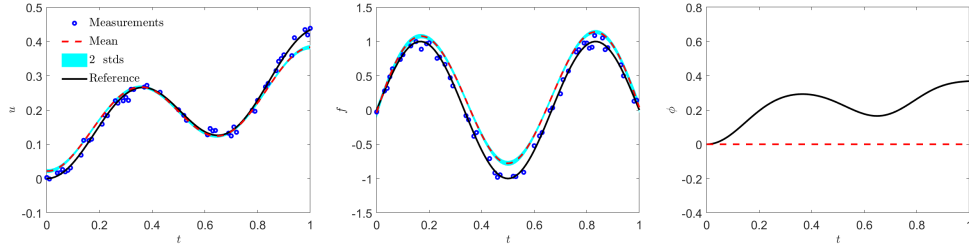
(c) Correcting the misspecified physical model in PINNs using an NN with fixed $\lambda = 0.2$.(d) Correcting the misspecified physical model in PINNs using an NN with fixed $\lambda = 0.1$.

Figure 5.4: ODE system: Predictions for u , f and ϕ from ensemble PINNs with clean and sufficient data. Ensemble PINNs with (a) the correct physical model, (b) a misspecified physical model, and (c) the misspecified physical model plus a correction. Blue circles: Representative measurements for u and f ; Red dashed line: Predicted mean; Black solid line: Reference solution.

discussion for the case with clean but gappy data can be found in the appendix of [346].



(a) B-PINN with the correct physical model.



(b) B-PINN with a misspecified physical model.

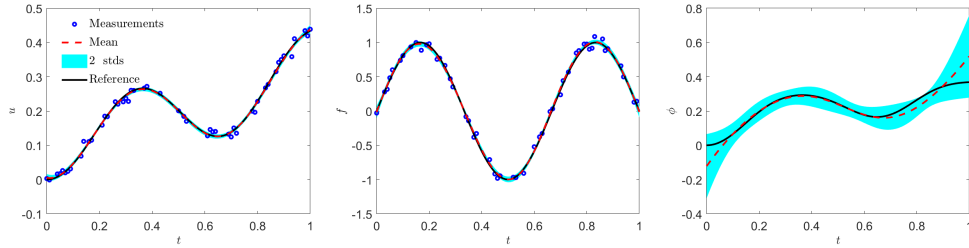
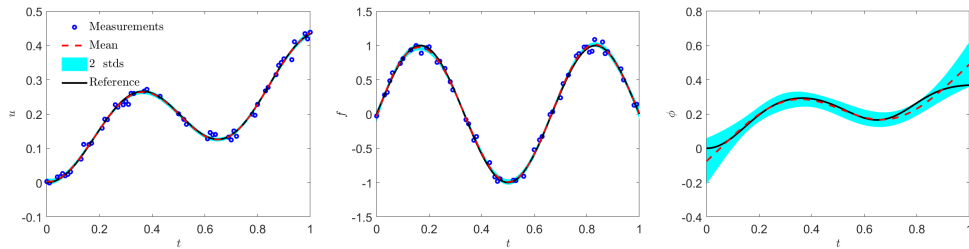
(c) Correcting the misspecified physical model with fixed $\lambda = 0.2$.(d) Correcting the misspecified physical model with fixed $\lambda = 0.1$.

Figure 5.5: ODE system: Predictions for u , f and ϕ from B-PINNs with noisy and gappy data. B-PINNs with (a) correct physical model, (b) misspecified physical model, and (c) misspecified physical model plus a correction. Blue circles: Noisy measurements for u and f ; Red dashed line: Predicted mean from B-PINNs; Black solid line: Reference solution. Our approach alleviates the model misspecification issue significantly, providing more accurate predictions. Compared to correctly specified model, the increase in the predicted uncertainty is caused by the lack of knowledge in the physical model and could be considered as model uncertainty.

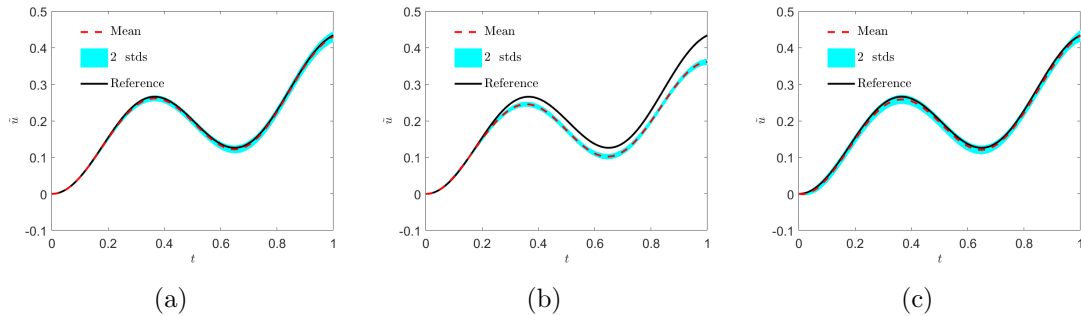


Figure 5.6: ODE system: Solving the identified system with the inferred f and ϕ from B-PINNs with the initial condition $u_0 = 0$. The inferred f and ϕ come from (a) correct physical model, (b) misspecified physical model, and (c) our approach. Red dashed line: Predicted mean; Black solid line: Reference solution. It shows that the misspecified physical model leads to wrong reconstruction $\tilde{u}$ while our approach is able to correct it. Due to the lack of knowledge of the model, our approach yields larger uncertainty than the correct physical model.

	$\tilde{\lambda}$	Error of $\tilde{u}$
Case (A): Known model	1.4867 ± 0.0488	1.56%
Case (B): Misspecified model	0.0000 ± 0.0000	12.73%
Case (C): Corrected model	0.2	2.61%

Table 5.2: ODE system: Relative L_2 error for the reconstructed u from B-PINNs. Here the computational errors are calculated using the predicted mean from B-PINNs.

We now keep the same setup as in Case (A) - (C) but assume that the training data are both noisy and gappy. In these cases, 50 data of f and u are randomly sampled from $t \in [0, 1]$ and corrupted by independent additive Gaussian noises with known noise scales (0.05 for f and 0.01 for u). As discussed in Section 5.2, we will employ the B-PINNs to quantify the uncertainties arising from the noise as well as the incompleteness of the training data. As shown in Figure 5.5, (1) B-PINNs with correct physical model can obtain accurate predictions for both u , f , and ϕ , and the computational errors for ϕ are generally bounded by the predicted uncertainties (Case (A), Figure 5.5(a)); (2) B-PINNs with the misspecified physical model fails to provide accurate inference for u , f , or ϕ (Case (B), Figure 5.5(b)); and (3) B-PINNs with a DNN as correction to the misspecified model is able to improve the predicted

accuracy for u , f , and ϕ , as compared to the results in Case (B) (Case (C), Figure 5.5(c)). Similar as in Case (A), the computational errors for ϕ are bounded by the predicted uncertainties here. In Figure 5.5(d), we present the result from fixing λ at a different value (0.1). As shown, the predicted means and uncertainties of u , f and ϕ are similar to the ones from $\lambda = 0.2$.

In addition to the relation between the computational errors and the predicted uncertainties for ϕ discussed above, we would like to further discuss that: (1) it is interesting to observe in Case (A) and Case (C) that the predicted uncertainty near the right end is larger than that near the left end. We note that the results here are reasonable since we do not have data for f on the right end; (2) the predicted uncertainties of ϕ for $t \in [0, 1]$ in Case (C) are larger than those in Case (A) although we use the same training data as well as prior for the BNN that is used to approximate u . We remark that the difference between the uncertainties in these two cases is caused by the lack of knowledge on the physical model and reflects the uncertainty in physical models.

We further verify the learned governing equation from the proposed approach. Specifically, we solve the equation $du/dt = f(t) + \phi(t)$ with the inferred f , ϕ , and the initial condition $u_0 = 0$, using MATLAB ode45 to see if the solution for u agrees with the reference solution. We denote the solution to the identified system by $\tilde{u}$. In particular, 1000 posterior samples of f and ϕ from the B-PINNs are used to compute $\tilde{u}$, as displayed in Figure 5.6. We further illustrate the relative L_2 errors between the mean for $\tilde{u}$ and the reference solution in Table 5.2. As we can see, PINNs with correct physical model leads to the lowest error in $\tilde{u}$. Adding a DNN to correct the misspecified model can significantly improve the accuracy in $\tilde{u}$ as compared to the case with misspecified model, and the computational errors are slightly larger than the ones from the correct physical model. We also present the computational

errors for $\tilde{u}$ from the ensemble PINNs in Table 5.1, which will not be discussed again since the results are similar as the above ones. Finally, we would like to point out that (1) the uncertainties in the reconstructed u (i.e., $\tilde{u}$) are caused by the predicted uncertainties for f and ϕ from B-PINNs, and (2) the computational errors for the mean of $\tilde{u}$ in Case (A) and (C) are of the same order as the noise in the measurements, suggesting the reasonableness of the current results.

As discussed in Section 5.2, the discovered physical models or governing equations up to now are expressed using the trained DNNs (i.e., s_ψ), which are not explicitly presented. We can then combine the proposed method with the symbolic regression as the further step to obtain the explicit governing equations. Here we take the results in Case (C) from both ensemble PINNs and B-PINNs to demonstrate how to get the analytical expressions for the trained NNs, i.e., s_ψ . Recall that with the present methodology the reaction term is now expressed as $\phi(t) = \tilde{\lambda} \cos(u_\theta(t)) + s_\psi(t)$. Assume that we now have the knowledge that the reaction term is autonomous with respect to the solution, i.e. $\phi = s(u)$. We can then adopt the symbolic regression developed in [58] to find the analytical expression of $s(u)$ from u_θ and s_ψ . The data for the symbolic regression are generated as $\{u_\theta(t_i), s_\psi(t_i) + \tilde{\lambda} \cos(u_\theta(t_i))\}_{i=1}^N$, where $t_i, i = 1, \dots, N$ are 101 uniform grids points on $[0, 1]$, and u_θ and s_ψ are the predicted means from the trained ensemble PINNs/B-PINNs. As displayed in Table 5.3, the recovered reaction model via symbolic regression agrees well with the reference solution as we have sufficient clean data. However, for the case with noisy and gappy data, the expression for the reaction model from symbolic regression does not match the reference solution. The result is expected since we can see in Figure 5.5(c) that the predicted mean for ϕ does not agree well with the reference solution well and the predicted uncertainties are generally large. Generally, increasing the number of training data is able to reduce the predicted uncertainty in B-PINNs [234, 348].

Here we also show that the learned reaction model agrees better with the reference solution as we increase the training data from 50 to 101 in B-PINNs. A recommendation here is to use the predicted uncertainty as guidance, i.e., we use the symbolic regression for discovering equation when the predicted uncertainty is small to achieve better accuracy. More results for the symbolic regression can be found in Appendix D.2.

The method described in this section for identifying the mathematical expression of the misspecified physics and the symbolic regression shares some similarity with the one from [52]: both utilize DNNs and automatic differentiation to compute the partial derivatives of the sought solution u . The main difference is that the sparse regression method from [52] requires specification of mathematical models for all possible physical processes as a candidate library, e.g. $\{u, |u|, u^2, \log(|u|), \dots\}$, while the symbolic regression only requires specification of unitary and binary operations [58], which brings flexibility and may avoid the curse of dimensionality.

Reference	Clean and sufficient data	Noisy and gappy data	Noisy and sufficient data
$1.5u(1 - u)$	$1.4980u - 1.4929u^2$	$u/(u + 0.6213)$	$1.3540u - u^2$

Table 5.3: ODE system: Discovered equations from $\phi(u)$ using a symbolic regression method [58] (see Fig D.1 for plots of the discovered models). In all three cases, the symbolic regression is performed on the predicted mean from ensemble PINNs and B-PINNs.

5.3.2 Reaction-diffusion equation

We now consider the following time-dependent PDE that describes a reaction-diffusion system:

$$\begin{aligned}\frac{\partial u}{\partial t} &= D \frac{\partial^2 u}{\partial x^2} + \lambda g(u) + f(x, t), \quad x \in [0, 1], \quad t \in [0, 1], \\ u(x, 0) &= 0.5 \sin^2(\pi x), \quad x \in [0, 1], \\ u(0, t) &= u(1, t) = 0, \quad t \in [0, 1],\end{aligned}\tag{5.11}$$

where $D = 0.01$ denotes the diffusion coefficient, $g(u) = u(1 - u)$ is the reaction model, λ is the reaction rate, and $f(x, t) = 0$ is the source term. Similar as in Section 5.3.1, we test the following three cases:

1. Case (A): The reaction model is correctly specified and the reaction rate $\lambda \geq 0$ is unknown.
2. Case (B): The reaction model is misspecified as u^2 and the reaction rate $\lambda \geq 0$ is unknown.
3. Case (C): The reaction model is misspecified as u^2 and we correct it with a DNN, i.e., $\lambda u^2 + s$ (with known $\lambda = 1$).

We denote the reaction term as ϕ and we would like to identify ϕ from data of u , f and different physical models. In Case (A)/(B)/(C), the estimate of ϕ is $\tilde{\lambda} u_\theta (1 - u_\theta) / \tilde{\lambda} u_\theta^2 / \tilde{\lambda} u_\theta^2 + s_\psi$, where $\tilde{\lambda}$ is the estimate of λ . We note that in Case (C) we assume $\lambda = 1$ is known because the model misspecification induced by wrong λ can be fully covered by s .

Here we only consider a general scenario, i.e., the training data are noisy and

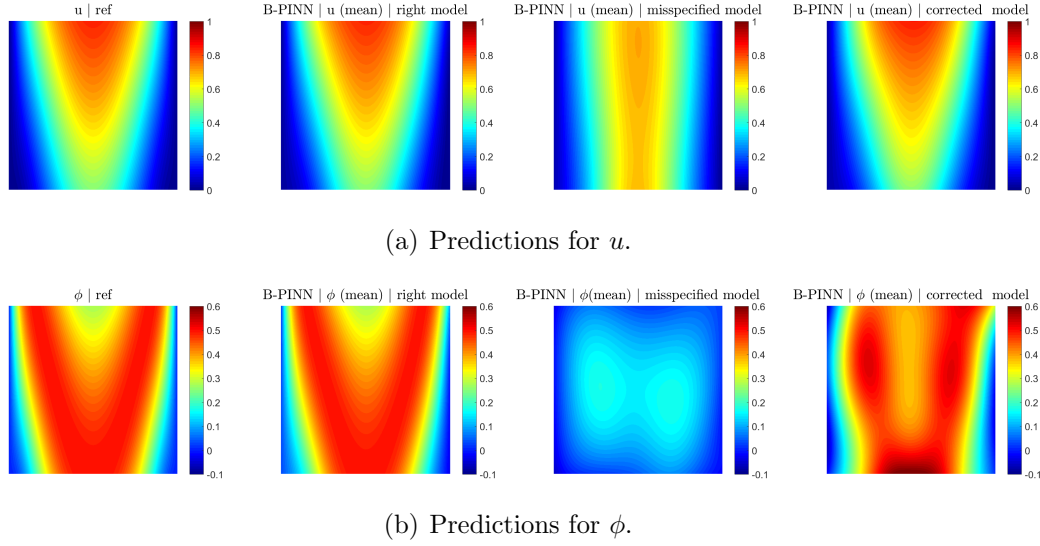


Figure 5.7: Time-dependent reaction-diffusion system: Predicted mean for u and ϕ from B-PINNs with noisy data of u and f and differently specified reaction models. Misspecified model (u^2) impairs the performance of B-PINNs significantly in terms of the accuracy in predicting u and ϕ . and the proposed approach is able to alleviate the issue caused by model misspecification; see Figure 5.8 for UQ.

gappy. In this case, we employ B-PINNs to identify ϕ with uncertainties. We assume that we have 121 measurements on u and 195 measurements of f , which are evenly distributed in the temporal-spatial domain $(x, t) \in [0, 1] \times [0, 1]$. In addition, the measurement errors for both u and f are assumed to be Gaussian, and the noise scales are 0.02 and 0.05 for u and f , respectively. Further, the reference solution is obtained by solving (5.11) with $g(u) = u(1 - u)$, $\lambda = 2$, and $f(x, t) = 0$. Similar as in Section 5.3.1, we use one DNN to model the sought solution u and another DNN to approximate the discrepancy s in Case (C). Details of the B-PINN method, e.g. hyperparameters of NNs and HMC, can be found in Appendix D.1.

As show in Figure 5.7, we observe that: (1) the predicted means for u and ϕ agree the best with the reference solutions as the reaction model is correctly specified (Case (A), second column in Figure 5.7); (2) the predictions for u and ϕ show significant discrepancy as compared to the reference solution when the reaction model

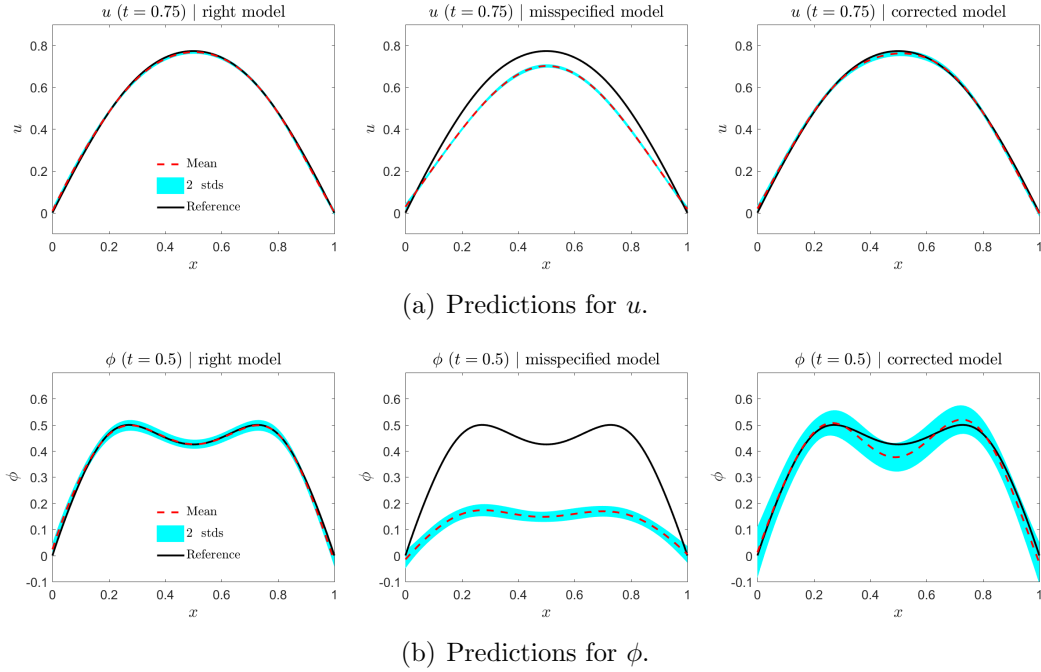


Figure 5.8: Time-dependent reaction-diffusion system in slices ($t = 0.5, 0.75$): Predicted uncertainty for u and ϕ from B-PINNs with noisy data of u and f and differently specified reaction models. Representative predicted uncertainties for (a) u , and (b) ϕ . Red dashed line: Predicted mean from B-PINNs; Black solid line: Reference solution. Our approach is able to improve the the predicted mean and the predicted uncertainty significantly. The decrease in the accuracy of predicted mean and increase in the predicted uncertainty of our approach are caused by the lack of knowledge over the reaction model.

is misspecified (Case (B), third column in Figure 5.7); (3) the predicted mean for u in Case (C) is similar as that in Case (A), in which the reaction model is correctly specified; and (4) the predicted mean for ϕ in Case (C) is not as good as the one in Case (A), but it is much better than the result in Case (B) where the reaction model is misspecified. All the results demonstrate that the added DNNs are able to correct the misspecified model which in turn enhance the computational accuracy in PINNs. A demonstration example for using the symbolic regression to obtain the explicit reaction model is presented in Appendix D.3.

We further illustrate the predicted uncertainties for both u and ϕ in Figure 5.8. In particular, we only depict the uncertainties at one representative time for simplicity.

As we can see, (1) the computational errors for both u and ϕ are generally bounded by the predicted uncertainties in Case (A) (first column in Figure 5.8) and Case (C) (third column in Figure 5.8), while they cannot be bounded by the predicted uncertainties in the case with misspecified reaction model i.e. Case (B) (second column in Figure 5.8), and (2) the predicted uncertainties for u and ϕ are generally larger in Case (C) (third column in Figure 5.8) than those in Case (A) (first column in Figure 5.8), which are similar as the results in Section 5.3.1. Again, we attribute the difference between the predicted uncertainties for Case (A) and (C) to the model uncertainty.

5.3.3 2D non-Newtonian flows

In this section, we test the capability of the present approach in non-Newtonian flows where we misspecify the constitutive relation as the Newtonian one, and then we employ DNNs to correct the misspecified models. In particular, we consider the steady flows in a 2D channel and a 2D cavity, and the employed non-Newtonian fluids are power-law ones. The constitutive relation in a power-law fluid is expressed as $\mu = \mu_0 |\mathbf{S}|^{n-1}$, where μ is the dynamic viscosity of a fluid, μ_0 is a constant, $\mathbf{S} = \nabla \mathbf{u} + (\nabla \mathbf{u})^T$ denotes the shear rate ($|\mathbf{S}| = \sqrt{(\mathbf{S} : \mathbf{S})/2}$), $\mathbf{u}$ is the fluid velocity, and n is the power law index. Without loss of generality, the fluids are shear-thinning (pseudoplastic, $n < 1$) and shear-thickening (dilatant, $n > 1$) in the channel and cavity flows, respectively.

5.3.3.1 Channel flow

The following 2D non-Newtonian channel flow is considered here and the corresponding governing equation is expressed as [295]:

$$\begin{aligned} \frac{\partial}{\partial y}(\mu(y) \frac{\partial u}{\partial y}) - \frac{\partial P}{\partial x} &= f(y), \quad y \in [-H/2, H/2], \\ u(-H/2) &= u(H/2) = 0, \end{aligned} \quad (5.12)$$

where $H = 1$, $\frac{\partial P}{\partial x} = c$ is a constant, and $f(y) = 0$. The analytical solution reads as

$$u(y) = \frac{n}{n+1} \left(-\frac{1}{\mu_0} \frac{\partial P}{\partial x} \right)^{1/n} \left[\left(\frac{H}{2} \right)^{1+1/n} - |y|^{1+1/n} \right]. \quad (5.13)$$

For this specific case, we set $n = 0.25$, $c = -1$, $H = 1$, $\mu_0 = 0.5$ to generate the reference solution as well as training data.

In the first test case, we assume that we have 30 clean measurements of u and 51 clean measurements of f , randomly and uniformly sampled from $[-H/2, H/2]$, respectively. We assume the fluid is a Newtonian one and the viscosity is an unknown constant. We then employ the PINN method to identify the unknown viscosity μ_1 in the following equation:

$$\frac{\partial}{\partial y}(\mu_1 \frac{\partial u}{\partial y}) - \frac{\partial P}{\partial x} = f(y), \quad (5.14)$$

which describes Newtonian flows. Specifically, the boundary condition is hard-encoded in the equation for u so that we only have two terms in the PINN loss function (5.2), i.e., $\mathcal{L}_{\mathcal{D}_u}$ for the data of u and $\mathcal{L}_{PDE}$ for the equation. As shown in Figure 5.2(b), we cannot minimize $\mathcal{L}_{\mathcal{D}_u}$ and $\mathcal{L}_{PDE}$ simultaneously in PINNs, regardless of the choice of belief weights for those two terms, indicating that the NN

surrogate model u_θ is not able to fit the non-Newtonian data and satisfy the Newtonian physical law at the same time.

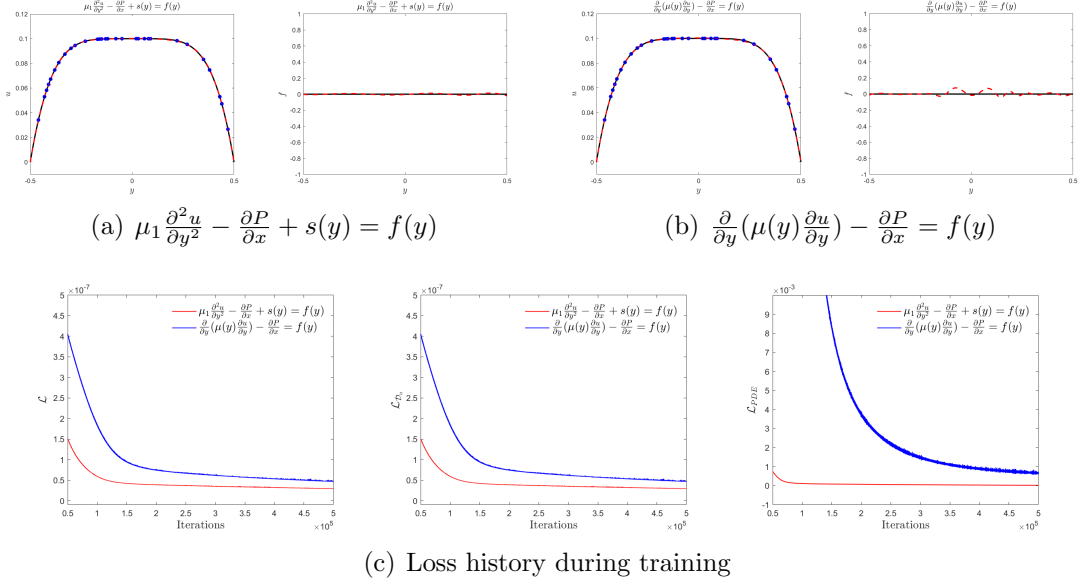


Figure 5.9: 2D non-Newtonian channel flow: Correcting the misspecified physics with two different approaches, i.e. (a) modeling the discrepancy in the equation $s(y)$ with a DNN, and (b) modeling the unknown space-dependent viscosity $\mu(y)$ with a DNN. In (c), the PINN training losses are presented and they converge to smaller values for both $\mathcal{L}_{\mathcal{D}_u}$ and $\mathcal{L}_{PDE}$ in the case where we add a DNN as correction as compared to the one in which we use a DNN to approximate the viscosity.

Similar as in previous two examples in this section, we can add a DNN as the correction in (5.14) to address the model misspecification issue in PINNs. The equation with the correction term can now be expressed as follows:

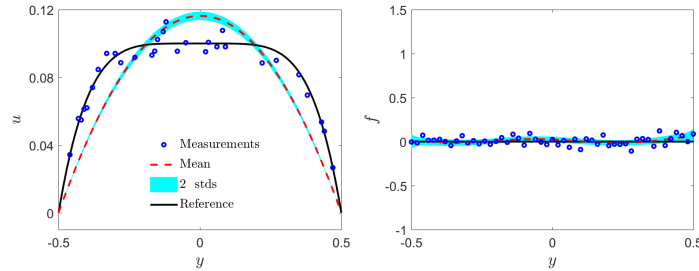
$$\frac{\partial}{\partial y}(\mu_1 \frac{\partial u}{\partial y}) - \frac{\partial P}{\partial x} + s(y) = f(y), \quad (5.15)$$

where s denotes the correction term. Here we treat μ_1 as known and set $\mu_1 = 0.1$ since we have a DNN to model the discrepancy. We present the results from the PINNs with a correction in Figure 5.9(a), and we can see that the current predictions for u and f agree with the reference solutions much better than in the case where the model is misspecified.

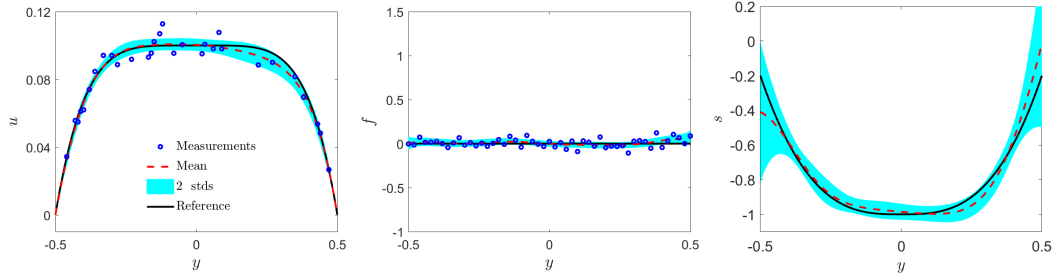
We note that in this specific case we can also employ other approaches to address the issue that we cannot fit the data and satisfy the equation simultaneously in PINNs if the physical model is misspecified. A natural way is to treat the viscosity in (5.14) as an unknown function and employ a DNN to directly learn the constitutive relation. We present the corresponding results in Figure 5.9(b), in which we employ the same setup, e.g., training data, NN architecture in PINNs, as used in Figure 5.2 as well as Figure 5.9(a). As shown, PINNs with a DNN for learning the viscosity now can fit u and f much better than the results in Figure 5.2. We then compare these results with the ones in Figure 5.9(a). As shown, the predictions for u are similar, but the PINN with a DNN as correction is able to provide more accurate predictions for f , i.e., less fluctuations are observed in Figure 5.9(a) than in Figure 5.9(b). Furthermore, we illustrate the loss histories for training PINNs in these two approaches in Figure 5.9(c). As we can see, the training loss decreases faster and converges to smaller values in the case where we add a DNN as correction to the misspecified model, as compared to the approach in which we model the viscosity as an unknown function. We conjecture that the superior performance of the proposed approach can be attributed to the following reason: modeling the viscosity as an unknown function via a DNN requires the accurate estimate of the derivatives in (5.14), which is quite challenging since $n < 1$ results in very sharp gradients for the viscosity around $y = 0$. However, the correction term, i.e., s in (5.14), is a quite smooth function, as shown in Figure 5.10. Hence, we can expect the easier training as well as better accuracy in the proposed method.

We now move to the noisy-data case and employ B-PINNs to quantify the model uncertainty. The same 30 noisy measurements of u and 51 noisy measurements of f are sampled but now they are corrupted by additive Gaussian noise with 0.005 and 0.05 noise scales, respectively. In B-PINNs, we test two specific cases, i.e., (1)

we assume the fluid is Newtonian and encode (5.14) in B-PINNs. The objective is then to infer the unknown viscosity given data, similar as the case in Figure 5.2; and (2) we are not sure if the fluid is Newtonian or not, we then add a DNN as the correction to the possible misspecified model, i.e., (5.15). Also, we note that in this specific case s can be computed analytically by plugging the analytic solution of the power-law fluid into (5.15): $s(y) = 4\mu_1 c^4 |y|^3 / \mu_0^4 + c$, which serves as the reference in what follows.



(a) B-PINN with misspecified model.



(b) Correcting the misspecified model with a DNN in B-PINNs.

Figure 5.10: 2D non-Newtonian channel flow: Predictions for u , f and the discrepancy s from B-PINNs with noisy data. B-PINNs with (a) misspecified model, and (b) misspecified model plus a correction.

The results from B-PINNs are illustrated in Figure 5.10. As observed, the NN surrogate u_θ is not able to fit the data and satisfy the misspecified physical model at the same time (Figure 5.10(a)). In addition, with the additional DNN for correcting the misspecified physical model in the previous case, u_θ can now fit the data and satisfy the corrected physical model simultaneously, as shown in Figure 5.10(b). Also, we can see that the computational errors between the predicted mean and

the reference solution for s are bounded by the predicted uncertainties in the entire computational domain.

5.3.3.2 Cavity flow

Next, we consider steady non-Newtonian flow in a two-dimensional lid-driven square cavity (i.e., $x, y \in [0, 1]$), which can be described by the incompressible Navier-Stokes equations as

$$\nabla \cdot \mathbf{u} = 0, \quad (5.16a)$$

$$\mathbf{u} \cdot \nabla \mathbf{u} = -\nabla P + \nabla \cdot [\nu(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)] + \mathbf{f}, \quad (5.16b)$$

where $\mathbf{u} = (u, v)$ denotes the velocity in x - and y - directions, respectively; P is the pressure; and ν is the kinematic viscosity. In particular, the boundary condition for the upper wall is expressed as

$$u = U \left(1 - \frac{\cosh[r(x - \frac{L}{2})]}{\cosh(\frac{rL}{2})} \right), \quad v = 0, \quad (5.17)$$

where U , r and L are constant. Specifically, $r = 10$, $L = 1$ is the length of the cavity. In addition, the remaining walls are stationary. Here, the viscosity model is the same as the one used in channel flow, i.e., $\mu = \rho\nu = \mu_0|\mathbf{S}|^{n-1}$. The objective is to learn the governing equations, which explain the data given measurements on $\mathbf{u}$, P , and $\mathbf{f}$. In particular, the training data as well as the reference solution in this case are generated by solving (5.16a) using the lattice Boltzmann equation model (LBE) in [295] with $n = 1.5$. Details for the computations in LBE can be found in Appendix D.4.

Similar as in Section 5.3.3.1, suppose we misspecify the governing equations as the ones for Newtonian flows, and we then employ the present approach to correct the model misspecification as follows:

$$\nabla \cdot \mathbf{u} = 0, \quad (5.18a)$$

$$\mathbf{u} \cdot \nabla \mathbf{u} = -\nabla P + \nu_0 \nabla^2 \mathbf{u} + \mathbf{f} + \mathbf{s}, \quad (5.18b)$$

where $\mathbf{s} = (s_x, s_y)$ are the two components correcting the momentum equations in x - and y - directions, respectively, and $f_x(x, y) = f_y(x, y) = 0$ for $x, y \in [0, 1]$. The continuity equation (5.18a) is correctly specified and hence does not need correction.

We assume that we have 400 random measurements of u , 400 random measurements of v , and 81×81 uniformly distributed measurements of P from the reference solution. We evaluate the PDEs at 99×99 uniformly distributed mesh grid (residual points). Here we consider the case with clean data and the ensemble PINNs are employed for quantifying the model uncertainty. The case with noisy data can be found in the appendix of [346]. We would like to mention that we do not use B-PINNs here since we need to utilize around 3.6×10^4 training data to achieve meaningful results, which is computationally expensive as the HMC is used for posterior estimation. Similarly, we test two specific cases, i.e., (1) the viscosity in (5.18b) is an unknown constant and we do not use the correction $\mathbf{s}$ in PINNs, and (2) we have an initial guess for the viscosity in (5.18b) and we add $\mathbf{s}$ in PINNs for the correction for the possible misspecification. Specifically, we use one DNN to model the velocity $\mathbf{u}$, one DNN to approximate the pressure p , and one DNN to model the discrepancy in (5.18b), i.e. $\mathbf{s}$. In particular, we set $\nu_0 = 0.0001$ as the initial guess in Case (2) which will not affect the computational accuracy in PINNs since the model discrepancy is already leveraged by $\mathbf{s}$.

	Error of u	Error of v	Error of f_x	Error of f_y
Misspecified model	7.79%	11.28%	3.9153×10^{-4}	3.1466×10^{-4}
Misspecified model with correction	1.33%	2.29%	7.8389×10^{-7}	8.2283×10^{-7}

Table 5.4: 2D non-Newtonian Cavity flow: Errors of the predicted means from ensemble PINNs. The error of u/v is defined as the relative L_2 error, while the error of f_x/f_y the mean squared error since the reference solutions are $f_x = 0$ and $f_y = 0$.

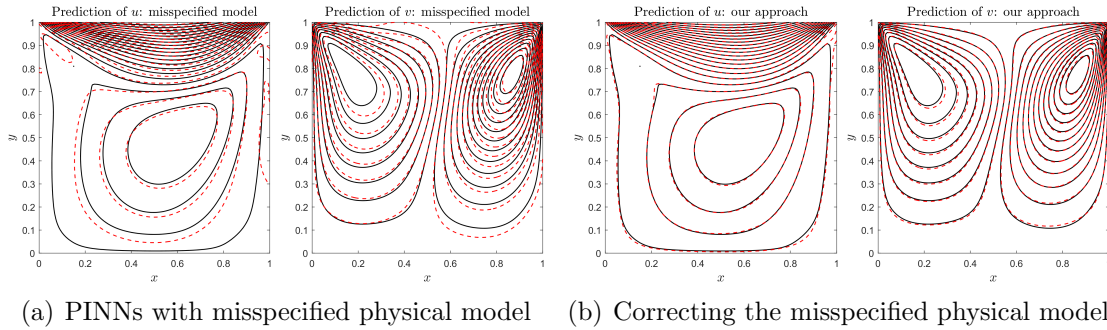


Figure 5.11: Non-Newtonian cavity flow: Predictions from ensemble PINNs for u and v in contours. Red dashed line: Predicted mean; Black solid line: Reference solution.

We present the predictions for $\mathbf{u}$ in Figure 5.11. As we can see, PINNs with misspecified model lead to inaccurate inference for both u and v (Figure 5.11(a)). The proposed approach, however, achieves much better results by adding a DNN to correct the model misspecification. We further illustrate the computational errors for both cases in Table 5.4, which clearly shows the effectiveness as well as superiority of the present method. In addition, we show some representative 1D slices for the predicted u, v, f_x , and f_y with uncertainties in Figure 5.12. Similarly, the PINNs with misspecified model cannot fit the data and satisfy the physical model at the same time (Figure 5.12(a)), while this issue can be handled well as we add a DNN to correct the misspecified model, as shown in Figure 5.12(b).

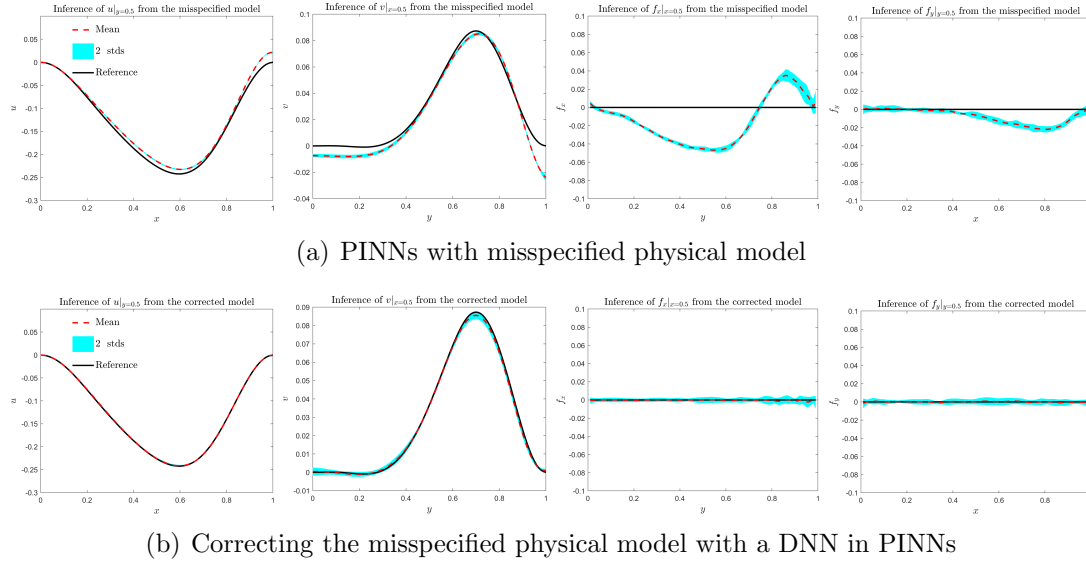


Figure 5.12: Non-Newtonian cavity flow: Predictions for u, v, f_x, f_y in 1D slices. Red dashed line: predicted mean; black solid line: reference solution; uncertainty bound: predicted uncertainty.

5.4 Summary

We presented a general approach to improve the accuracy of physics-informed neural networks (PINNs) for discovering governing equations from data when the physical model is misspecified. In particular, we first assume the mathematical models for the specific problem we considered and encode them in PINNs via automatic differentiation. We note that the assumed models may be misspecified in certain complex systems since not all the physical processes are fully understood here. We then utilize another deep neural networks (DNN) to model the discrepancy between the imperfect models and the observational data in order to correct the misspecified model. In addition, to quantify uncertainties arising from noisy and/or gappy data, we employ two typical uncertainty-induced PINNs methods, i.e., the Bayesian physics-informed neural networks (B-PINNs) and ensemble PINNs. A series of numerical examples are conducted to show the effectiveness of the proposed method. Specifically, we considered (1) a reaction-diffusion system where we misspecified the reaction model,

and (2) non-Newtonian flows in two-dimensional channel and cavity but misspecified the fluid viscosity. The results show that the present approach can achieve good accuracy since the added DNN is able to correct the misspecified physical models encoded in PINNs. Also, the B-PINNs and/or ensemble PINNs can provide reasonable uncertainty bounds for the discovered physical models, which reflects the uncertainties in physical models, i.e., *model uncertainty*. Furthermore, we showcase that we can seamlessly combine the proposed method with the symbolic regression to obtain explicit governing equations based on the trained NNs using examples of both ODE and PDE systems.

In the current study, we only consider model uncertainty or misspecification in the physical model or governing equations. However, there are also cases where both the governing equations and the boundary/initial conditions are misspecified, e.g., flows with slip in a channel. The present method needs to be further extended to fit in the problems with uncertainties arising from multiple sources rather than one. In addition, lots of high-fidelity data are required to obtain accurate correction to the model misspecification in both PINNs and B-PINNs, which may restrict the use of the present method in real-world applications. A possible solution is to integrate the current approach and the multi-fidelity modeling to reduce the amount of expensive high-fidelity data. We will leave these interesting topics as future study. Finally, we expect the proposed approach to be a promising tool in a wider class of applications for solving problems where the physical processes are not fully understood.

CHAPTER SIX

**NeuralUQ: A comprehensive
library for uncertainty
quantification in scientific machine
learning**

6.1 Introduction

Physics-informed and more generally scientific machine learning (SciML) provide potent tools for accurately modeling and predicting the dynamic response of physical systems; see [142, 60, 305, 281] for comprehensive reviews as well as [33, 234]. Neural networks (NNs) for solving ordinary/partial differential equations (ODEs, PDEs), such as the physics-informed NN (PINN), and for learning operator mappings, such as the deep operator network (DeepONet), seamlessly combine scattered observational data with available physical models (e.g. [181, 144, 283, 247, 275]). Specifically, they can address mixed problems when only partial observational data and information regarding the underlying physical system are available (e.g., [245, 33]); it is straightforward to incorporate noisy and multi-fidelity data (e.g., [315, 201, 344, 199, 198]); they can provide physically consistent predictions even for extrapolatory tasks (e.g., [315]); they do not require computationally expensive mesh generation (e.g., [244, 180]); and open-source software is available and continuously extended (e.g., [180, 121, 159, 45, 237, 238, 114, 313]).

However, uncertainty, associated with noisy and limited data as well as NN overparameterization, for example, can degrade the accuracy of these models significantly [2, 234, 282, 91]. As a result, uncertainty quantification (UQ) is paramount for deep learning to be *reliably* used in critical applications involving physical and biological systems [307, 235, 280, 15, 340, 285, 321]. In Figure 6.1, we provide an overview of SciML problems of interest as well as different sources of uncertainty. In addition to reliability concerns, UQ is also necessary for efficient and economical design; for capital budgeting and risk-adaptive decision making in related projects; and even for assessing investment opportunities involving such systems in the form of real options [249, 69]. Clearly, this broad view of applications gives to UQ an additional

dimension pertaining to the potential of *harnessing uncertainty*, and renders it an indispensable research area of SciML applied to computational science and engineering problems and beyond.

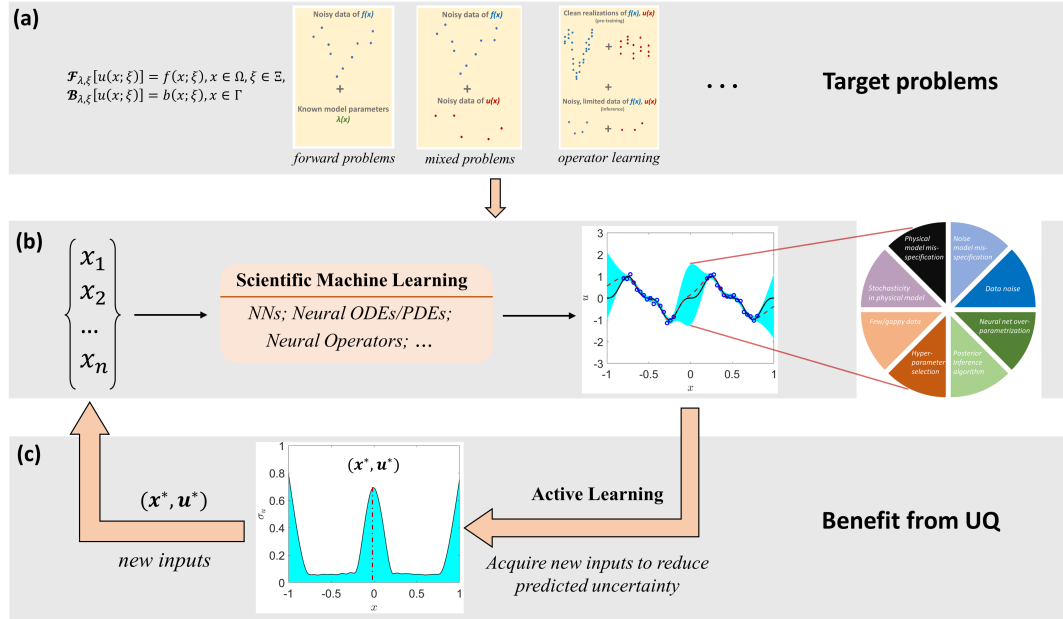


Figure 6.1: Schematic overview of a SciML problem scenario. As shown in (a), SciML problems of interest include, but are not limited to, forward, inverse and mixed ODEs/PDEs, and operator learning problems. Next, in the pie chart of (b) a qualitative breakdown of total uncertainty is shown, describing the contributions from data (noisy, gappy); physical models (misspecification, stochasticity); neural networks (architecture, hyperparameters, overparameterization); and posterior inference. Note that we mainly address two types of uncertainty in this chapter, i.e., aleatoric (data noise) and epistemic (NN overparameterization and limited data) uncertainties. The latter is related to the complex loss landscapes and random initialization of NNs. Addressing physical model or NN architecture uncertainties, indicatively, is left as future work. As shown in (c), active learning for improving the predictions can subsequently be performed, i.e., to acquire new inputs according to an acquisition function that is based on the output uncertainty from part (b).

Further, UQ in SciML is considerably more challenging than in traditional numerical solvers because of optimization-related issues of over-parameterized NNs, as well as the various uncertainties shown in Figure 6.1. In addition, SciML is applied to increasingly more diverse problems, from steady to time-dependent and high-dimensional ones, and to problems involving systems given only in the form

of black-box models (see, e.g., [315, 201, 234, 117, 64]). Moreover, evaluating the performance of the available UQ methods is not straightforward, especially in situations involving limited data. In this regard, Psaros et al. [234] provided an extensive review and proposed novel methods of UQ for SciML. Nevertheless, employing current methods in large-scale applications and developing new ones, where trustworthy implementations and fast experimentation are needed, requires an up-to-date, well-maintained, and versatile software package. To this end, we present in this chapter an open-source Python library termed NeuralUQ, which can be accessed at <https://github.com/Crunch-UQ4MI>. We expect that NeuralUQ will be a useful toolbox for both practitioners and researchers, who either choose to apply off-the-shelf methods for solving their problems, or to build their own methods based on the ones the library currently supports. In passing, we note that NeuralUQ is constantly updated and its most recent capabilities can be viewed in the GitHub repository.

We organize the chapter as follows. In Section 6.2, we briefly present SciML, specifically the PINN and DeepONet methods, as well as recent advances in UQ for SciML, which are the building blocks for the library; however, other neural models for PDEs and operators can be employed as well. In Section 6.3, we describe the design of the NeuralUQ library, which includes model construction, posterior inference, and performance evaluation as well as post-training calibration. In Section 6.4, we provide four computational examples for demonstrating the applicability and capabilities of our library. Lastly, in Section 6.5, we conclude with the findings of this work and discuss future research.

6.2 Uncertainty quantification for scientific machine learning

In this section, we first briefly describe various SciML problem scenarios and the recently developed deep learning methods for solving them. Subsequently, we review the related UQ methods as well as the approaches for model performance evaluation and for post-training calibration. We also provide a glossary table of frequently used terms in this chapter, and in ML/SciML more generally, in Appendix E.1. The interested reader is directed to [234] for more detailed information regarding UQ methods for SciML.

6.2.1 Scientific machine learning

Consider a nonlinear ODE/PDE describing the dynamics of a physical system as follows:

$$\mathcal{F}_{\lambda,\xi}[u(x;\xi)] = f(x;\xi), \quad x \in \Omega, \quad \xi \in \Xi, \quad (6.1a)$$

$$\mathcal{B}_{\lambda,\xi}[u(x;\xi)] = b(x;\xi), \quad x \in \Gamma, \quad (6.1b)$$

where x is the D_x -dimensional space-time coordinate, Ω is a bounded domain with boundary Γ , ξ is a random event in a probability space Ξ , while $f(x;\xi)$ and $u(x;\xi)$ represent the D_u -dimensional source term and sought solution evaluated at (x, ξ) , respectively. Further, $\mathcal{F}_{\lambda,\xi}$ is a general differential operator; $\mathcal{B}_{\lambda,\xi}$ is a boundary/initial condition operator acting on the domain boundary Γ ; and b is a boundary/initial condition term. Furthermore, $\lambda(x;\xi)$, with $x \in \Omega_\lambda \subset \Omega$, denotes the problem parameters. We refer to problems where the operators $\mathcal{F}_{\lambda,\xi}$ and $\mathcal{B}_{\lambda,\xi}$ are specified as

neural ODEs/PDEs, because we use neural networks (NNs) throughout the present work to solve them. We refer to problems involving unknown operators $\mathcal{F}_{\lambda,\xi}$ and $\mathcal{B}_{\lambda,\xi}$ as *neural operators*, because we use NNs to learn them.

Given noisy data of u and λ sampled at random finite locations in Ω and Ω_λ , respectively, as well as noisy data of f and b , the objective of this problem is to obtain the solution $u(x)$ at every $x \in \Omega$ and the problem parameters $\lambda(x)$ at every $x \in \Omega_\lambda$. The dataset in this case is expressed as $\mathcal{D} = \{\mathcal{D}_f, \mathcal{D}_b, \mathcal{D}_u, \mathcal{D}_\lambda\}$, where $\mathcal{D}_u = \{x_i, u_i\}_{i=1}^{N_u}$, $\mathcal{D}_\lambda = \{x_i, \lambda_i\}_{i=1}^{N_\lambda}$, $u_i = u(x_i)$, $\lambda_i = \lambda(x_i)$, and $\mathcal{D}_f, \mathcal{D}_b$ are given as above. The PINN method, developed in [244], addresses the PDE problems deterministically, by constructing NN approximators, parameterized by θ , for modeling the solution $u(x)$. The approximator $u_\theta(x)$ is substituted into (6.1) via automatic differentiation [1] for producing $f_\theta = \mathcal{F}_{\lambda,\xi}[u_\theta]$ and $b_\theta = \mathcal{B}_{\lambda,\xi}[u_\theta]$. For solving the mixed deterministic PDE problem, an additional NN can be constructed for modeling $\lambda(x)$. Finally, the optimization problem for obtaining θ in the mixed problem scenario with the dataset $\mathcal{D} = \{\mathcal{D}_f, \mathcal{D}_b, \mathcal{D}_u, \mathcal{D}_\lambda\}$ is cast as

$$\begin{aligned} \hat{\theta} = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(\theta), \text{ where } \mathcal{L}(\theta) = & \frac{w_f}{N_f} \sum_{i=1}^{N_f} \|f_\theta(x_i) - f_i\|_2^2 \\ & + \frac{w_b}{N_b} \sum_{i=1}^{N_b} \|b_\theta(x_i) - b_i\|_2^2 + \frac{w_u}{N_u} \sum_{i=1}^{N_u} \|u_\theta(x_i) - u_i\|_2^2 + \frac{w_\lambda}{N_\lambda} \sum_{i=1}^{N_\lambda} \|\lambda_\theta(x_i) - \lambda_i\|_2^2. \end{aligned} \quad (6.2)$$

In this equation, $\{w_f, w_b, w_u, w_\lambda\}$ are objective function weights for balancing the various terms in (6.2). Following determination of $\hat{\theta}$, $u_{\hat{\theta}}$ and $\lambda_{\hat{\theta}}$ can be evaluated at any x in Ω, Ω_λ , respectively. Overall, for neural ODEs/PDEs, two different scenarios are considered in the present work through (6.2): (1) *Forward problems*, where $\mathcal{F}_{\lambda,\xi}$, $\mathcal{B}_{\lambda,\xi}$, and the problem parameters $\lambda(x)$ are specified and deterministic. That is, ξ is fixed, whereas $u(x)$ can either be unknown or partially known. (2) *Inverse (or Mixed) problems*, in which $\mathcal{F}_{\lambda,\xi}$ and $\mathcal{B}_{\lambda,\xi}$ are specified and deterministic. That is, ξ is

fixed, $u(x)$ is partially known, and the problem parameters $\lambda(x)$ are either unknown or partially known.

For neural operators, $\mathcal{F}_{\lambda,\xi}$, $\mathcal{B}_{\lambda,\xi}$ are unknown and stochastic, while $\lambda(x; \xi)$, $u(x; \xi)$ are partially unknown. Note that the stochasticity of $\mathcal{F}_{\lambda,\xi}$ and $\mathcal{B}_{\lambda,\xi}$ arises from the stochasticity in $\lambda(x; \xi)$ with $\xi \in \Xi$. The available data consists of paired realizations of f, b, u , and λ , each pair associated with a different $\xi \in \Xi$. The realizations are considered noisy in general for u , and clean for f, b , and λ . Further, the objective of this problem is to learn the operator mapping defined by (6.1). The DeepONet method, developed in [178] and designed for general nonlinear operator learning, addresses this problem by constructing an approximator using NNs, denoted by G_θ and parameterized by θ , to model the mapping from the input function domain to the output function domain. In the following and without loss of generality, we consider both f and b in (6.1) as known and the problem becomes learning the mapping from the parameter λ to the solution u . Further, each value of $\xi \in \Xi$ in (6.1) corresponds to a different $\lambda(x; \xi)$ function, and thus to a different solution $u(x; \xi)$. For any $\xi \in \Xi$, the approximator G_θ takes as input a representation of $\lambda(x; \xi)$, $x \in \Omega_\lambda$, and outputs a function, $G_\theta(\lambda)(x; \xi)$, $x \in \Omega$. The DeepONet consists of two sub-networks, one that takes the coordinates $x \in \Omega$ as input, called trunk net, and one that takes the discretized function λ as input, called branch net. Considering a one-dimensional u , i.e., $D_u = 1$, each of the two networks produces a w -dimensional output. The two outputs are merged by computing their inner product for producing the final prediction $G_\theta(\lambda)(x)$, $x \in \Omega$. For learning the mapping from λ to u using the vanilla DeepONet [179], the minimization problem with the mean squared error (MSE) loss

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(\theta), \text{ where } \mathcal{L}(\theta) = \frac{1}{NN_u} \sum_{i=1}^N w_i \sum_{j=1}^{N_u} \|G_\theta(\lambda(\cdot; \xi_i))(x_j) - u_j^{(i)}\|_2^2 \quad (6.3)$$

is solved, using the training dataset $\mathcal{D}' = \{x_j, \xi_i, \lambda^{(i)}, u_j^{(i)}\}_{i,j=1,1}^{N,N_u}$, where $\{x_j\}_{j=1}^{N_u}$ are the locations, $\{\xi_i\}_{i=1}^N$ are the sampled events and in this case represent the unique pairing between u and λ , $u_j^{(i)} := u(x_j, \xi_i)$ is the measurement of u at the location x_j for event ξ_i , $\lambda^{(i)}$ is the representation of $\lambda(x; \xi_i)$, and $\{w_i\}_{i=1}^N$ are belief weights for different realizations [299]. In this work, $w_i = 1, \forall i \in \{1, \dots, N\}$. Further, to be consistent with neural ODEs/PDEs, we use u_θ to denote the approximator for operators in the rest of this chapter. In this case, $u_\theta = u_\theta(x; \lambda(\cdot; \xi))$. DeepONets, once trained, could act as PDE-agnostic surrogates encoding the hidden physics for further learning and inferences [201]. Finally, note that in addition to supervised learning using data as described above, DeepONet can alternatively be trained by employing only the PINN loss during training. As shown in [107], the most effective training is the hybrid one, where both physics and data are used in the training loss; see also [298].

6.2.2 Uncertainty quantification

In contrast to the point estimates obtained via the optimization problems of (6.2) and (6.3), the goal of UQ is to estimate the predictive distribution

$$p(u|x, \mathcal{D}) = \int p(u|x, \theta) p(\theta|\mathcal{D}) d\theta, \quad (6.4)$$

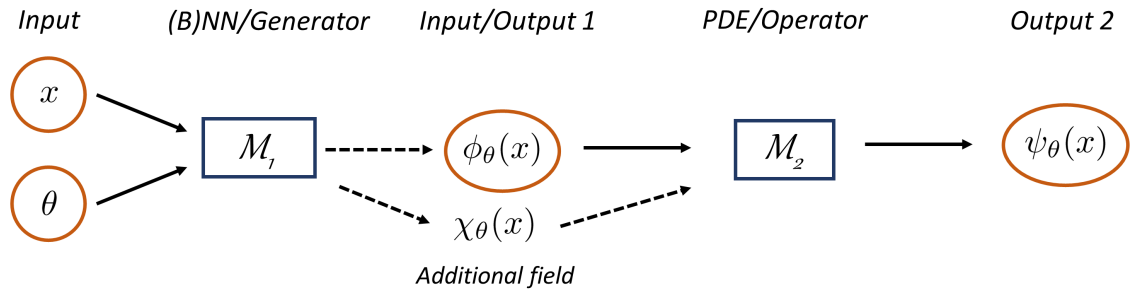
which is often referred to as Bayesian model average (BMA). In (6.4), $p(\theta|\mathcal{D})$ denotes the *posterior density*, which can be obtained using Bayes' rule via

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}. \quad (6.5)$$

In (6.5), $p(\mathcal{D}|\theta)$ is the likelihood of the data, i.e., $p(\mathcal{D}|\theta) = \prod_{i=1}^N p(u_i|x_i, \theta)$ for independent and identically distributed (i.i.d.) data; $p(\theta)$ is the probability density function for the prior distribution of the parameters θ ; and $p(\mathcal{D})$ is called *marginal likelihood* or *evidence*, because it represents the probability that we observe $\mathcal{D}$. The evidence is given as

$$p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)d\theta. \quad (6.6)$$

The uncertainty of parameters θ is typically referred to as *epistemic uncertainty*.



Unified model

Figure 6.2: A unified view of the UQ models studied in Section 6.2.2; adopted from [234]. The model $\mathcal{M}_1$ can be a NN or a generator network, or nothing. The model $\mathcal{M}_2$ can be a PDE or an operator, such as a pretrained DeepONet. The quantities ϕ, ψ can be the solution u and the source f in (6.1), or vice-versa; x represents the space-time input domain; and θ represents the parameters of the NN or the input to the generator. Further, “ χ ” is used for modeling additional fields and can represent, for example, the model parameters λ in (6.1). The dashed arrow indicates that the connection may not exist for some problem scenarios.

Generally, inferring θ given $\mathcal{D}$, i.e., computing the posterior exactly via (6.5) is computationally and analytically intractable. This is addressed by approximate posterior inference, which approximates the posterior by another distribution and/or obtains representative samples from the posterior. Specifically, the Bayesian and the deterministic frameworks are the two main families of UQ methods for SciML. The former employs either a sampling method or a variational inference (VI) method to obtain samples from the posterior distribution, whereas the latter can be interpreted as an extension of standard training of deterministic NNs. Various UQ methods

have been developed for SciML within these two frameworks. We adopt the unified framework proposed in [234] to include all the UQ models employed in this study in Figure 6.2, and present a short overview for them in the rest of this section.

For neural ODEs/PDEs, the solution u and/or the problem parameters λ are modeled with *surrogates* parameterized by θ and denoted by $\mathcal{M}_1$ in Figure 6.2. The Bayesian PINNs (B-PINNs) [319] and the functional prior (FP) [201] based on the physics-informed generative models [317, 321] are two recently proposed physics-informed learning approaches with UQ within the Bayesian framework. Specifically, the Bayesian neural networks (BNNs) and the generators from trained physics-informed generative adversarial networks (PI-GANs) [317] are employed as the surrogate for u/λ in these two methods, respectively. Note that the main difference between BNNs/B-PINNs and the functional prior lies in the prior in function space of each model. In BNNs/B-PINNs, an arbitrary prior in function space is implicitly considered, which depends on the NN architecture and the weight/bias prior distributions that are used. However, in the functional prior model, we train GANs/PI-GANs to learn a certain stochastic process in the function space given historical data. Upon completion of the training phase, the output of the generator in GANs/PI-GANs, i.e., $\psi_\theta(x)$ in Figure 6.2, is used to approximate the stochastic process represented by the historical data and acts as the prior in function space for new tasks. Intuitively, $\psi_\theta(x)$ plays a similar role as the Karhunen-Loève (KL) expansion of Gaussian processes, although $\psi_\theta(x)$ is more expressive and can in principle approximate arbitrary stochastic processes. Next, given the prior in function space, in the form of a BNN or functional prior, either sampling or variational inference (VI) approaches can be utilized for obtaining the posterior samples for θ given data on a new task. In addition to the aforementioned methods, the Monte Carlo dropout [89] (MCD) as well as the deep ensemble [163] (DEns) approaches, which employ standard, i.e.,

deterministic, NNs as surrogates for u and/or λ , can be combined with PINNs to quantify uncertainties in problems with known physics [329, 234].

For neural operators, a DeepONet parameterized by θ is employed as the surrogate for the solution u and denoted by $\mathcal{M}_2$ in Figure 6.2. In the Bayesian framework, [173] employed BNNs in DeepONet and a sampling method is then utilized to obtain the posterior samples for the parameters in BNNs. As for deterministic methods, extensions of standard training of DeepONets, such as DEns, are used for quantifying uncertainties in predictions [234, 320, 230]. These models correspond to Figure 6.2 with $\mathcal{M}_2$ being a DeepONet and the dashed arrows are not used. Further, the inference input in these models, i.e. the input to the branch net, should be represented in the same way as used in the training, i.e., the number of the discrete values as well as their locations should be the same. For a more general case in which the inference input and/or output data are noisy and measured at a certain number of random locations, [201] proposed to combine a pretrained DeepONet with a GAN-based FP to quantify the uncertainties in predictions. In [234], a BNN is utilized as an alternative to the generator of a GAN in [201]. These two models correspond to Figure 6.2 with $\mathcal{M}_1$ being either a BNN or the generator from a pretrained GAN and $\mathcal{M}_2$ a pretrained DeepONet.

In all cases, different θ values correspond to different functions u_θ . We refer to the surrogate combined with a distribution $p(\theta)$ for θ as a *process*. Following approximate posterior inference, the posterior samples for θ are used for obtaining the statistics of the function samples. In this regard, the value of u at a location x given the data $\mathcal{D}$ is a random variable denoted by $(u|x, \mathcal{D})$. The mean of $(u|x, \mathcal{D})$ is

denoted by $\hat{u}(x) = \mathbb{E}[u|x, \mathcal{D}]$ and approximated by $\bar{\mu}(x)$ as

$$\hat{u}(x) \approx \bar{\mu}(x) = \frac{1}{M} \sum_{j=1}^M u_{\hat{\theta}_j}(x), \quad (6.7)$$

where $\{u_{\hat{\theta}_j}(x)\}_{j=1}^M$ is the set of NN predictions corresponding to the samples $\{\hat{\theta}_j\}_{j=1}^M$. Further, the diagonal part of the covariance matrix of $(u|x, \mathcal{D})$ is given by

$$Var(u|x, \mathcal{D}) \approx \bar{\sigma}^2(x) = \underbrace{\Sigma_u^2}_{\bar{\sigma}_a^2(x)} + \underbrace{\frac{1}{M} \sum_{j=1}^M (u_{\hat{\theta}_j}(x) - \bar{\mu}(x))^2}_{\bar{\sigma}_e^2(x)}. \quad (6.8)$$

In this equation, $\bar{\sigma}^2(x)$ represents the approximate total uncertainty of $(u|x, \mathcal{D})$, while $\bar{\sigma}_a^2(x)$ and $\bar{\sigma}_e^2(x)$ denote the aleatoric and the epistemic parts of the total uncertainty, respectively. In practice, a Gaussian (or, e.g., a Student-t) distribution is commonly fitted to the posterior samples. For example, we denote as $\mathcal{N}(\bar{\mu}(x), \bar{\sigma}^2(x))$ the Gaussian approximation to the samples $\{u_{\hat{\theta}_j}(x)\}_{j=1}^M$ at an arbitrary location x , where $\bar{\mu}(x)$ and $\bar{\sigma}^2(x)$ are obtained via (6.7) and (6.8), respectively.

6.2.3 Performance evaluation and post-training calibration

Performance evaluation is necessary for model selection, e.g., prior/architecture optimization, comparison between UQ methods, and overall quality evaluation of the UQ design procedure [231]. As reviewed in [234], performance evaluation metrics include, but are not limited to, the relative ℓ_2 error for evaluating accuracy, the mean predictive likelihood for evaluating predictive capacity, and the root mean squared calibration error for evaluating statistical consistency; i.e., how closely the predicted model matches the data-generating distribution.

Further, calibration approaches can be adopted to improve statistical consistency in UQ for SciML [234]. Various approaches have been developed for calibration, e.g., [170] re-weights the predicted variance optimally to calibrate the output uncertainty; [161] approximates the mis-calibration function and applies it to all predictive distributions, and thus modifies both the output mean and variance; and [328] fits an empirical distribution to the scaled residuals, and also calibrates both the output mean and variance. To use these approaches, an additional dataset is required after training. This can be a left-out calibration dataset, similar to validation and test sets.

6.3 Overview and details of NeuralUQ

In this section, we first present an overview of the NeuralUQ library, and subsequently provide details regarding its design, use, and also customization for extended capabilities.

6.3.1 Library overview

As mentioned in the previous section, the value of u at a location x given the data $\mathcal{D}$ is a random variable denoted by $(u|x, \mathcal{D})$. The aim of UQ in the library is to obtain, calibrate, compute the statistics, and evaluate the quality of samples drawn from the posterior distribution $p(u|\mathcal{D})$. To that end, in the design phase a model is constructed that includes surrogates representing the solution u and potentially f and/or λ , as well as likelihood functions representing the data distribution. Surrogates are parameterized by θ and when combined with probability distributions of θ

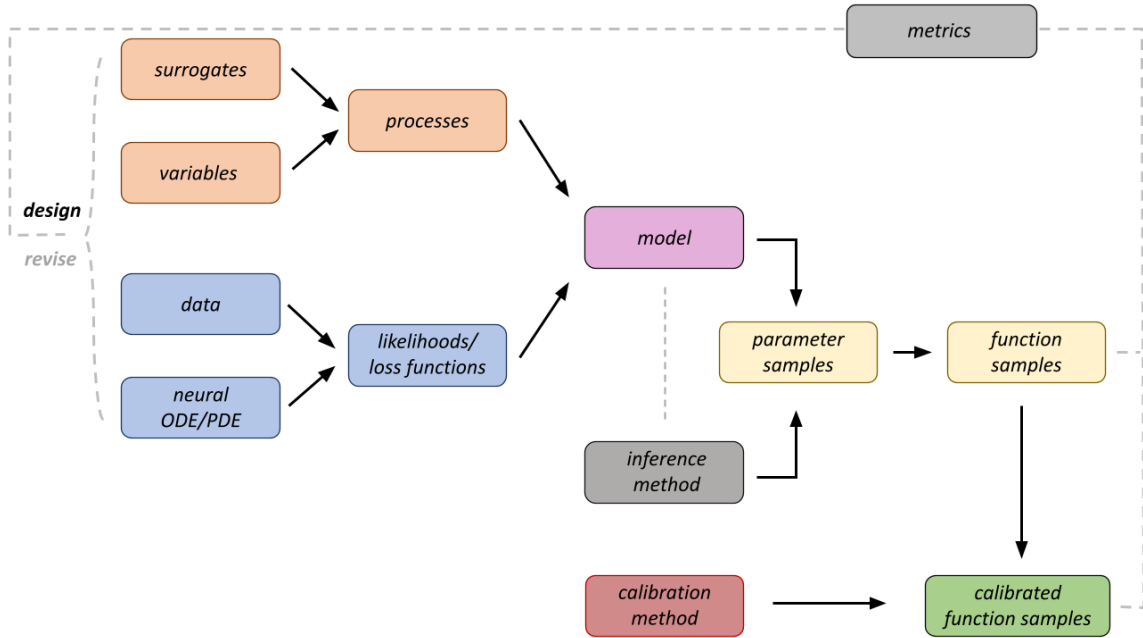


Figure 6.3: Library design overview. In the design phase, a model is constructed that includes processes representing a proposed solution family and likelihoods or loss functions representing the data distribution. Processes include surrogates, which can be construed as assumed functional forms, and priors distributions or variables for the parameters of the surrogates, depending on the type of method used. Likelihoods or loss functions are constructed using the data and the underlying neural ODE/PDE. To select the solution function samples that best explain the data, a posterior inference method is combined with the model. Finally, the obtained function samples can be calibrated using additional data. In the revision phase, metrics are used for evaluating the performance and re-designing the solution procedure.

are represented as stochastic processes; i.e., a proposed family of solutions. To obtain the function samples for the solutions, a posterior inference method is combined with the model. Finally, the obtained function samples can be calibrated using additional data. In the revision phase, metrics are used for evaluating the performance and re-designing the solution procedure. A schematic workflow of the library is provided in Figure 6.3 and Procedure 1.

Procedure 1: NeuralUQ for solving SciML problems.

- Step 1 Specify a process, which is composed of a surrogate as well as its parameters θ to approximate the solution of an ODE/PDE, using the `surrogates`, `variables`, and `process` modules.
 - Step 2 Specify the differential equations using automatic differentiation.
 - Step 3 Construct the likelihoods or loss functions based on the equations, boundary/initial conditions, and/or given data, using the `likelihoods` module.
 - Step 4 Combine the process and likelihoods/loss functions to construct a `model`.
 - Step 5 Specify an inference method using the `inference` module, and compile the model with it to obtain a sampler for posterior distributions by calling `model.compile`.
 - Step 6 Call `model.run` to run the compiled sampler to obtain posterior samples.
 - Step 7 Compute predictions with uncertainties based on (6.7) and (6.8) using the posterior samples of θ from Step 6.
 - Step 8 Use the `metrics` module for performance evaluations using the predictions from Step 7.
 - Step 9 Use the `calibrations` module for post-training calibration (if applicable).
-

6.3.2 Library design

As discussed in Section 6.2.2, there are two general UQ frameworks for SciML, namely the Bayesian and the deterministic frameworks. The former relies on Bayes' theorem, and includes sampling methods, such as Monte Carlo Markov Chain (MCMC), and VI methods, such as mean-field variational inference (MFVI). The latter relies on standard NN training, i.e., minimizing/maximizing an objective function, typically using the MSE or the maximum a posteriori (MAP) metric, between measurements and NN predictions.

The procedural differences of the aforementioned methods necessitate the devel-

UQ methods in NeuralUQ			
UQ framework	UQ family	Process	Inference methods
Bayesian	<i>Samplable</i>	BNNs, Gens	HMC, LD, MALA, NoUTurn
	<i>Variational</i>	BNNs, Gens	MFVI, MCD
Deterministic	<i>Trainable</i>	FNNs, Gens, DeepONets	DEns, SEns, LA

Table 6.1: Summary of the UQ methods implemented in NeuralUQ. NeuralUQ includes three different UQ method families i.e., *Samplable*, *Variational*, and *Trainable*. Each UQ method family has various processes and inference methods, e.g., BNNs+HMC in *Samplable*; BNNs+MFVI *Variational*; and FNN+DEns in *Trainable*. Gens: Generators in GANs. Short and detailed descriptions for these inference methods are provided in Table 6.2 and [234, 252, 311, 125, 23], respectively. Note that BNNs/Generators/FNNs combined with known ODEs/PDEs are used for UQ for neural ODE/PDE problems, while BNNs/Generators combined with pretrained DeepONets and DeepONets using DEns are used for UQ in neural operator problems.

opment of three different UQ method families in our library, i.e., the *Samplable*, *Variational*, and *Trainable* to include all the sampling-, VI- and standard-NN-based methods, respectively. Specifically, the sampling methods target directly the unnormalized posterior density function, without employing and optimizing any trainable variables. The VI-based methods employ pre-defined distributions parameterized by trainable variables, which are optimized to approximate the true posterior distributions. Finally, the standard-NN-based UQ methods obtain point estimates for the parameters in the surrogates using an optimizer, rather than directly sampling from or approximating the posterior distributions. A summary of the currently supported UQ methods in NeuralUQ is provided in Table 6.1.

6.3.3 Details of NeuralUQ components

In this section, we describe the main components, built-in functionalities and how to use NeuralUQ.

6.3.3.1 Processes

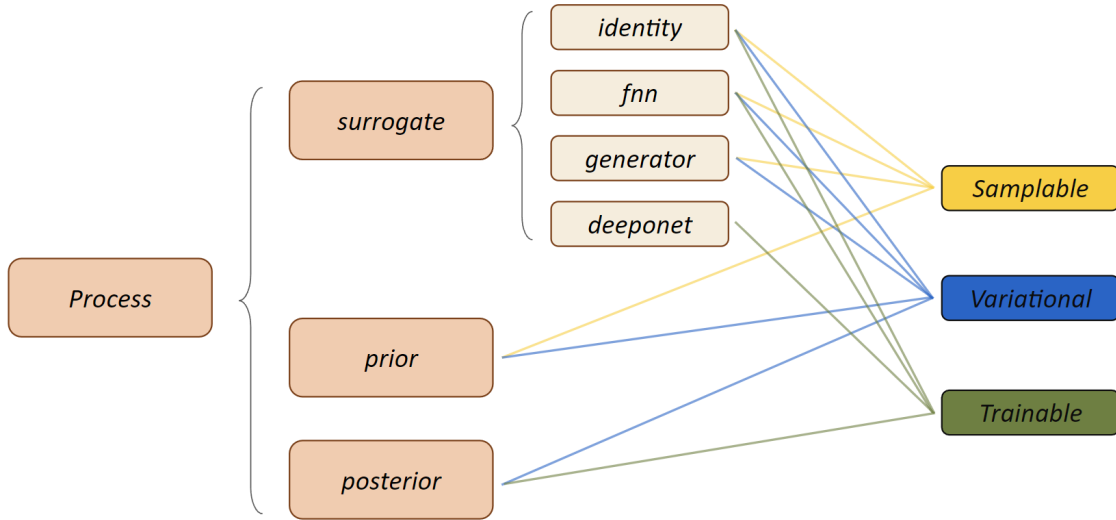


Figure 6.4: A breakdown of the **Process** module. A **Process** instance contains a surrogate as well as the attributes for its parameters. Processes, used in different UQ method families, have different surrogates and attributes for their parameters, which are denoted with different line colors in this figure. Different combinations of the components yield different processes, e.g., an **fnn** surrogate combined with a prescribed prior distribution for its parameters results in a BNN.

As described in Procedure 1, the first step for using NeuralUQ is to specify a process for the sought solution, e.g., BNNs or pretrained generators, according to the task at hand. In this regard, the **surrogates**, **variables**, and **Process** modules have been developed in NeuralUQ. Specifically, the **surrogates** module expresses how x and θ are combined to yield the approximators $u_\theta(x)$ or $\lambda_\theta(x)$ ($\mathcal{M}_1$ or $\mathcal{M}_2$ in Figure 6.2); the **variables** module is used to specify the attributes of the parameters in surrogates, e.g., prior/posterior distributions (Figure 6.4); and the **Process** module is an interface for the implementation of the unified models in Figure 6.2, which connects the surrogate with its associated parameters.

Currently, NeuralUQ provides four built-in surrogates, namely, `surrogates.identity`, used as a surrogate for an unknown constant; `fnn`, for a fully-connected neural net-

work (FNN); `generator`, for the generator of a PI-GAN pretrained with historical data; and `deeponet`, for a DeepONet. The `variables` module, on the other hand, is used to specify the prior and/or posterior distributions of surrogate parameters. For instance, the prior distribution is required for surrogate parameters in the Samplable family, while both the prior and the posterior distributions are required for parameters in the Variational family. The built-in variables of NeuralUQ are `variables.const`, used for parameters of unknown constants as well as the generators of pretrained GANs/PI-GANs; and `fnn`, used for FNNs and also DeepONets. Further, as presented in Figure 6.4 (also in Table 6.1 and Section 6.3.2), different UQ method families support different `surrogates` and are compatible with different `variables`. In NeuralUQ, `variables.fnn` and `variables.const` are decomposed into three different sub-modules, each one pertaining to a UQ method family of Table 6.1, i.e., `variables.fnn/const.Samplable`, `Variational`, and `Trainable`.

With the specified surrogate and parameters, we can then instantiate a `Process` object to create a corresponding process. The surrogate and the attributes for its parameters can then be accessed via `Process.surrogate` and `Process.prior/posterior`, respectively. Also, for problems with unknown parameters or multi-physics, more than one process may be employed. The module `Process.key` can be used to trace each process in NeuralUQ.

6.3.3.2 Likelihoods

In the Samplable and Variational families, likelihood distributions for the given data are required to construct the unnormalized posterior distributions of the surrogate parameters. Similarly, a metric, namely, MSE or MAP, is required for methods in the Trainable family. In NeuralUQ, the `likelihoods` module can be used to define the

likelihood distributions in Samplable and Variational, as well as the MSE/MAP metrics in Trainable (i.e., Step 3 in Procedure 1). In addition, the currently supported likelihood for Samplable and Variational is the Gaussian distribution, accessed via `likelihoods.Normal`, and the supported metric for Trainable is the MSE, accessed via `likelihoods.MSE`. Each likelihood receives *inputs*, *targets* and *processes* as input arguments, for specifying the locations of the measurements, the measurements, and the corresponding processes used to model them. For tasks involving PDEs, the argument *pde* to the likelihood, which is a Python callable defined separately (i.e., Step 2 in Procedure 1), is also required.

6.3.3.3 Inferences

Uncertainty in the sought solution and/or unknown problem parameters is associated with the uncertainty of the surrogate parameters. The `inferences` module has been developed to obtain the posterior samples for the surrogate parameters in NeuralUQ (i.e., Step 5 in Procedure 1). As shown in Table 6.1, NeuralUQ provides various built-in inference methods in each of the UQ method families. Specifically, NeuralUQ currently supports `inferences.HMC`, `LD`, `MALA`, and `NoUTurn` in Samplable; `inferences.VI`, `MCD` in Variational; and `inferences.DEns` and `SEns` in Trainable. Short descriptions of these methods can be found in Table 6.2.

6.3.3.4 Performance evaluation and post-training calibration

Performance evaluation, model revision and post-training prediction improvement are crucial for reliably deploying UQ methods in risk-sensitive SciML settings. In NeuralUQ, we have developed the `metrics` module, which currently includes two

Inference methods		
Full name	Abbreviation	Short description
Hamiltonian Monte Carlo [211]	HMC	Simulates Hamiltonian dynamics followed by the Metropolis-Hasting algorithm
Metropolis-adjusted Langevin algorithm [252, 311]	MALA	Simulates Langevin dynamics followed by the Metropolis-Hasting algorithm
Langevin dynamics [211, 301]	LD	One leap-frog step of HMC
NoUTurn [125, 23]	NUTS	A variant of HMC that adaptively chooses the number of steps for the leap-frog integrator
Mean-field variational inference [25]	MFVI	Approximates $p(\theta \mathcal{D})$ by a Gaussian variational distribution with diagonal covariance matrix
Monte Carlo dropout [89]	MCD	Approximates $p(\theta \mathcal{D})$ by a Bernoulli variational distribution with fixed dropout rate
Deep ensemble [163]	DEns	Standard training M times independently with random initializations
Snapshot ensemble [129]	SEns	Standard training with learning rate schedule and snapshots
Laplace approximation [279, 251]	LA	Standard training followed by fitting a Gaussian to approximate $p(\theta \mathcal{D})$

Table 6.2: Overview of the inference methods in NeuralUQ; adopted from [234]. HMC, MALA, LD, NUTS, and MFVI are inference methods corresponding to the Bayesian framework in Table 6.1; and DEns to the deterministic one. Note that MCD can be interpreted and implemented in both frameworks [89]. In the present study, we treat it as a Bayesian method.

different metrics, i.e., `metrics.RL2E/MSE`, and NLL (i.e., Step 8 in Procedure 1). The former serves as an evaluation metric of the predicted mean and computes the relative ℓ_2 error (RL2E) or MSE between the predicted mean and the reference solution, whereas the latter serves as a metric of the predicted uncertainty and computes the negative log likelihood (NLL) [323, 240].

Further, we have developed the `calibrations` module for performing post-training calibration (i.e., Step 9 in Procedure 1). Specifically, this module currently includes one sub-module, i.e., `calibrations.var`, for the implementation of the method proposed in [170]. We present a workflow for calibration in Procedure 2, and ; a demonstration example involving post-training calibration can be found in the GitHub repository of the library. We will keep updating NeuralUQ to support more metrics and calibration methods, e.g., [56], in the future.

Procedure 2: A workflow for calibration using NeuralUQ.

Step 1 Obtain uncalibrated posterior function samples at the locations x where new measurements are available, using the posterior samples of the parameters from Step 6 in Procedure 1.

Step 2 Choose and apply a calibration method from the `calibrations` module, which takes as input the function samples from Step 1 and new measurements.

Step 3 Obtain calibrated function samples.

6.3.4 Customization

The components of NeuralUQ, including surrogates, variables, likelihoods, and inferences, are highly flexible and customizable. In this section, we describe how to configure the library for extending current capabilities and addressing problem scenarios beyond the ones considered in this study.

6.3.4.1 Surrogates

As mentioned in Section 6.3.3.1, NeuralUQ currently supports four surrogates, namely, `identity`, `fnn`, `generator`, and `deeponet`. Users may need to develop and incorporate in the library alternative surrogates. In NeuralUQ, this can be achieved in a straightforward manner by following Procedure 3. Demonstration examples involving customizing new surrogates are presented in Appendix E.2.

6.3.4.2 Variables

As mentioned in Section 6.3.3.1, `variables.fnn/const` is decomposed into three sub-modules for different UQ method families, and can be customized according to

Procedure 3: Customization of a new surrogate, `MySurrogate`, which inherits from `surrogates.Surrogate`.

```
class MySurrogate(Surrogate):
    def __call__(self, inputs, var_list):
        """
        Returns the outputs of the function, defined by samples stored in
        'var_list', to the input 'inputs', and also the reshaped and
        tiled input, for derivative computation.
        """
```

the task at hand. In this section, customization of the variables in the first two families is discussed, as the corresponding variables influence more the performance as compared to the variables in the third one. The built-in prior distribution for `Samplable` and `Variational` variables is the Gaussian distribution. A straightforward customization in this context pertains to modifying the mean and/or the standard deviation to obtain different priors. Indicatively, adding a `Samplable` in `NeuralUQ` variable can be done by following Procedure 4, which includes the setup of the initial values as well as the log probability density function. Another option pertains to selecting alternative parameterized distributions for performing VI. Demonstration examples involving prior customization as well as selecting alternative distributions for performing VI are presented in Appendix E.3.

Procedure 4: Customization of a new `Samplable` variable, `MySamplable`, which inherits from `variables._Samplable`.

```
class MySamplable(_Samplable):
    @property
    def initial_values(self):
        """Returns initial values of the variable."""

    def log_prob(self, samples):
        """Returns the log probability density of 'samples'."""
```

6.3.4.3 Inference methods

As presented in Section 6.3.3.3 and summarized in Table 6.2, NeuralUQ supports various inference methods for each of the UQ method families, i.e., Samplable, Variational, and Trainable. However, depending on the task at hand, users may need to develop in the library new inference methods, e.g., for reducing computational cost. Indicatively, adding a new sampling method can be done by following Procedure 5. In passing, note that all sampling methods included currently in NeuralUQ are implemented using the TensorFlow Probability MCMC python package [166, 73], which is computationally efficient and straightforward to use. For consistency reasons, we recommend that users of NeuralUQ employ this package for adding new sampling methods to the library.

Procedure 5: Customization of a new inference, `MyInference`, which inherits from `inferences.Inference`.

```
class MyInference(Inference):
    def make_sampler(self):
        """Creates and returns a sampler, which will be used in 'sampling'
        method."""

    def sampling(self):
        """Performs the inference and returns samples."""
```

6.4 Demonstration examples

In this section, we provide four computational examples for demonstrating the applicability and capabilities of our library. Specifically, we quantify uncertainty in both neural ODEs/PDEs and neural operators applied to the following problems: (1) a Kraichnan-Orszag system, (2) a susceptible-infected-recovered-dead (SIRD) model

for COVID-19, (3) a Korteweg-de Vries problem, and (4) a 100-dimensional Darcy problem. In the first three cases, we obtain and present results using uncertain PINNs, i.e., B-PINNs combined with HMC and PINNs combined with DEs, as representative examples of the Bayesian and deterministic methods, respectively. For the 100-dimensional Darcy problem, we present results using the DeepONet combined with two physics-agnostic models, i.e., physics-agnostic GAN functional prior (PA-GAN-FP) and physics-agnostic BNN functional prior (PA-BNN-FP), as well as DEs. Results corresponding to additional UQ methods and details of the computations in each example, e.g., architectures of NNs and hyperparameters of the methods, are provided in the GitHub repository of our library.

6.4.1 Kraichnan-Orszag system

The Kraichnan-Orszag model consisting of three coupled nonlinear ODEs describes the temporal evolution of a system with several interacting inviscid shear waves, whose initial conditions are drawn from a Gaussian distribution [291]. Here, we use NeuralUQ to solve the *inverse* Kraichnan-Orszag problem, i.e., identify the unknowns in the governing equations given sparse and noisy measurements on the solutions without the knowledge of the initial conditions. The governing equations are:

$$\frac{dx_1}{dt} = ax_2x_3, \quad \frac{dx_2}{dt} = bx_1x_3, \quad \frac{dx_3}{dt} = -(a+b)x_1x_2, \quad (6.9)$$

$$x_1(0) = 1.0, \quad x_2(0) = 0.8, \quad x_3(0) = 0.5, \quad (6.10)$$

where $a = 1$ and $b = 1$ are constants.

Particularly for this problem, we assume that a and b are unknown, whereas 11 noisy measurements of x_1 and x_3 are available, and 7 of x_2 ; see also Figure 6.5. The

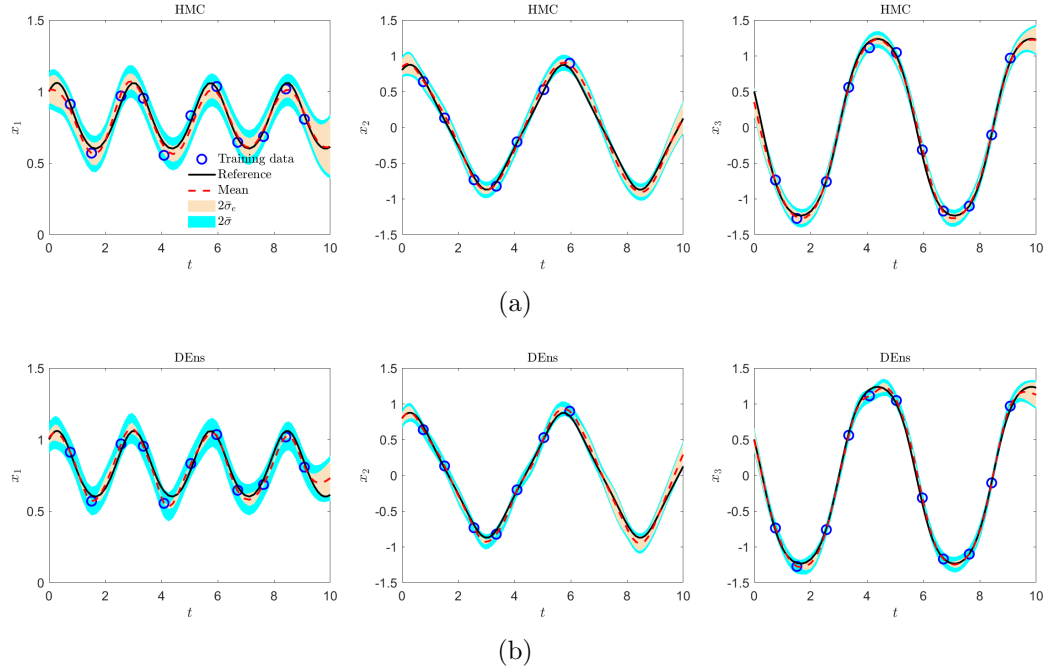


Figure 6.5: Inverse Kraichnan-Orszag problem: Predictions corresponding to x_1 , x_2 , and x_3 using HMC (a) and DEns (b). Epistemic uncertainty is denoted by $2\bar{\sigma}_e$, whereas total uncertainty by $2\bar{\sigma}$.

noise for all measurements is assumed to be Gaussian with zero mean and a known standard deviation set to 0.05. The objective here is to identify a and b , as well as to reconstruct the complete solutions x_1 , x_2 , and x_3 for $t \in [0, 10]$, with uncertainties.

We present the predictions for x_1, x_2, x_3 and a, b using different approaches in Figure 6.5 and Table 6.3, respectively. As shown in Figure 6.5, for both HMC and DEns, the predicted uncertainty increases at locations with no available data. Further, the error between the predicted mean and the reference solution for x_1, x_2, x_3 is bounded by the total uncertainty. Furthermore, as shown in Table 6.3, the predicted values for a and b , using both approaches, are close to the reference values. In the table, the results using MFVI and MCD are also provided for comparisons. In Table 6.4, we evaluate the predictions for x_1, x_2, x_3 using the RL2E and the NLL metrics. As shown, HMC and DEns provide the most reasonable predictions for the mean as well as uncertainties according to the computed metrics. Next, the respec-

tive computational costs can be found in Table 6.5. As expected, the UQ methods HMC and DEns are more expensive than the deterministic PINN method. In this case, we implement DEns by considering 50 independently trained PINNs and its computational cost is approximately the same as the cost of HMC. The 50 trained NNs used in the ensemble are trained simultaneously and independently on one CPU by using efficient vectorization techniques [1], which deliver a considerable speedup over training multiple PINNs in a serial manner. If more computational resources, e.g., CPUs or GPUs, are available, DEns can be further accelerated to the level of training a single deterministic PINN by using parallel computing. Additional implementation details for this case can be found in the caption of Table 6.5 as well as in <https://github.com/Crunch-UQ4MI>.

We further demonstrate the effect of the number of training data on the predicted uncertainty based on the above test case. In particular, we vary the number of noisy measurements for x_1 , from 11 to 22 and 44, and keep the rest of the problem setup the same. For simplicity, we only present in Figure 6.6 the predictions for x_1 from HMC and DEns. It is shown that the predicted *epistemic* uncertainty decreases as we increase the number of training data, whereas the uncertainty arising from data noise, i.e., *aleatoric* uncertainty, cannot be reduced by adding more training data.

	HMC	DEns	MFVI	MCD
a (mean $\pm$ std)	0.8567 ± 0.0500	0.8427 ± 0.0976	0.5706 ± 0.0205	0.6141 ± 0.0202
b (mean $\pm$ std)	1.0547 ± 0.0347	1.1557 ± 0.0404	0.9388 ± 0.0166	0.8931 ± 0.0161

Table 6.3: Inverse Kraichnan-Orszag problem: Predictions corresponding to a and b using different methods. The reference values are $a = b = 1$.

	HMC	DEns	MFVI	MCD
RL2E	0.0620	0.0578	0.2232	0.1527
NLL ($\times 10^2$)	-3.7969	-4.0402	-0.1187	-1.8930

Table 6.4: Inverse Kraichnan-Orszag problem: Performance evaluation of the considered methods on test dataset. Note that NLL is computed using the total uncertainty. Smaller RL2E and NLL mean better accuracy for the predictions of mean and uncertainty, respectively. RL2E corresponds to relative L_2 error, whereas NLL to the negative log likelihood.

Methods	HMC	DEns	Deterministic PINN
Wall time (seconds)	45.4	49.4	7.1

Table 6.5: Inverse Kraichnan-Orszag problem: Computational time for HMC and DEns for the case in Figure 6.5. The computational time for a deterministic PINN model is also included for comparison. For HMC, the number of burn-in steps is set to 3,000, the step size is 0.01, and the leap-frog step is 50. Further, the adaptive step size technique of [9] is also utilized to achieve acceptable acceptance rates. We collect 1,000 posterior samples after the burn-in phase to compute the predicted means and uncertainties in Figure 6.5. For DEns, 50 PINNs are trained simultaneously and independently, using the vectorization techniques of [1]. Further, we employ the Adam optimizer [152] for the training of PINNs with 20,000 iterations in both DEns and the deterministic PINN method. We note that this problem falls into the small-data regime, and thus full batch training is employed in all three methods. All the computations are performed on one AMD Ryzen 9 6900HS CPU. Additional implementation details can be found in <https://github.com/Crunch-UQ4MI>.

6.4.2 SIRD model for COVID-19

Scientific models are critical tools for understanding and forecasting the spread of the COVID-19. However, constructing models that are capable of accurately describing the COVID-19 still remains a challenge. Here, we demonstrate the capability of NeuralUQ for learning the corresponding model with uncertainties given measurements. In particular, we consider the SIRD compartmental model with time-dependent parameters/coefficients/rates for the COVID-19 [51, 36, 263], expressed as:

$$\begin{aligned} \frac{dS}{dt} &= \frac{-\beta(t)SI}{n}, \quad \frac{dI}{dt} = \frac{\beta(t)SI}{n} - \gamma(t)I - \nu(t)I, \\ \frac{dR}{dt} &= \gamma(t)I, \quad \frac{dD}{dt} = \nu(t)I, \end{aligned} \tag{6.11}$$

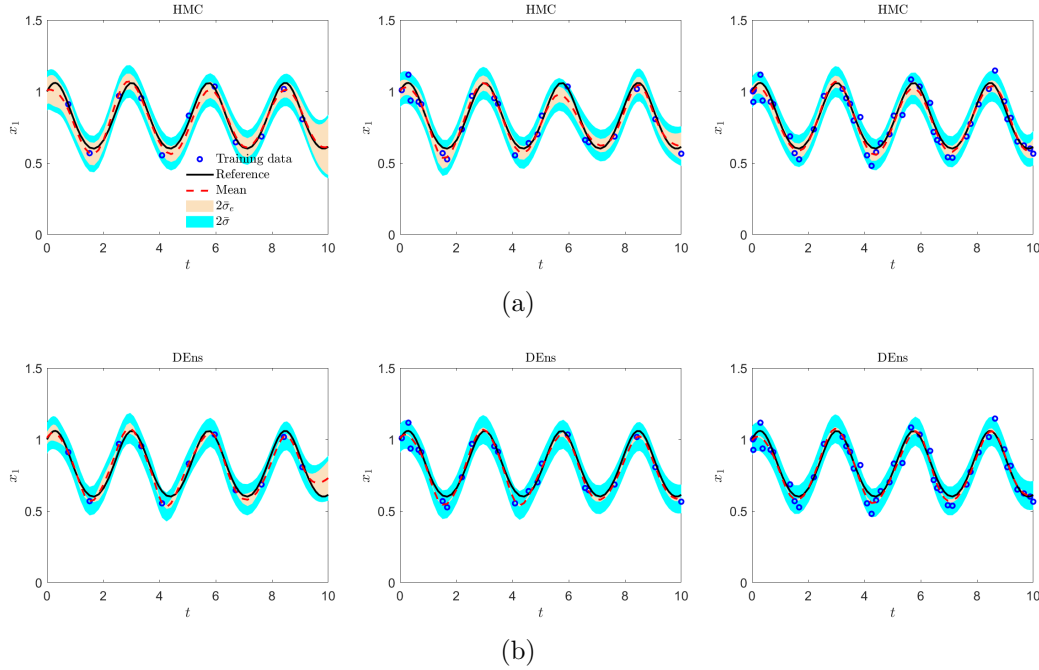


Figure 6.6: Inverse Kraichnan-Orszag problem: Predictions for x_1 with different numbers of measurements on x_1 , using (a) HMC and (b) DEns. In each row, the numbers of training data for x_1 are 11, 22, and 44, from left to right. Epistemic uncertainty is denoted by $2\bar{\sigma}_e$, whereas total uncertainty by $2\bar{\sigma}$. The predicted epistemic uncertainty decreases as we increase the number of training data, whereas the uncertainty arising from data noise, i.e., aleatoric uncertainty, cannot be reduced by adding more training data.

where β , the transmission rate, γ , the recovery rate, and ν , the death rate, are unknown and assumed to be time-dependent functions, and n is the total population, which is assumed to be constant and equal to the summation of S, I, R, D throughout the time span. $S(t), I(t), R(t)$ and $D(t)$ are the number of individuals susceptible to the disease, the number of individuals infected, the (cumulative) number of individuals recovered, and the (cumulative) number of individuals deceased from the disease, respectively, at time t . In this example, we use the COVID-19 dataset from Italy [75], which is the same as in [150].

In this problem, we utilize the data on I, R , and D in the first 160 days for training. The objective is to predict I, R , and D for $t \in [161, 200]$, and identify the

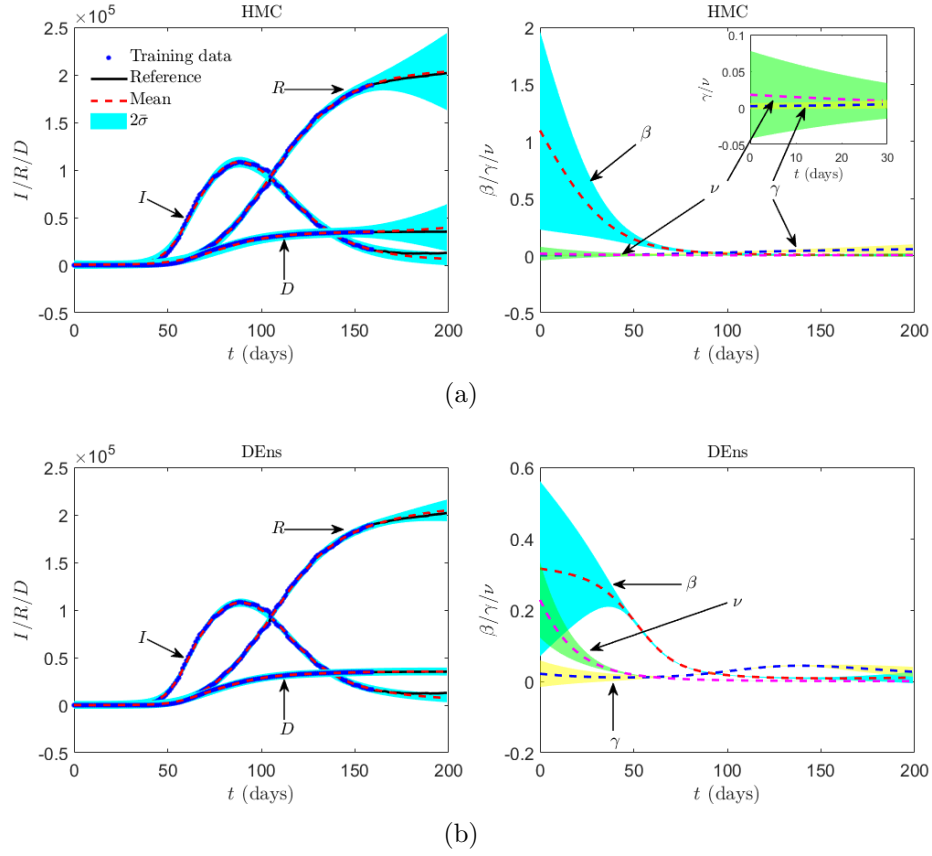
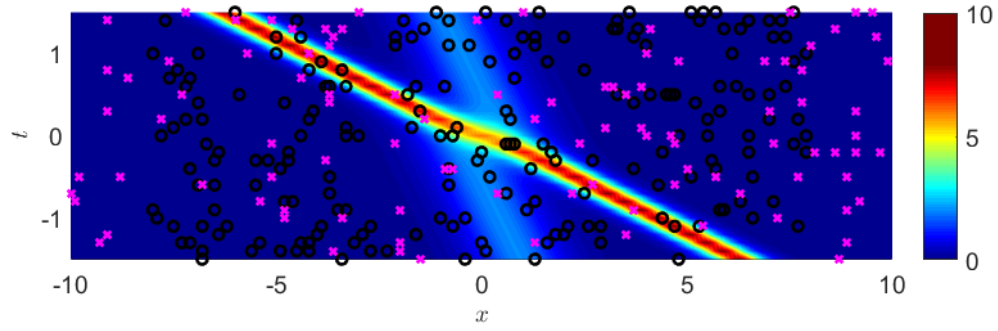
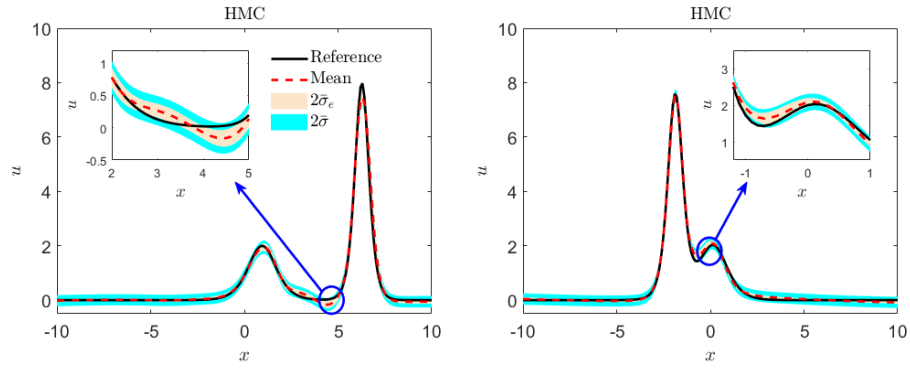


Figure 6.7: Inverse SIRD problem: Predictions for I , R , D and also β , γ and ν , using HMC (a) and DEns (b). Total uncertainty is denoted by 2σ .

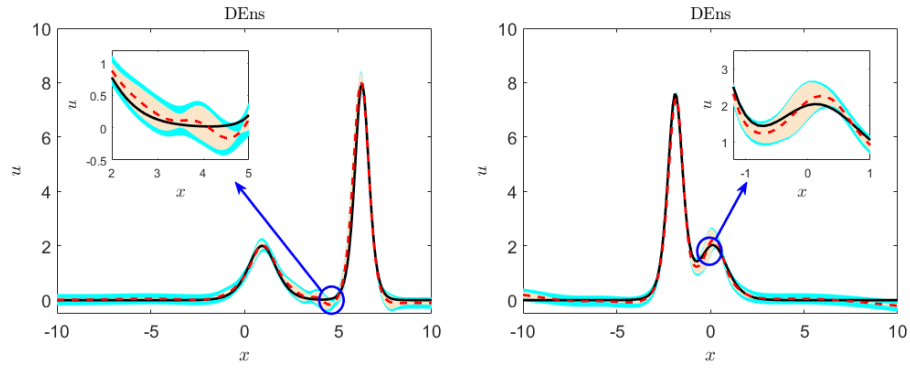
unknown functions β , γ , and ν for $t \in [0, 200]$. The results from HMC and DEns are presented in Figure 6.7. We observe that the error between the predicted mean and the reference solution of $I/R/D$ from both approaches is bounded by the total uncertainties. Further, HMC and DEns yield different predictions for β and ν for times $t < 100$, while they are quite similar for $t > 100$. Also, both methods predict a decrease of β and ν with time. In addition, the predictions for γ from both methods are similar, i.e., they are close to zero for $t \in [0, 200]$. Note that reference solutions for the unknown functions β , γ , and ν , are not available and thus corresponding comparisons are not provided here.



(a)



(b)



(c)

Figure 6.8: Inverse KdV problem: Predictions for $u(x, t)$ at different times, as obtained by using HMC and DENs. (a) Locations of training data for u and f ; black circles are used for u and magenta cross symbols for f . (b) HMC; $t = -1.5$ in the left side and $t = 0.4$ in the right side. (c) DENs; $t = -1.5$ in the left side and $t = 0.4$ in the right side. Epistemic uncertainty is denoted by $2\bar{\sigma}_e$, whereas total uncertainty by $2\bar{\sigma}$. Note that the scaling for t is the same as in [21]. One could also use a different scaling for t to make it positive only.

6.4.3 Inverse KdV problem

The Korteweg-de Vries (KdV) equation, derived more than 150 years ago, can be used to model surface waves in a shallow canal. It has been applied in various disciplines for modeling surface gravity waves, internal solitons in the ocean, magma flows, and conduit waves, for example. In this section, we solve an inverse KdV problem using NeuralUQ. Specifically, we discover the KdV equation given data for a system with interactions between two solitary waves. The KdV equation is expressed in the general form:

$$u_t - \lambda_1 uu_x - \lambda_2 u_{xxx} = f, \quad (6.12)$$

where λ_1 and λ_2 are constants, and $f = 0$ is considered. The exact solution can be expressed as [21]

$$u = 2\partial_x^2 R, \quad (6.13)$$

where

$$R = \log \left[\exp(-\eta_1 - \eta_2) + \exp(\eta_1 - \eta_2) + \exp(\eta_2 - \eta_1) + \frac{(a_1 - a_2)^2}{(a_1 + a_2)^2} \exp(\eta_1 + \eta_2) \right], \quad (6.14)$$

and

$$\begin{aligned} \eta_1 &= a_1 x + a_1^3 \mu t + b_1, \eta_2 = a_2 x + a_2^3 \mu t + b_2, \\ a_1 &= 1, a_2 = 2, b_1 = \log(3)/2, b_2 = b_1, \mu = 1, \end{aligned} \quad (6.15)$$

where $\lambda_1 = 1.5$ and $\lambda_2 = 0.25$ are considered.

For the inverse problem, the objective is to identify λ_1 and λ_2 , which are considered unknown, and to predict u in the entire spatial-temporal domain, given partial noisy measurements on u and f . Specifically, 200 and 100 random noisy measure-

ments on u and f , respectively, are considered available; see also Figure 6.8(a). The noise for the measurements on both u and f is assumed to be Gaussian with zero mean and known standard deviation set to 0.1.

Shown in Figures 6.8(b)-6.8(c) are the errors between the HMC and DEns predicted means, for u at $t = -1.5$ and 0.4 , and the reference solutions bounded by the predicted total uncertainties. Further, the predicted values for λ_1 and λ_2 , which are provided in Table 6.6, are close to the reference values for both methods.

	HMC	DEns
λ_1 (mean $\pm$ std)	1.4095 ± 0.0372	1.5452 ± 0.0672
λ_2 (mean $\pm$ std)	0.2642 ± 0.0057	0.3249 ± 0.0234

Table 6.6: Inverse KdV problem: Predictions corresponding to λ_1 and λ_2 using HMC and DEns; the reference values are $\lambda_1 = 1.5$ and $\lambda_2 = 0.25$.

6.4.4 100-dimensional Darcy problem

Flow through porous media is studied in various disciplines, such as chemical engineering, e.g., for flows in packed-bed catalytic reactors, oil recovery, e.g., for the displacement of oil with water, and in geophysics e.g., for the transport of pollutant in soil. A steady flow through porous media in two dimensions is considered here, which is described by the Darcy's law as follows:

$$\nabla \cdot (\lambda(x, y; \xi) \nabla u(x, y; \xi)) = f, \quad x, y \in [0, 1], \quad (6.16)$$

where λ is the hydraulic conductivity field, u denotes the hydraulic head, and $f = 50$ is considered. The boundary conditions are:

$$u(x = 0, y) = 1, \quad u(x = 1, y) = 0, \quad (6.17a)$$

$$\partial_{\mathbf{n}} u(x, y = 0) = \partial_{\mathbf{n}} u(x, y = 1) = 0, \quad (6.17b)$$

where $\mathbf{n}$ denotes the unit normal vector of the boundary.

In general, $\lambda(x, y; \xi)$ is determined by the pore structure. In this study, we employ a stochastic model for $\lambda(x, y; \xi)$ to consider porous media with different micro-structures. In particular, $\lambda(x, y; \xi) = \exp(\tilde{\lambda}(x, y; \xi))$, where $\tilde{\lambda}(x, y; \xi)$ is a truncated Karhunen-Loève expansion of a Gaussian process with zero mean and the following kernel:

$$k_{\tilde{\lambda}}(x, y, x', y') = \exp\left(-\frac{(x - x')^2}{2l^2} - \frac{(y - y')^2}{2l^2}\right), \quad (6.18a)$$

$$0 \leq x, y, x', y' \leq 1, \quad l = 0.25, \quad (6.18b)$$

and we only keep the first 100 leading terms of the expansion.

We consider two different scenarios here, i.e., Case A, in which we employ a DeepONet mapping $\tilde{\lambda}$ to u pretrained with 9900 different paired data $(\tilde{\lambda}, u)$, and the test data are partial noisy measurements on λ and u ; and Case B, in which we employ the DEns method to train a DeepONet using the same training data as in Case A, which also maps $\tilde{\lambda}$ to u . Complete λ measurements are used as test data, i.e., a measurement is available at each required location of the DeepONet input. Particularly, we refer to the DeepONet trained with DEns as uncertain DeepONet (U-DeepONet) following [234].

6.4.4.1 Physics-agnostic functional prior

For Case A, 20 noisy measurements of λ and u are assumed to be available; see also Figures 6.9 and 6.10 for the respective locations. The objective is to reconstruct the entire λ and u given partial noisy measurements of λ/u . Specifically, a pretrained

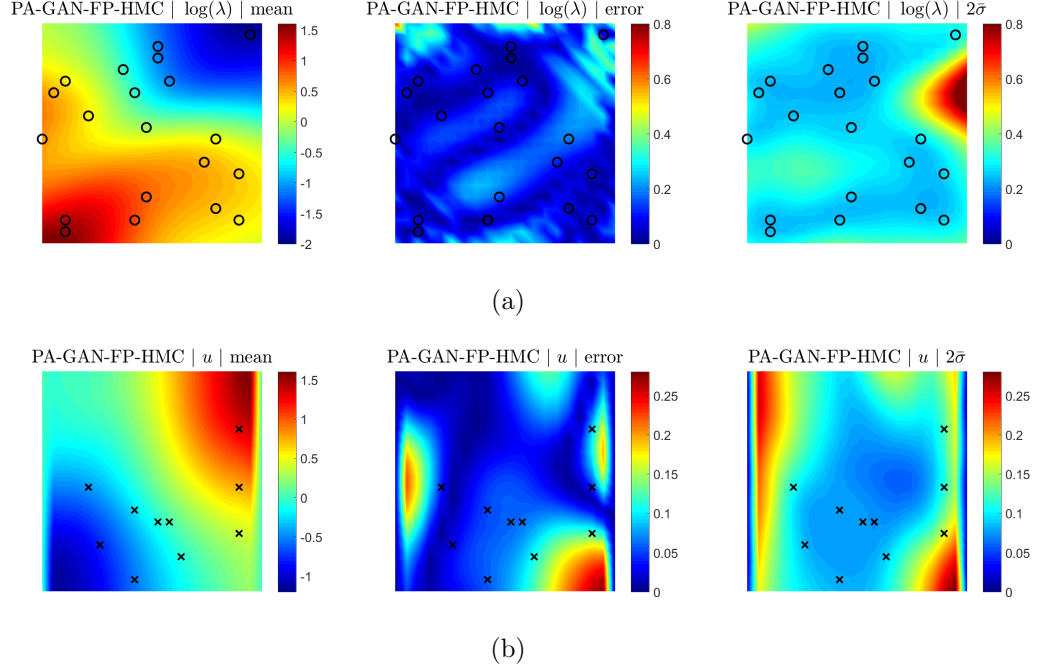


Figure 6.9: 100-dimensional Darcy problem (Case A): Predictions for $\log(\lambda)$ and u from PA-GAN-FP+HMC. Training data for λ are shown with circles, whereas training data for u with cross symbols. Total uncertainty is denoted by $2\bar{\sigma}$.

DeepONet combined with two physics-agnostic models, i.e. a PA-BNN-FP and a PA-GAN-FP, are considered. In particular, the PA-GAN-FP is pretrained using the same data for training the DeepONet to learn the functional prior for $\tilde{\lambda}$, while we specify a standard normal distribution as the prior for each parameter in the PA-BNN-FP for the prior of $\tilde{\lambda}$. For posterior inference, HMC is used for both models. The details for the pretrained of DeepONet and PA-GAN-FP can be found in the GitHub repository.

With the pretrained DeepONet and the physics-agnostic models as the prior in function space, we present the predictions for λ and u in Figures 6.9 and 6.10, respectively. We observe that the errors between the predicted means and the reference solutions for λ and u are mostly bounded by the predicted uncertainties in both methods. Further, the predicted uncertainties from PA-GAN-FP are smaller than those from PA-BNN-FP, which is expected since PA-GAN-FP is equipped with a

more informative prior. Interested readers can refer to [201, 234] for more comparison between the GAN- and BNN-FP.

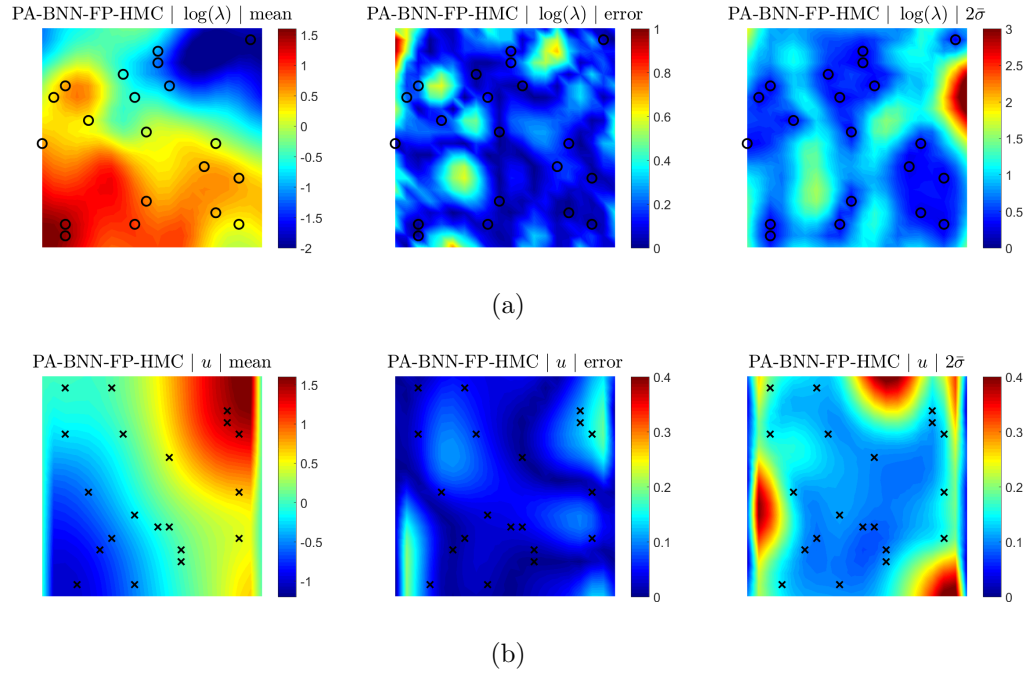


Figure 6.10: 100-dimensional Darcy problem (Case A): Predictions for $\log(\lambda)$ and u from PA-BNN-FP+HMC. Training data for λ are shown with circles, whereas training data for u with cross symbols. Total uncertainty is denoted by $2\bar{\sigma}$.

6.4.4.2 Uncertain DeepONet

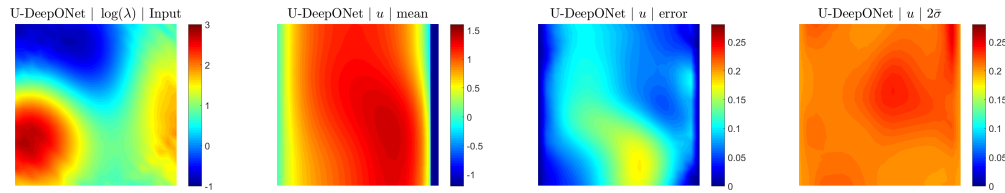


Figure 6.11: 100-dimensional Darcy problem: Predictions obtained from U-DeepONet and corresponding to u . Total uncertainty is denoted by $2\bar{\sigma}$.

For Case B, we assume that a complete input λ is available, i.e., a datapoint is available at each required location of λ . We quantify model uncertainty in the prediction of u using DEns. In particular, we run the DeepONet training algorithm five times with different parameter initializations, and subsequently compute the

predicted mean and uncertainties based on the five predictions of u ; see (6.7)-(6.8). The error between the predicted mean and the reference solution of u for an unseen $\log(\lambda)$ is bounded by the predicted uncertainties, as shown in Figure 6.11.

6.5 Summary

Scientific machine learning (SciML) has emerged recently as an effective and powerful tool for data fusion, solving ordinary/partial differential equations (ODEs, PDEs), and learning operator mappings in various scientific and engineering disciplines. Indicative SciML models are the physics-informed neural networks (PINNs) for solving forward and inverse ODE/PDE problems and the deep operator networks (DeepONets) for learning operators that can, for instance, be used as fast solvers of ODEs/PDEs. In this context, quantifying predictive uncertainties is crucial for risk-sensitive applications as well as for efficient and economical design. In this regard, a comprehensive study of UQ methods for various SciML models, e.g., neural networks (NNs), PINNs, DeepONets, has been provided in [234].

However, due to the fact that the implementations of current UQ methods are not straightforward, we have developed in this chapter an open-source Python library, termed NeuralUQ (available at <https://github.com/Crunch-UQ4MI>), that serves as an efficient and reliable toolbox for UQ in SciML. In this study, we have presented a detailed tutorial of NeuralUQ, followed by four numerical examples, including dynamical systems and high-dimensional parametric and time-dependent PDEs, to demonstrate the applicability, reliability, and efficiency of NeuralUQ. We have further demonstrated the flexibility and customizability of NeuralUQ for supporting alternative surrogates, prior and posterior parameter distributions, as well

as the inference methods. Indicative such modifications have been described and applied in regression and differential equation problems. More generally, in addition to the UQ methods presented herein for addressing the problem scenarios of Section 6.2.1, NeuralUQ enables the users to address additional scenarios by combining the built-in components, such as the different surrogates and inference methods.

Despite the plurality of methods implemented in NeuralUQ, appropriately selecting a UQ method for solving a specific problem is not trivial. In this regard, we provide a few empirical remarks in the following. Deterministic methods, such as deep ensemble, are computationally efficient, and they provide reasonable model uncertainties for both neural ODE/PDE and neural operators problems. Further, they can handle big data by using data mini-batches, as in standard neural network (NN) training. For problems with noisy data, NN parameter regularization can be used for addressing overfitting. However, this may have a negative effect on predicted uncertainties. Specifically, a large regularization weight can lead to small model uncertainty, whereas a small weight may be insufficient for addressing overfitting issues. As a potential remedy, different regularization weights can be used and compared by evaluating their corresponding predicted uncertainties using metrics and validation data. As for Bayesian methods, Hamiltonian Monte Carlo (HMC) typically achieves both satisfactory computational accuracy and efficiency for posterior estimation for problems with relatively small datasets, of the order of a few hundred measurements. For problems involving big data, the methods Langevin dynamics (LD), mean-field variational inference (MFVI), and Monte Carlo dropout (MCD), which support mini-batch training in a straightforward manner, are computationally more efficient. For solving neural ODE/PDE problems without and with historical data using Bayesian methods, we recommend the use of a BNN and the generator of a pretrained physics-informed generative adversarial network as surro-

gates, respectively. Further, as a rule of thumb for neural operator problems, the physics-agnostic functional prior (PA-FP) can be used to quantify the uncertainties caused by incomplete data, and the deep ensemble for quantifying the epistemic uncertainties. Furthermore, the methods HMC and LD/MFVI are recommended for problems with small and big data, respectively, if FP is used.

Overall, we envision that NeuralUQ will be used in the future in large-scale computational science and engineering applications as a toolbox of reliable and efficient UQ implementations. Clearly, such applications require up-to-date, well-maintained, and versatile software packages. In this regard, indicative extensions of our library relate to incorporating more neural models, such as neural ordinary differential equations (NeuralODE) [48], sparse identification of nonlinear dynamics (SINDy) [29], and ensemble-SINDy [83], more sampling methods that can handle big data, e.g., stochastic gradient Hamiltonian Monte Carlo, and variational inference methods with more accurate parameterized posterior distributions, e.g., normalizing flows. Finally, we will continue updating the library and we also encourage future users and researchers to develop additional components for NeuralUQ with the goal of expanding its current capabilities and advancing the field.

CHAPTER SEVEN

Summary and Future Work

In this dissertation, we have explored theory, methods, and software for uncertainty quantification (UQ) in scientific machine learning (SciML), presenting a comprehensive and systematic study. Building upon existing work, much of which has focused on uncertain, imprecise, or incomplete data [234, 281, 315, 318, 174, 321, 320, 317, 335, 92, 112, 206, 173, 329, 175, 90, 325, 136], we have addressed additional challenges in this field. Specifically, this dissertation has tackled:

1. Data sparsity and high noise levels in Chapter 2.
2. Interpretability, large-scale data, incremental/continual learning under limited computational resource, and hyperparameter tuning in Chapter 3.
3. Noise model misspecifications in Chapter 4.
4. Physical model misspecifications in Chapter 5.
5. Implementation of modern UQ methods through the open-source library NeuralUQ in Chapter 6.

While these contributions represent significant progress, UQ in SciML still faces notable limitations. For large and complex systems, scalable UQ methods are still insufficient. For example, although our work in Chapter 3 connects Bayesian inference in SciML with Hamilton-Jacobi partial differential equations (HJ PDEs) to enable incremental/continual learning with big or streaming data, it only applies to linear ODEs/PDEs. Extending this approach to general nonlinear problems poses substantial theoretical challenges. Furthermore, the proposed real-time UQ method, while effective for Gaussian priors and likelihoods, cannot handle more complex data distributions. Practical UQ approaches, like ensemble-based methods [234], have proven valuable, but their theoretical foundation and the interpretation of the quantified uncertainty, remains insufficiently explored.

Another key remaining challenge is managing model misspecifications and model uncertainty in physical systems, where uncertainty reflects incomplete knowledge about models (e.g., physical models in [334, 67] and Chapter 5 and noise models in Chapter 4). In Chapter 5, for example, we addressed physical model misspecifications in physics-informed neural networks (PINNs) [244] by introducing additional NNs. However, this study does not explore the uncertainty levels of the physical model or the neural operator (NO) as the surrogate model. Similarly, while Chapter 5 discusses noise model misspecification, it focuses only on one specific scenario: neglecting noise in the input of PINNs and NOs. Overall, our prior work on model uncertainty primarily examines extreme cases of either completely known or entirely unknown models, leaving many intermediate scenarios unexplored.

Given these limitations, the research presented in this dissertation can be extended in the following directions:

1. **Scalable UQ for SciML:** Scalable UQ methods for SciML face significant challenges, including high-dimensional neural network parameters and large data volumes. While ensemble-based methods, such as deep ensembles [163], have demonstrated practical success in SciML [234], they often produce overconfident predictions, particularly in noisy data scenarios, as regularization techniques to prevent overfitting tend to suppress estimated uncertainty [348]. This presents opportunities to develop more robust and scalable UQ methods for large systems, leveraging recent advances that integrate ensemble approaches with Bayesian learning [126, 62, 303, 119]. One promising method is Stein variational gradient descent [176], a general variational inference approach that generates samples of the target distribution via functional gradient descent. This method has shown promise in solving prototype SciML problems

with UQ [342, 341] and warrants further exploration for more complex systems and real-world engineering problems.

2. **Real-time UQ:** Real-time UQ is crucial for risk-sensitive and complex applications such as digital twins [20, 202], autonomous systems [253, 339], and healthcare [55, 274, 28]. However, several challenges remain. The primary one is achieving computational efficiency without compromising accuracy, as many existing UQ methods involve high-dimensional computations that are too slow for real-time applications. Another obstacle lies in handling noisy, incomplete, or rapidly evolving data, which requires robust data assimilation techniques and adaptive models. Furthermore, integrating UQ into dynamic systems with feedback control introduces further complexity (e.g., [186, 18]), as uncertainty must be quantified and addressed within tight time constraints. One possible direction is to leverage UQ-related continual learning methods such as [5, 78, 331, 28] to address these challenges, as continual learning often involves streamed data, incremental updates, and real-time inferences [296, 288, 222].
3. **Beyond optimal control:** Recent studies [47, 46, 345] have highlighted the potential of linking machine learning problems with HJ PDEs (and optimal control) offers a promising approach to solving ODEs/PDEs with UQ, especially for large or streaming datasets. So far, this connection has only been established for linear problems: convex loss functions, from the machine learning point of view, and linear quadratic regulators, from the optimal control point of view. If the assumption on convexity is relaxed by allowing for general loss functions—equivalent to nonquadratic costs in optimal control and nonconvex Hamiltonians in HJ PDEs—then this theoretical connection could be extended to be between machine learning problems and differential games [80, 127] rather than optimal control such that more general learning problems

in SciML may be interpreted or enhanced to handle more complex systems. This connection with differential games to tackle nonlinear differential equations with high complexity is to be further explored.

4. **Model uncertainty:** In prior work on model misspecifications [346, 347, 67, 334], we focused on either fully known or unknown models, overlooking intermediate uncertainty levels. In many scenarios, model uncertainty exists to a certain degree. For example, in [334], we assumed that certain properties of the reaction model for Alzheimer’s disease must be satisfied, which simultaneously reduced the model uncertainty. Moreover, model uncertainty can vary over time. For instance, the lack of knowledge about physical models could be progressively reduced by acquiring more information [120, 53]. To address this, more advanced UQ methods need to be developed to quantify this type of uncertainty and capture the associated time-dependent behavior in SciML applications.
5. **UQ for rare events:** UQ for rare events focuses on analyzing and quantifying low-probability but high-impact phenomena across various fields, including climate science, finance, engineering, and natural disaster modeling [140, 30, 254]. These events are characterized by their infrequent occurrence and substantial consequences, making their prediction both essential and challenging. Recent efforts have sought to address these challenges, particularly in modeling, sampling, and mitigating data scarcity, by leveraging machine learning techniques (e.g., [230, 273, 12, 26]). Despite these advancements, further research is needed to develop more efficient and robust methods tailored to rare event analysis.

Finally, several open challenges in UQ persist and require further investigation:

1. **Hyperparameter selection and tuning:** Selecting and tuning hyperparam-

eters in the context of UQ, such as prior distributions, regularization constants, and NN architectures, is neither straightforward nor computationally efficient (see, e.g., [85]).

2. **Overparameterization and multi-modal posteriors:** The overparameterization of NNs results in overly multi-modal posterior distributions in UQ. Developing efficient methods to obtain representative samples from such distributions remains an open problem (see, e.g., [308, 212]).
3. **Performance evaluation:** Evaluating the performance of UQ methods is not as straightforward as in standard NN training, where validation or test data are typically used. This is because the true data-generating distribution is unknown and depends on factors such as the number and quality of measurements and the choice of hyperparameters (see, e.g., [323]).
4. **Marginal predictions:** Most UQ studies focus on predictions for single inputs, yielding marginal distributions that fail to capture correlations between, for instance, spatially or temporally adjacent points (see, e.g., [217, 302]).
5. **Robustness under distribution shift:** Assessing the robustness of UQ models in cases of distribution shift, where model inputs deviate from the distribution of the training data, remains a significant challenge (see, e.g., [218, 304]). Developing UQ models capable of detecting such cases and producing appropriately less confident predictions, i.e., admitting “ignorance”, is critical for their safe deployment in real-world applications.

Addressing these challenges will require continued advancements in theoretical understanding, algorithmic innovations, and practical implementations. By building on the foundations laid in this dissertation, future research can move closer to achieving robust, interpretable, and scalable UQ solutions for SciML.

APPENDIX

APPENDIX A

Appendix for Chapter 2

A.1 Details of NN architectures and training hyperparameters

For all examples in Section 2.4 and Appendix A.3 except the one in Section 2.4.5, MH-PINNs are implemented as fully-connected NNs (FNNs) with three nonlinear hidden layers, each of which is equipped with 50 neurons and hyperbolic tangent activation function. In Section 2.4.5, MH-PINNs are FNNs with four nonlinear hidden layers (200 neurons and hyperbolic tangent activation function). The number of heads is the same as the number of tasks in the corresponding examples: 1,000 in Section 2.4.1, 2,000 in Sections 2.4.2, 2.4.3 and 2.4.4, and 10,000 in Section 2.4.5. Weights in the body of MH-PINNs are initialized with Glorot uniform initialization [101] and biases are initialized with zero, while heads are initialized by sampling from random normal distribution with 0.05 standard deviation, for fast training of NFs and better performance of the learned generators.

Except for the forward problem in Section 2.4.2, NFs in this chapter are chosen to be MAF [221] with ten bijectors, i.e. the invertible map in NFs, each of which is a MADE [97], a NN with masked dense layers, with two nonlinear hidden layers equipped with 100 neurons and ReLU activation function. The RealNVP [74] and IAF [153] used in the forward problem in Section 2.4.2 also have ten bijectors, each of which is a NN with two nonlinear hidden layers equipped with 100 neurons and ReLU activation function. The implementation mostly follows the instructions of TensorFlow Probability library [73] for NFs.

PI-GANs [317] implemented in Section 2.4.2 have the following architecture: the discriminator is a FNN with three nonlinear hidden layers, each of which is equipped with 128 neurons and Leaky ReLU activation function; the generator that takes as

input t is a FNN with three nonlinear hidden layers, each of which is equipped with 50 neurons and hyperbolic tangent activation function; the other generator takes as input a Gaussian random variable in 50 dimensions with zero mean and identity covariance matrix, and is implemented as a FNN with three nonlinear hidden layers, each of which has 128 neurons and hyperbolic tangent activation function. The input dimensions of those three FNNs are 65, 1 and 50, and the output dimensions are 1, 50, 50, respectively.

For the training of MH-PINNs, full-batch training is deployed with Adam optimizer for 50,000 iterations. For the training of NFs, except for the forward problem in Section 2.4.2, mini-batch training is deployed with batch size being 100 and Adam optimizer for 1,000 epochs. NFs in the forward problem in Section 2.4.2 are trained for 500 epochs instead, and L_2 regularization is imposed to the parameters of Real-NVP for better performance. For all NFs, to achieve stable training, a hyperbolic tangent function is imposed on the logarithm of the scale, computed from each bijector, such that the logarithm of the scale lies in $(-1, 1)$. For the training of PI-GANs, mini-batch training is deployed with batch size being 100 and Adam optimizer for 100,000 iterations. Besides, the same as in [317, 201], physics-informed Wasserstein GANs (PI-WGANs) with gradient penalty are employed, in which the coefficient for gradient penalty is set to be 0.1. Iteratively, five updates of the discriminator are performed and followed by one update of the generators. Except in training PI-GANs, the learning rate of Adam optimizer is set to be 10^{-3} and other hyperparameters of Adam are set as default. In training PI-GANs, the learning rate is set to 10^{-4} , $\beta_1 = 0.5$ and $\beta_2 = 0.9$ in Adam optimizer for both discriminator and generators.

The training of MH-PINNs, NFs, and PI-GANs was performed on a single NVIDIA TITAN Xp GPU.

A.2 Details for performing Bayesian inference

Hamiltonian Monte Carlo (HMC) [211] is employed in all Bayesian inference examples for uncertainty quantification (UQ) while Laplace approximation [145] is only employed in the first example. In this chapter, HMC with adaptive step size [166] is used, in which the initial step size is set to be either 0.1 or 0.01, tuned for better acceptance rate. The number of burn-in samples and the number of posterior samples are set to be 1,000. The number of steps for the leapfrog scheme is set to be either 30 or 50, also tuned for better acceptance rate. The open-source NeuralUQ library [348] is used as a tool for physics-informed Bayesian inference in the downstream tasks. The ideal acceptance rate in HMC, as discussed in [201, 348], is around 60%. In this chapter, we found chains with acceptance rate from 50% to 80% acceptable.

A.3 Multi-task learning with multi-head neural networks

In this chapter, we primarily focused on using MH-NNs in conjunction with NFs to estimate stochastic functions and learn informative prior knowledge from tasks $\{\mathcal{T}^k\}_{k=1}^M$. This was accomplished by first training MH-NNs in a MTL framework and then training NFs to estimate the PDF of the head. The success in learning functional priors can largely be attributed to the ability of MH-NNs in MTL to capture shared information across tasks. For physics-informed MTL with MH-PINNs, ODEs/PDEs are solved simultaneously, and by assuming that solutions share the same set of basis functions, we obtain samples of the coefficient set, enabling generative modeling followed by few-shot learning, which is the central objective of the

proposed method. However, the cost or benefit of enforcing the same basis functions across all solutions has not yet been thoroughly examined.

On the one hand, the shared body links task training, which may be beneficial when tasks exhibit similar patterns. On the other hand, forcing all solutions to share the same basis functions can be counterproductive if task behaviors diverge. In particular, for tasks with sufficient data and physics, forcing them to share the same body with all other tasks may act as a negative regularization, and single-task learning (STL) may outperform MTL in prediction accuracy for those specific tasks. In this section, we explore the effects of MTL using MH-NNs and provide preliminary results and analysis by revisiting the simple function approximation example in Section 2.4.1. We hope this initial investigation offers useful insights and guidance for future, more rigorous research.

A.3.1 Basis function learning and synergistic learning

As discussed before, the quality and behavior of basis functions learned in MTL are crucial to generative modeling and learning the relation and the representative information of tasks $\{\mathcal{T}^k\}_{k=1}^M$. We consistently noticed from numerical examples that the initialization of the head in MH-NNs has great impact on the average accuracy of MTL, the learning of the basis functions, and the distribution of the head. Here, we test three initialization strategies, random normal method with 0.05 standard deviation referred to as *RN (0.05)*, Glorot uniform method [101] referred to as *GU*, and random normal method with 1 standard deviation referred to as *RN (1)*. In the downstream few-shot learning tasks, we fine-tune the head without the learned PDF, which is in fact the TL method from [71], by which the information from the distribution of the head is excluded and the prediction accuracy is fully determined

by the level of prior knowledge contained in the basis functions.

As shown in Figure A.1, RN (0.05) yields the least informative basis functions, whose behavior is dominated by the hyperbolic tangent activation function of NNs. This is further demonstrated in the downstream few-shot learning tasks using the TL method. It also provides the worst prediction accuracy on average in MTL, as presented in Table A.1. GN and RN (1) perform similarly. Plots of some basis functions seemingly indicate that RN (1) yields better basis functions, whose behaviors are more similar to the family of functions displayed in Figure A.1(b), which, however, does not necessarily imply richer prior knowledge in the downstream tasks, as shown in Figure A.1(c).

It is shown empirically that compared to other two initialization strategies, MH-NNs with RN (0.05) does not deliver accurate MTL nor synergistic learning in basis functions. However, we noticed that, in generative modeling, it performs significantly better in terms of accuracy and convergence speed. As shown in Figure A.1(d), samples from the learned generator are of higher quality. We consistently found that initializing heads with relatively small values often led to easy and fast training of NFs and accurate learning of the generative models. We conjecture that this happens because MH-NNs in MTL tend to contain the representative and informative information in the heads when heads are initialized with small values, while contain it in the basis functions when heads are initialized with relatively large values.

	RN (0.05)	GU	RN (1)
Error (%)	0.8373 ± 0.2341	0.1907 ± 0.0690	0.3131 ± 0.0937

Table A.1: L_2 relative errors, from MTL, for 1,000 tasks, using different initialization methods. The errors are displayed in the format of mean $\pm$ standard deviation, computed over all tasks.

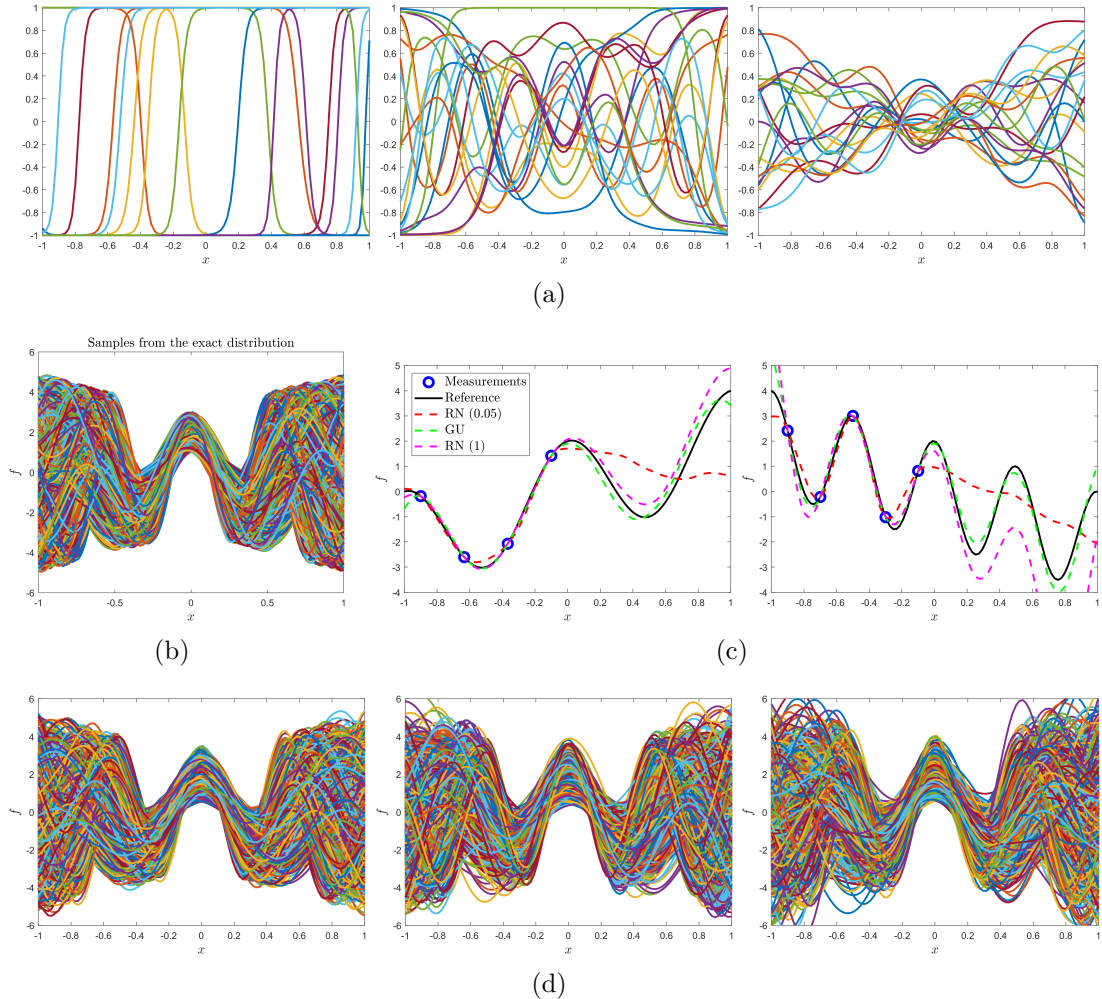


Figure A.1: The effect of different initialization methods of the head, in basis functions learning, few-shot learning, and functional prior learning. (a) Samples of 20 basis functions from MH-NNs, trained for approximating 1,000 f generated from (2.14), using, from left to right, RN (0.05), GU and RN (1) initialization methods. (b) 1,000 training samples of f . (c) Results for two downstream few-shot regression tasks, using TL method without the regularization informed by the learned PDF, as opposite to the proposed approach. (d) Results for functional prior learning, using, from left to right, RN (0.05), GU and RN (1) initialization methods.

A.3.2 Multi-Task Learning (MTL) versus Single-Task Learning (STL)

As discussed earlier, MTL with MH-NNs does not necessarily result in synergistic learning nor higher accuracy for all tasks on average. Here, we use again the function

approximation example in Section 2.4.1, to investigate the effectiveness of MTL with MH-NNs, as compared to STL. The first case we consider here is assuming that the data is sufficient. For that, we randomly choose 100 out of the 1,000 training samples, each one of which is approximated by a NN trained independently, and compare the results with MH-NNs in terms of prediction accuracy. Note that in this case, a MH-NN is trained on 1,000 functions as before and tested on the chosen 100 functions, while a single-head NN with the same architecture is trained on 100 functions directly. Results are shown in Table A.2, from which it is verified empirically that MTL is outperformed by STL under certain circumstances, e.g., when the random normal initialization methods are used.

	RN (0.05)	GU	RN (1)	STL
Error (%)	0.7575 ± 0.2477	0.1362 ± 0.0259	0.3664 ± 0.1031	0.2102 ± 0.0794

Table A.2: L_2 relative errors of f , from MTL with MH-NNs and STL with NNs, on 100 tasks. Different initialization methods are used for the heads in MH-NNs. The errors are displayed in the format of mean $\pm$ standard deviation, computed over all 100 tasks.

The second case we consider is assuming that the data is sufficient for some tasks while insufficient for other tasks. For that, we split equally the 1,000 tasks into two subsets of tasks. For the first 500 tasks, we assume we only have 10 measurements randomly sampled on $[-1, 1]$, while for the other 500 tasks, we assume we have full 40 measurements equidistantly distributed on $[-1, 1]$. MTL with MH-NNs is performed on those 1,000 regression tasks all at once, and the tasks are treated as equal. The results are presented in Table A.3 and Figure A.2. We can see that, compared to STL, MTL improves the prediction accuracy on tasks with insufficient data, providing empirical evidence of synergistic learning. Also, interestingly, RN (1) initialization method, which yields the worst generative models, performs the best among all three, which agrees with our previous conjecture on the basis functions learning with MH-NNs, that heads initialized with large values tend to force representative

and informative information to be encoded in the basis functions.

	RN (0.05)	GU	RN (1)
Error (%)	63.60 ± 24.08	40.49 ± 20.49	16.91 ± 11.08

Table A.3: L_2 relative errors of f , from MTL with MH-NNs, on 500 tasks equipped with insufficient data. The errors are displayed in the format of mean $\pm$ standard deviation, computed over all 500 tasks.

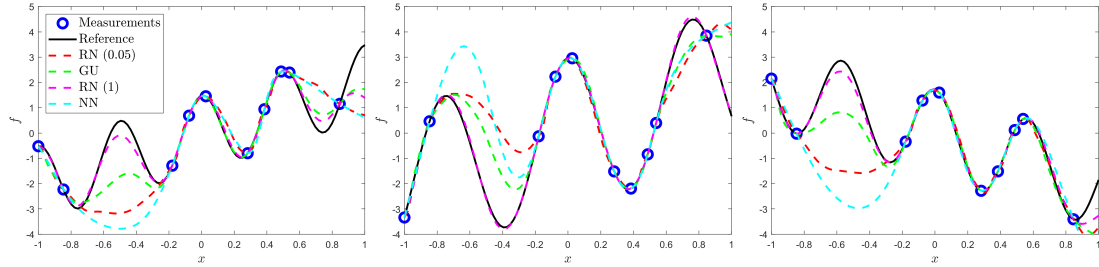


Figure A.2: Results for three tasks with insufficient data from MTL with MH-NNs, using different initialization methods over the head, and from STL with NNs with the same architecture. We note that tasks with sufficient data and tasks with insufficient data are treated equally in MTL.

APPENDIX B

Appendix for Chapter 3

B.1 Details of the hyperparameters in the numerical examples

In Section 3.4, we use RK4 to numerically solve the Riccati ODEs, where the step size h of RK4 is chosen to achieve a balance between high accuracy and stable, efficient computations. We use Python, the NumPy library [118], the JAX library [27], and double precision in all numerical examples. An NVIDIA A100 GPU is used to accelerate computations.

In Section 3.4.1.1, we use $h = 5 \times 10^{-5}$. In Section 3.4.1.2, the baseline solution using $\sigma = 1$ is computed using the method of least squares. Then, σ is tuned using the values of σ and h shown in Table B.1. In Appendix B.2, the baseline solution using the entire training set is computed using the method of least squares, and we use RK4 with step size $h = 1 \times 10^{-5}$ to delete each outlier. In Section 3.4.2.1, we use $h = 1 \times 10^{-1}$, and in Section 3.4.2.2, we use $h = 1 \times 10^{-2}$. In Section 3.4.3, we use $h = 2 \times 10^{-6}$ for both domain decomposition patterns.

σ	$1 \rightarrow 0.5$	$1 \rightarrow 2$	$2 \rightarrow 5$	$5 \rightarrow 10$	$10 \rightarrow 20$
h	10^{-5}	10^{-5}	10^{-6}	10^{-7}	10^{-7}

Table B.1: Step size h of RK4 when tuning σ in the hyperparameter tuning example in Section 3.4.1.2. $\sigma_0 \rightarrow \sigma_1$ denotes the case where σ is tuned from σ_0 to σ_1 .

B.2 Additional results for example 1

In this section, we present the results of removing outliers from the trained model when solving (3.17) using our Riccati-based methodology from Section 3.3.2.1. Specifically, we first train the model with the entire noisy training set, and then

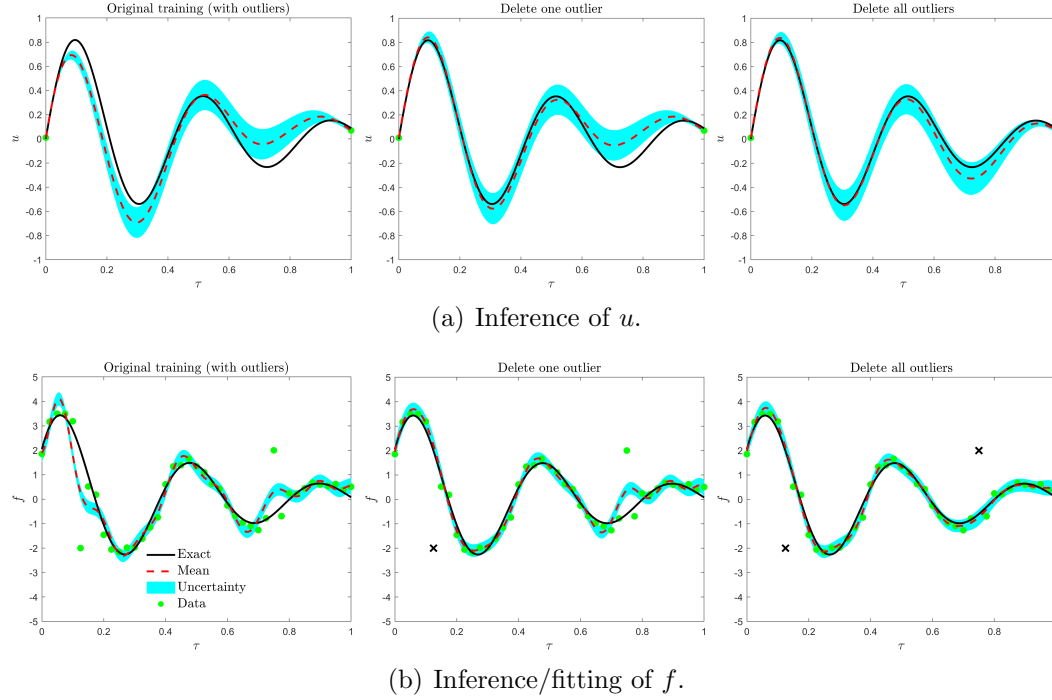


Figure B.1: Deleting two outliers ($\times$) sequentially from the trained model using our Riccati-based approach to solve (3.17). The **left** column of (a) and (b) shows the inferences of u and f , respectively, after the initial training with all data points. The **middle** column shows the inferences after the first outlier is deleted, and the **right** column shows the inferences after two outliers are deleted. The outliers are deleted by solving the associated Riccati ODEs backward in time. This process only uses the results of the previous training step and information of the point to be deleted, which provides potential computational advantages over more standard SciML approaches that would otherwise require retraining on the entire remaining dataset.

remove the effect of identified outliers from the learned model. Figure B.1 shows the results of the initial training and the subsequent sequential removal of two outliers. As expected, removing the outliers improves the model prediction accuracy. After both outliers are removed, the exact values now lie within the uncertainty band around the predicted means, which indicates that our models have become more reliable. Recall that the outliers are removed by solving one step of Riccati ODEs backward in time. As such, we do not require retraining on or access to the entire remaining dataset, which provides potential computational savings.

B.3 Additional results for example 2

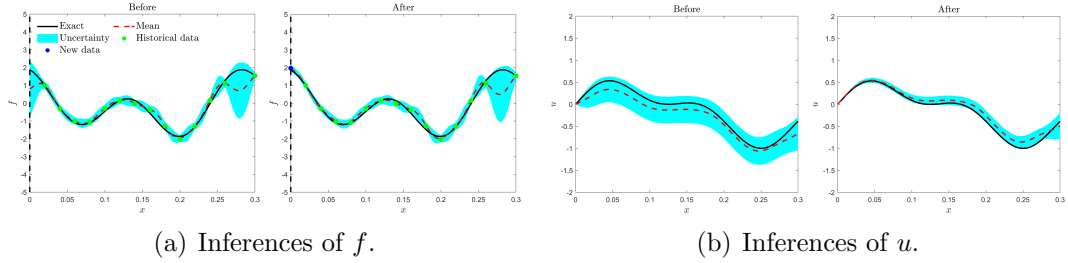


Figure B.2: Intermediate result for adding the 47th data point when using active learning and our Riccati-based approach to solve (3.20). The **left** side of (a) and (b) shows the inferences of f and u , respectively, before the placement of the new sensor and incorporation of the 47th data point. The **right** side of (a) and (b) shows the updates to f and u after these actions. The vertical black dashed line represents the location ($x = 0$) where the 47th sensor will be placed. Note that we zoom in on the region around the new sensor instead of showing the full domain for clarity of comparison. Our Riccati-based approach naturally complements this active learning framework as it can efficiently perform repeated updates of the predicted mean and uncertainty.

In this section, we present additional results for Case B of Example 2. Recall that in Section 3.4.2.2, we consider the scenario where we have access to high quality data with low-level noise but sensors are expensive to deploy. As such, the location of new data measurements must be chosen carefully. Following an active learning framework [248], we choose the location of the next sensor to be where the predicted uncertainty is the highest. In Figure B.2, we show the intermediate results when adding the 47th sensor. In this case, the predicted uncertainty of u more dramatically reduces. This example demonstrates how the potential of our Riccati-based approach for real-time inferences allows it to be seamlessly integrated into active learning applications.

APPENDIX C

Appendix for Chapter 4

C.1 Details of hyperparameters in the numerical experiments

In this section, we provide details of all numerical examples presented in Section 4.4. We use Hamiltonian Monte Carlo (HMC) [211] with adaptive step size [9] to estimate all posterior distributions, in which the number of steps for the leap-frog scheme is set to 50 and 1,000 posterior samples are obtained. The number of burn-in samples and the initial step size are tuned such that the acceptance rate between 50% and 70% are achieved. An open-source Python library for UQ in SciML, termed NeuralUQ [348], is used for implementations. For examples involving NOs, the functions for hybrid problems are randomly sampled from the testing data rather than data used to train NOs beforehand.

In Section 4.4.1, we employ BNNs with two hidden layers, each of which is equipped with hyperbolic tangent as the activation function and has 50 neurons. The prior of the NN parameter, i.e. $p(\theta)$, is set to independent Gaussian with mean zero and standard deviation one. The prior of the actual inputs, i.e. $p(\chi)$, is set to independent Gaussian with mean zero and standard deviation 100, representing weak regularization of the actual inputs. We observed that in practice, omitting $p(\chi)$ in the posterior, which indicates that χ follows independent uniform distribution on a sufficiently large interval, was able to provide accurate inferences and reasonable uncertainties as well. The prior of $\log(\lambda)$ is set to Gaussian with mean zero and standard deviation one.

In Section 4.4.2, we employ one BNN for p_{im} and one BNN for p_{em} , and both BNNs have two layers, each of which is equipped with hyperbolic tangent and 50 neurons. The prior of the NN parameter is set to independent Gaussian with mean

zero and standard deviation five. The prior of the actual inputs is omitted in the posterior estimation, indicating uniform distribution on a sufficiently large interval. The priors of $\hat{c}_1$ and $\hat{c}_2$ are Gaussian with mean zero and standard deviation one. The inferences of c_1 and c_2 are obtained by $10^4\hat{c}_1$ and $10^3\hat{c}_2$, respectively. In addition, we rescale p_{im} and p_{em} by $p_{im}/10^4 - 15$ and $p_{em}/10^4 - 15$, respectively. The right-hand sides of (4.17) are rescaled by the factor of 10^4 accordingly.

In Section 4.4.3, we employ one BNN with two hidden layers to model u . Each layer has 50 neurons and hyperbolic tangent as the activation function. The prior of the NN parameter is set to independent Gaussian with mean zero and standard deviation one, and the priors of $\log D$ and λ are Gaussian with mean zero and standard deviation one. In this example, we impose an additional prior over the actual inputs to ensure that they remain inside the computational domain $[0, 1]^2$. To achieve this, we model $\xi_x := F_Z^{-1}(x)$ and $\xi_y := F_Z^{-1}(y)$ instead, where (x, y) is the actual input and F_Z denotes the cumulative distribution function of a standard normal random variable, i.e. $F_Z(z) = \int_{-\infty}^z \frac{1}{\sqrt{2\pi}} \exp(-\frac{s^2}{2}) ds$. The priors of ξ_x and ξ_y are Gaussian with mean zero and standard deviation one.

In Section 4.4.4, a vanilla FNO is employed to learn the solution operator of (4.24), which maps the initial condition to the solution at $t = 1$. All hyperparameters of this FNO and its training are the same as the ones in [171]. We use the same data from [171] (1,000 for training and 200 for testing where the initial condition is generated from the distribution $\mathcal{N}(0, 625(-\Delta + 25I)^{-2})$) and discretize the input and output functions on a uniform mesh with 128 grids. The pretrained FNO used for the downstream hybrid problem achieved 0.50% relative L_2 error in average on testing data. The prior of the discretized input function is set to multi-variate Gaussian with mean zero and covariance matrix estimated from the 1,000 data used to train the FNO.

In Section 4.4.5, a multi-input DeepONet [138] is employed to learn the solution operator of (4.25), which maps the function k in the diffusion term and the source term f to the solution. Both input functions are discretized on a uniform mesh with 100 grids. There are two branch nets in the multi-input DeepONet and they both have one hidden layer with 200 neurons and ReLU activation function. The trunk net is of the same architecture. The output dimensions of the trunk net and the branch nets are 200. The Adam optimizer [152] with learning rate 0.001 and default setting is employed to train the multi-input DeepONet for 200,000 iterations. The data of k and f are generated by a Gaussian process with zero mean and the following exponential squared kernel:

$$k(x_1, x_2) = \exp\left(-\frac{(x_1 - x_2)^2}{2l^2}\right), \quad x_1, x_2 \in [0, 1], \quad (\text{C.1})$$

where $l = 0.2$ is the correlation length. We used 5,000 data to train the DeepONet and achieved 2.77% relative L_2 error in average on 1,000 testing data.

In Section 4.4.6, a vanilla DeepONet is employed to learn the solution operator of (4.27), which maps the logarithm of the hydraulic conductivity $\log(\lambda)$ to the hydraulic head u . Both $\log(\lambda)$ and u are discretized on a 31×31 uniform grid. The branch net of the DeepONet has three hidden layers while the trunk net has two hidden layers. Each layer has 128 neurons and has hyperbolic tangent as the activation function. The output dimensions of the branch and trunk nets are 128. We employed the same Adam optimizer to train the DeepONet for 50,000 iterations on 10,000 training data and achieved 2.08% relative L_2 error on 1,000 testing data.

C.2 Additional results

C.2.1 1D nonlinear Poisson equation with PINNs

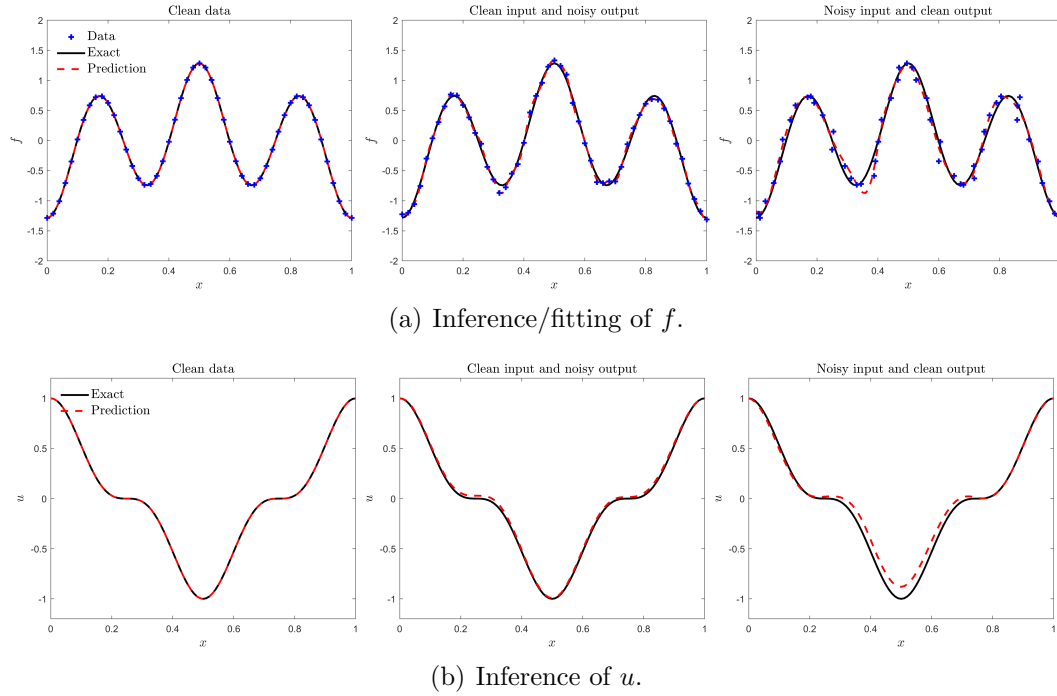


Figure C.1: The effect of noisy inputs-outputs in solving (4.15) using PINNs. Three scenarios are considered: clean data, noisy output but clean input data, and clean output but noisy input data of f . The noises for input and output data are additive Gaussian noises with the same standard deviation 0.01. Comparison shows that noise in the input data has more significant effect upon the performance of PINNs than noise in the output data. L_2 regularization is employed to prevent overfitting in all two noisy-data scenarios. Quantitative results can be found in Table 4.2.

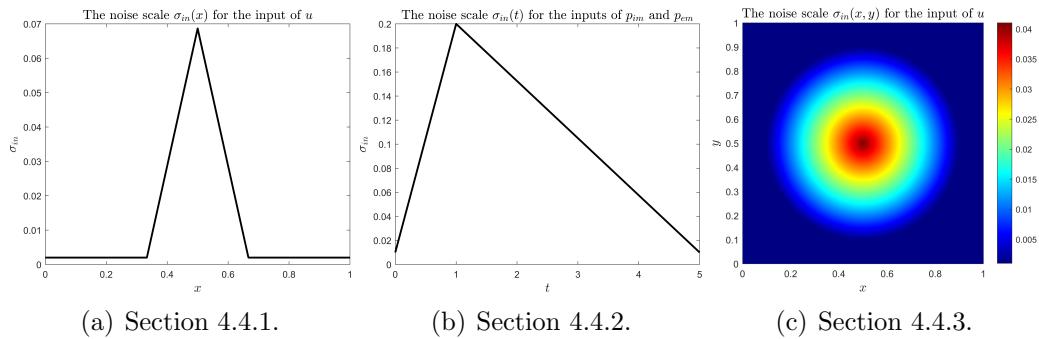


Figure C.2: Noise models for the input of u considered in different PINN problems.

In this section, we present additional results in Figure C.1 for the example in Section 4.4.1 where we employed PINNs to solve (4.15) with different types of data. We observe that, even with the same noise level, the noise in the input data has a more significant effect on the performance of PINNs than the noise in the output data, indicating the necessity and importance of accounting for input noise in solving PDEs.

C.2.2 1D time-dependent reaction-diffusion with DeepONets

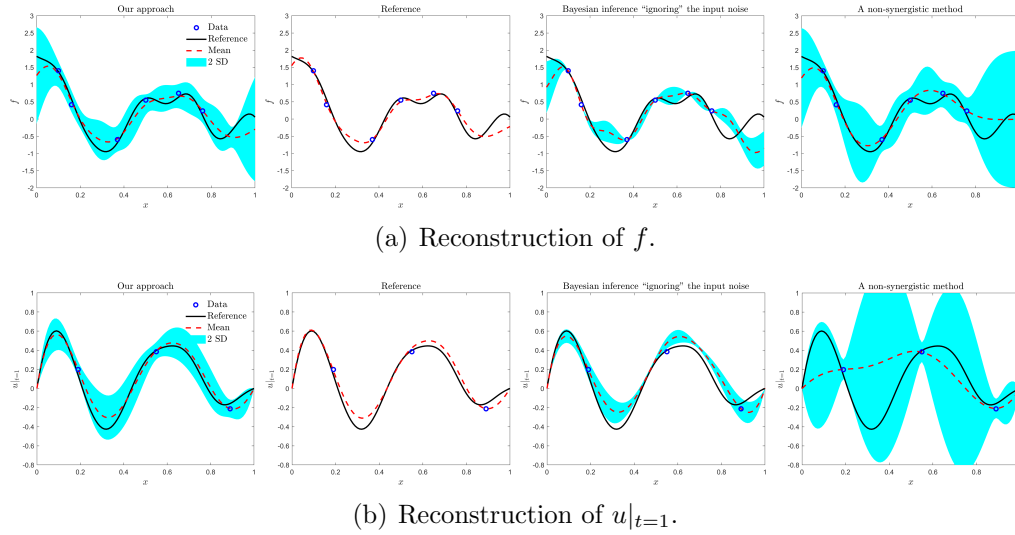


Figure C.3: The homogeneous reaction-diffusion equation: Reconstructing f and $u|_{t=1}$ from their sparse and noisy measurements, in which a pretrained DeepONet serves as an equation-free surrogate that encodes the physics. From left to right we present results from our approach with correctly specified input noise, the reference method (a MAP estimate), our approach but misspecifying the scale of the input noise as a relatively small value (0.01), and a non-synergistic method.

In this section we consider a simplified case of Section 4.4.5 where the diffusion term is a known constant across the field:

$$\frac{\partial u}{\partial t} = D \frac{\partial^2 u}{\partial x^2} + \kappa u^2 + f(x), x \in [0, 1], t \in [0, 1], \quad (\text{C.2})$$

where $D = 0.01$ and $\kappa = 0.01$ and f is the source term. We train a vanilla DeepONet [178] with clean and sufficient data to learn the solution operator from f to $u|_{t=1}$, and then use the pretrained NO to reconstruct f and $u|_{t=1}$ from their sparse and noisy measurements. Specifically, we assume six measurements of f and three measurements of $u|_{t=1}$ are available. They are randomly sampled from $x \in [0, 1]$ and corrupted by additive Gaussian noises with scales 0.2 and 0.05, respectively.

Results are presented in Figure C.3. We can see that our approach is able to provide accurate and trustworthy reconstructions of the input and output functions of the NO. Comparing our approach with the MAP estimate, we observe the necessity of UQ in reconstructing functions with pretrained NOs when noisy data are encountered. Similar to results in Section 4.4.4, misspecifying/ignoring the input noise yields significantly worse and unreliable predictions in inferring both the input and output functions. We also test the non-synergistic method in this case, and larger predicted uncertainties and errors demonstrate that reconstructions of the input and output functions are performed synergistically with our approach. Besides, as shown in Figure C.3, the predicted uncertainty of f around $x = 0.9$ from our approach yields a relatively small value compared to the nearby region even though no measurement of f is available near $x = 0.9$. This is caused by the existence of a measurement of $u|_{t=1}$ around $x = 0.9$ and therefore showcases the synergistic learning.

APPENDIX D

Appendix for Chapter 5

D.1 Details of hyperparameters in numerical experiments

In Section 5.3.1, DNNs used to model u/s have two hidden layers, each of which has 50 neurons and is equipped with hyperbolic tangent as the activation function, and take as input t and output u/s . In Section 5.3.2, DNNs have two hidden layers and each layer has 50 neurons and a hyperbolic tangent activation function. They take as input x, t and output u or s . In Section 5.3.3.1, DNNs have two hidden layers (50 neurons and a hyperbolic tangent activation function) and take as input y and output u or s . In Section 5.3.3.2, a DNN with three hidden layers (100 neurons and a hyperbolic tangent activation function) is used to model u, v . That is, this DNN takes as input x, y and outputs u, v . Another DNN with three hidden layers (100 neurons and a hyperbolic tangent activation function) is used to model p . Besides, an additional DNN with three hidden layers (100 neurons and a hyperbolic tangent activation function) is used to correct two misspecified equations and it takes as input x, y and outputs s_2, s_3 .

The ensemble PINN method [234, 348] is used for clean data to obtain predictions with UQ. In Section 5.3.1, 20 PINNs are independently trained with Adam optimizer [152] (10^{-3} learning rate) for 100,000 iterations. In Section 5.3.3.2, 10 PINNs are independently trained with Adam optimizer (10^{-3} learning rate) for 200,000 iterations. To obtain results shown in Figure 5.9, PINNs are trained with Adam optimizer for 500,000 iterations. The learning rate decays from 10^{-3} to 10^{-5} for stable training.

The B-PINN method [315] is used for noisy data to obtain predictions with UQ. In all related examples, parameters of DNNs are assumed to be identically independent distributed (i.i.d.) Gaussian as the prior distribution with mean zero and standard

deviation one. The likelihood distribution is also assumed to be independent Gaussian where the standard deviation is the known noise scale for additive Gaussian noise. In addition, Hamiltonian Monte Carlo (HMC) with adaptive step size [211] is used to obtain posterior samples, in which the number of posterior samples is set to 1,000 and the number of steps for the leap-frog scheme is set to 50. The number of burn-in samples and the initial step are tuned such that the acceptance rate lies between 50% and 70%. A open-sourced Python library for UQ in scientific machine learning, termed NeuralUQ [348], is used for fast and convenient implementations.

D.2 Additional results for the ODE system

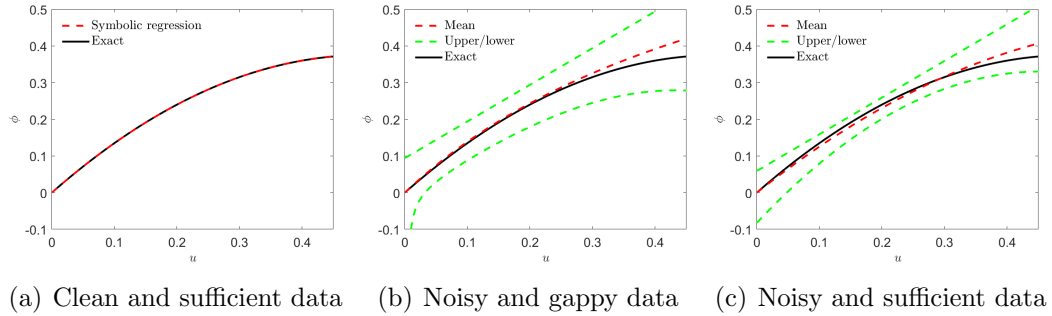


Figure D.1: ODE system: Plots of the discovered equation $\phi(u)$ using symbolic regression, following our approach in correcting the model misspecification; see Table 5.3 for the mathematical expressions. In (a), the symbolic regression is performed on the predicted mean from ensemble PINNs. In (b) and (c), the symbolic regression is performed on the predicted mean as well as the upper and lower bounds of the uncertainty band ($\text{mean} \pm 2\text{stds}$) from B-PINNs. We note that the symbolic regression is performed on $\phi(u)$ instead of $\phi(t)$.

In Figure D.1, we present plots of the discovered equation $\phi(u)$ using symbolic regression in various cases; see Section 5.3.1 for details and Table 5.3 for the discovered mathematical expressions. Recall that in noisy-data cases, we employed B-PINNs and obtained uncertainties in predicting u and ϕ . By forming the data for symbolic regression as $\{u_\theta(t_i), s_\psi(t_i) + \tilde{\lambda} \cos(u_\theta(t_i))\}_{i=1}^N$, we in fact obtain uncertainty in $\phi(u)$

as well, which is leveraged in the following symbolic regression. As shown in Figure D.1, for noisy-data cases, we also perform symbolic regression on the upper and lower bounds of the uncertainty band in predicting $\phi(u)$. Although the discovered mathematical expressions in Table 5.3 from noisy-data cases are not as accurate, the error in function values is bounded by the upper and lower bounds. Besides, due to the increase of data, the predicted uncertainty from the noisy-and-sufficient-data case is significantly smaller.

D.3 Additional results for the 1D reaction-diffusion equation

In this section, we utilize the proposed method in combination with the symbolic regression to (1) correct the misspecified physics in these cases independently and (2) discover the explicit expression for the reaction model based on the trained NNs. Specifically, the first test case here is Case (C) in Section 5.3.2, where the reaction model is misspecified as λu^2 (with known $\lambda = 1$), and we use an additional NN to correct it. The misspecified part, i.e., $\phi = \lambda u^2 + s_\psi$ in the governing equation is illustrated in Figure 5.7. The symbolic regression is then employed to identify the mathematical expression for ϕ , which is now treated as a function of u . The results are presented in Table D.2, and discovered reaction model is close to the reference solution as observed.

To further improve the predicted accuracy, we assume that we have additional data from the same reaction-diffusion system, i.e., the governing equation is the same, but with different initial/boundary conditions, which are shown in Table D.1. Specifically, we assume that we have data from two additional cases where the ini-

tial/boundary conditions are different from the one used in Case (C). We note that the problem setups for the two new cases are the same as in Case (C), and we then employ the proposed method to correct the misspecification of the reaction model in PINNs. We denote the solutions as $u_1/u_2/u_3$ and the corrections as $s_1/s_2/s_3$ for the cases in Table D.1 (from left to right). The learned reaction terms $\phi_1/\phi_2/\phi_3$ can then be expressed as $\lambda u_1^2 + s_1/\lambda u_2^2 + s_2/\lambda u_3^2 + s_3$.

Initial condition	$u(x, 0) = 0.5 \sin^2(\pi x)$	$u(x, 0) = x(1 - \cos^2(\pi x))$	$u(x, 0) = \cos^2(\pi x)$
Boundary condition	$u(0, t) = u(1, t) = 0$	$u(0, t) = u(1, t) = 0$	$u(0, t) = u(1, t) = 1$

Table D.1: Different initial and boundary conditions for the reaction-diffusion equation. The initial/boundary conditions are for Case (C) of Section 5.3.2.

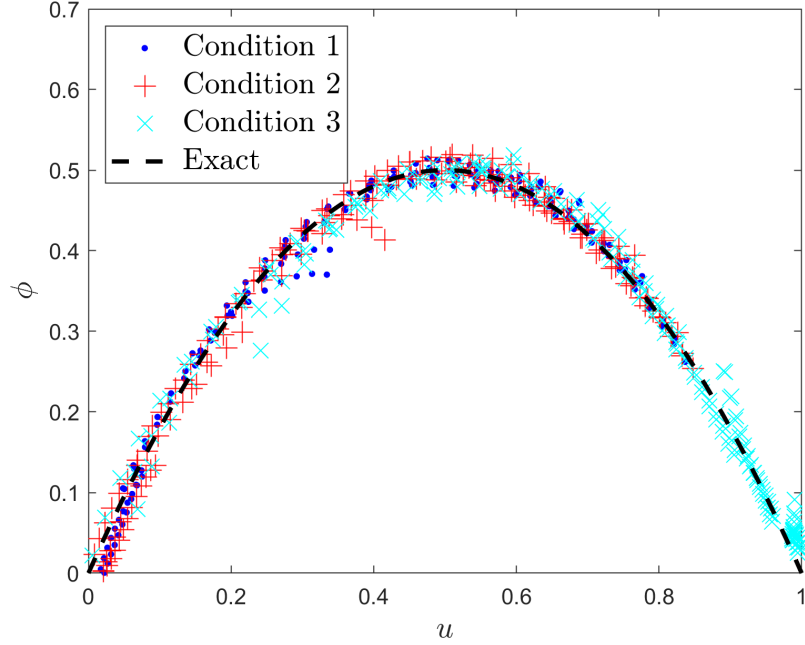


Figure D.2: Data for the identification of the reaction term $\phi(u)$, obtained from correcting the misspecified physics in three initial/boundary conditions (presented in Table D.1) independently using our approach. The exact ϕ - u relation is $\phi = 2u(1 - u)$.

We now proceed to identify the misspecified physics using symbolic regression given data on all the three cases. Similar to Case (C), we assume that the reaction term ϕ is a function of the solution u , i.e. $\phi = \phi(u)$, and it has the same formulation in the there test cases. We then apply the method from [58] to regress the analytic

One initial/boundary condition	Three initial/boundary conditions
$u(2.0878 - 2.0745u) - 0.0229$	$u(1.9698 - 1.9524u)$

Table D.2: The identified mathematical expressions of the reaction term $\phi(u)$ in the reaction-diffusion equation (5.11) using symbolic regression and data from multiple initial/boundary conditions. The exact is $2u(1 - u)$. The data for the symbolic regression are displayed in Figure D.2.

expression of the reaction term using the follownig data:

$$\cup_{j=1}^3 \{u_{\theta_j}(x_f^i, t_f^i), \lambda u_{\theta_j}^2(x_f^i, t_f^i) + s_{\psi_j}(x_f^i, t_f^i)\}_{i=1}^{N_f},$$

where $x_f^i, t_f^i, i = 1, \dots, N_f$ are grid points on which the source term f is evaluated and N_f is the number of data. The training data used in the symbolic regression are shown in Figure D.2, in which we can see that data are scattered around the exact ϕ - u relation. In addition, we observe that the data from the same equation but different initial conditions are able to provide extra information, e.g. data near $u = 1$, which is able improve the accuracy for identifying the reaction model, as illustrated in Table D.2. We note that the current approach for identifying the mathematical expressions for the misspecified physics is quite flexible and does not require the candidate library for the all possible formulations of the reaction process as in [52].

D.4 Details for the computations in lattice Boltzmann equation model

In this section, we provide details for generating data of the non-Newtonian Cavity flow in Section 5.3.3.2 using the lattice Boltzman equation model (LBE) in [295]. We set $\text{Re} = U^{2-n}L/\nu_0 = 100, L = 1, U = 0.01, \rho = 1, n = 1.5$ and use a uniform mesh of size 101×101 on $[0, 1]^2$ to perform the simulations.

APPENDIX E

Appendix for Chapter 6

In this section, we provide a glossary table in Appendix E.1, and an example for surrogate customization, i.e., a one-dimensional regression problem, in Appendix E.2. We then demonstrate the customization for prior in Samplable and posterior estimation in Variational, using a one-dimensional differential equation problem, in Appendix E.3, and provide the details for data generation in Appendix E.4.

E.1 Glossary for frequently used terms

In Table E.1, we provide a glossary table of frequently used terms in this chapter, and in ML/SciML more generally.

Glossary			
UQ	uncertainty quantification	SciML	scientific machine learning
DNNs	deep neural networks	BNNs	Bayesian neural networks
GANs	generative adversarial networks	FNN	fully-connected neural network
PINNs	physics-informed neural networks	DeepONets	deep operator networks
FP	functional prior	PI-GANs	physics-informed GANs
PA-FP	physics-agnostic FP	GAN-FP	GAN-based FP
BNN-FP	BNN-based FP		

Table E.1: Glossary of frequently used terms in this chapter.

E.2 Customization of surrogates

We demonstrate the surrogate customization, using a one-dimensional regression problem. The target function is:

$$u = 1.5 \sin(11x), \quad x \in [-1, 1]. \quad (\text{E.1})$$

We assume that we have 3 noisy measurements of u , which are equidistantly distributed in $[-0.7, -0.3]$. The noise for the measurements is assumed to be Gaussian

with zero mean and known standard deviation set to 0.05.

We employ three different surrogates for different scenarios: (1) We have neither historical data nor prior knowledge on the formulation of the target function. We then employ a BNN with one hidden layer which has 50 neurons and the hyperbolic tangent activation function as the surrogate. In addition, the standard normal distribution is utilized as the prior distribution for each weight and bias in the BNN; (2) the expression describing the target function is not considered as known, but we have historical data for it. We then train a GAN to learn the functional prior using historical data; and (3) we have prior knowledge regarding the target function expressing that it is a sine function with unknown amplitude and frequency, i.e., $A \sin(\omega x)$, where A and ω are parameters to be inferred given measurements. The details regarding the pre-trained GAN in (2) can be seen in the GitHub repository, and the posteriors for the parameters in each surrogate are computed using HMC.

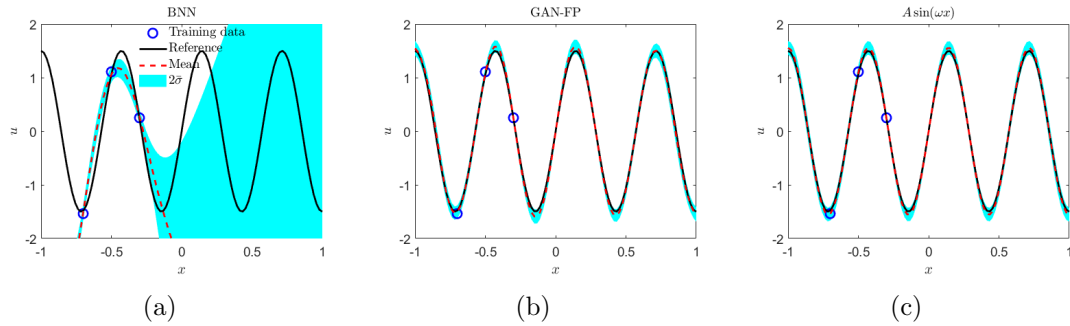


Figure E.1: 1D regression problem: Predictions for u using different surrogates. (a) BNN; (b) GAN-FP, and (c) $A \sin(\omega x)$. $2\bar{\sigma}$: Total uncertainty.

As shown in Figure E.1, the predictions from the GAN-FP and $A \sin(\omega x)$ surrogates are quite similar for both the predicted means and uncertainties, which are more accurate than the results from the BNN surrogate in terms of the means. Also, the errors between the predicted means and the reference solution are mostly bounded by the predicted uncertainties from GAN-FP and $A \sin(\omega x)$, whereas the errors are

only partly bounded by the predicted uncertainties in BNN. Overall, we demonstrate the flexibility and customizability of NeuralUQ by customizing problem-dependent surrogates to achieve better predicted accuracy.

E.3 Customization of prior and inference

We demonstrate the customization for prior in Samplable and posterior estimation in Variational, using a one-dimensional differential equation problem. Specifically, we consider a nonlinear diffusion-reaction system expressed as follows:

$$D\partial_x^2 u - k_r u^3 = f, \quad x \in [-1, 1], \quad (\text{E.2})$$

where u represents the concentration of a certain solute, D denotes the diffusion coefficient, k_r is the reaction rate, and f is the source term.

We first demonstrate the customization of prior using the example of an inverse problem described by (E.2). In this problem, $D = 0.01$ is a known constant, and we have a few noisy measurements on u , and f . The objective is to infer u and f for $x \in [-1, 1]$, and the reaction rate k_r which is an unknown constant. Specifically, we have noisy measurements of u at 5 random locations, and 17 noisy equidistant measurements of f . We then employ a BNN as well as a Samplable variable as the surrogate for u and k_r , respectively. The equation can then be encoded using the automatic differentiation as in PINNs. The employed BNN has 3 hidden layers with 50 neurons per layers, and the hyperbolic tangent activation function is used as the activation function. We assume that each weight/bias has the same prior distribution in this BNN, which is independent Gaussian here, i.e., $N(0, 1^2)$. Further, we employ

three different prior distributions for k_r , i.e., $k_r \sim N(0, 1^2)$, $k_r \sim \text{HalfN}(0.5, 0.5^2)$ and $\log k_r \sim N(0, 1^2)$, where *HalfN* refers to half-normal distribution. The posterior samples for the parameters in BNN as well as k_r are obtained using HMC. We only present the predictions for k_r in Table E.2, for simplicity. As shown, the predictions for k_r are strongly affected by its prior distribution. Overall, NeuralUQ enables the users to utilize different prior distributions to achieve better accuracy.

Priors for k_r	$k_r \sim N(0, 1^2)$	$k_r \sim \text{HalfN}(0, 1^2)$	$\log k_r \sim N(0, 1^2)$
mean $\pm$ std	0.5727 ± 0.4407	0.4131 ± 0.2488	0.4538 ± 0.2550

Table E.2: Nonlinear diffusion-reaction problem: Predictions for k_r with different priors. The reference solution is $k_r = 0.2$.

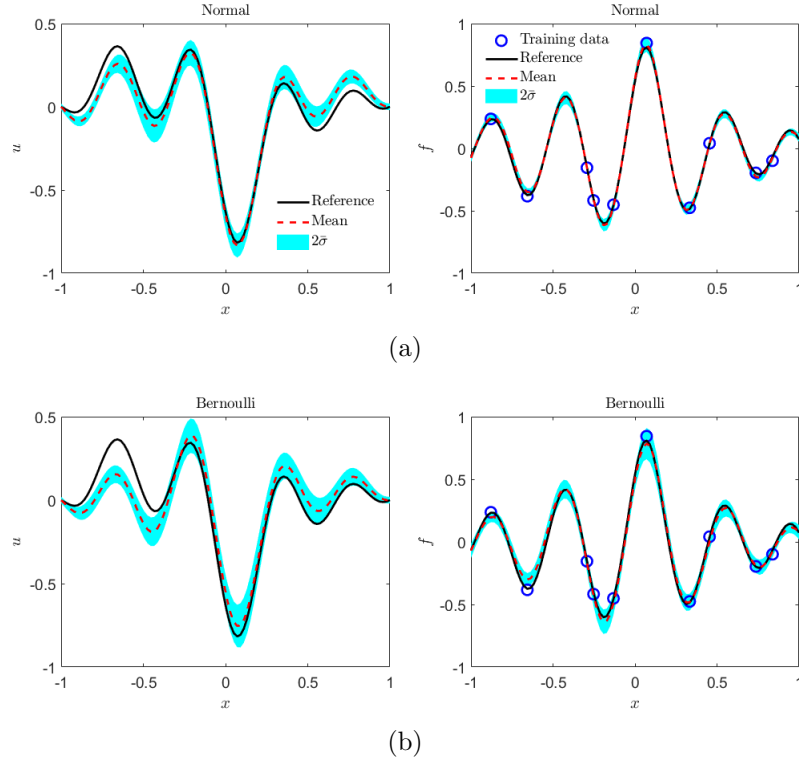


Figure E.2: Nonlinear diffusion-reaction problem: Predictions for u and f from VI with different parameterized posteriors. (a) Mean-field Gaussian, (b) Mean-field Bernoulli.

We proceed to show the customization of posterior estimate in Variational using the example of a forward problem governed by (E.2). In particular, $D = 0.01$, $k_r = 0.2$, and the boundary condition is $u(-1) = u(1) = 0$. In addition, we assume

that we have 10 noisy measurements of f which are randomly distributed at $x \in [-1, 1]$. The noise for the measurements, which is $N(0, 0.01^2)$, is assumed to be known. The objective is to infer u given data of f . We employ a PI-GAN, which is the same as in [201], to solve this problem. The same historical data in [201] is utilized here to train the PI-GAN to obtain the functional priors for u and f . Note that HMC is employed in [201] for the posterior estimation of the parameters in PI-GAN. Here we utilize VI with two different parameterized posterior distributions to demonstrate the flexibility of NeuralUQ, i.e., independent Gaussian with unknown means and standard deviations (mean-field Gaussian), and independent Bernoulli distributions with fixed $p = 0.5$ as the probability of success/failure, multiplied by unknown variables (mean-field Bernoulli). The predictions for u and f are displayed in Figure E.2, it is observed that (1) the predicted means for u and f from the two parameterized posteriors are similar, and (2) the errors between the predicted means for u and f are mostly bounded by the predicted uncertainties in both methods.

Finally, note that we employ these specific prior/posterior distributions for demonstration purposes. Interested users of NeuralUQ can also customize other problem-dependent prior/posterior distributions by following the examples presented above.

E.4 Data generation

Regarding data generation, in Section 6.4.1, the nonstiff differential equations solver *ode45* in Matlab is utilized for solving (6.9) to generate the training data as well as the reference solutions; in Section 6.4.3, the training data are generated from the exact solution of u from [21]; and in Section 6.4.4, the finite-element-based Partial

Differential Equation Toolbox in Matlab is utilized for solving (6.16) to generate the training data as well as the reference solutions. Note that all data as well as the employed NNs related to this study can be accessed in the GitHub repository of NeuralUQ.

Bibliography

- [1] M. ABADI, A. AGARWAL, P. BARHAM, AND ET. AL., *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. Software available from tensorflow.org.
- [2] M. ABDAR, F. POURPANAH, S. HUSSAIN, D. REZAZADEGAN, L. LIU, M. GHAVAMZADEH, P. FIEGUTH, X. CAO, A. KHOSRAVI, U. R. ACHARYA, ET AL., *A review of uncertainty quantification in deep learning: Techniques, applications and challenges*, Information Fusion, 76 (2021), pp. 243–297.
- [3] M. J. ABLOWITZ AND A. ZEPPETELLA, *Explicit solutions of Fisher’s equation for a special wave speed*, Bulletin of Mathematical Biology, 41 (1979), pp. 835–840.
- [4] B. ADLAM, J. LEE, L. XIAO, J. PENNINGTON, AND J. SNOEK, *Exploring the uncertainty properties of neural networks’ implicit priors in the infinite-width limit*, arXiv preprint arXiv:2010.07355, (2020).
- [5] H. AHN, S. CHA, D. LEE, AND T. MOON, *Uncertainty-based continual learning with adaptive regularization*, Advances in neural information processing systems, 32 (2019).
- [6] L. AITCHISON, A. YANG, AND S. W. OBER, *Deep kernel processes*, in Proceedings of the 38th International Conference on Machine Learning, PMLR, 2021, pp. 130–140.
- [7] M. ALBER, A. BUGANZA TEPOLE, W. R. CANNON, S. DE, S. DURABERNAL, K. GARIKIPATI, G. E. KARNIADAKIS, W. 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, 2 (2019), pp. 1–11.
- [8] A. AMINI, W. SCHWARTING, A. SOLEIMANY, AND D. RUS, *Deep evidential regression*, Advances in Neural Information Processing Systems, 33 (2020), pp. 14927–14937.
- [9] C. ANDRIEU AND J. THOMS, *A tutorial on adaptive MCMC*, Statistics and computing, 18 (2008), pp. 343–373.

- [10] A. N. ANGELOPOULOS AND S. BATES, *Conformal prediction: A gentle introduction*, Foundations and Trends® in Machine Learning, 16 (2023), pp. 494–591.
- [11] H. ARBABI AND T. SAPSIS, *Generative stochastic modeling of strongly nonlinear flows with non-gaussian statistics*, SIAM/ASA Journal on Uncertainty Quantification, 10 (2022), pp. 555–583.
- [12] S. ASGHAR, Q.-X. PEI, G. VOLPE, AND R. NI, *Efficient rare event sampling with unsupervised normalizing flows*, Nature Machine Intelligence, (2024), pp. 1–12.
- [13] G. ATKINSON AND A. M. NEVILL, *Statistical methods for assessing measurement error (reliability) in variables relevant to sports medicine*, Sports medicine, 26 (1998), pp. 217–238.
- [14] S. ATKINSON, *Bayesian hidden physics models: Uncertainty quantification for discovery of nonlinear partial differential operators from data*, arXiv preprint arXiv:2006.04228, (2020).
- [15] H. BABAEI, C. BASTIDAS, M. DEFILIPPO, C. CHRYSOSTOMIDIS, AND G. KARNIAKAKIS, *A multifidelity framework and uncertainty quantification for sea surface temperature in the Massachusetts and Cape Cod Bays*, Earth and Space Science, 7 (2020), p. e2019EA000954.
- [16] B. BAHMANI AND W. SUN, *Training multi-objective/multi-task collocation physics-informed neural network with student/teachers transfer learnings*, arXiv preprint arXiv:2107.11496, (2021).
- [17] C. BAJAJ, L. MCLENNAN, T. ANDEEN, AND A. ROY, *Recipes for when physics fails: Recovering robust learning of physics informed neural networks*, Machine Learning: Science and Technology, 4 (2023), p. 015013.
- [18] A. BANASZUK, V. A. FONOBEROV, T. A. FREWEN, M. KOBILAROV, G. MATHEW, I. MEZIC, A. PINTO, T. SAHAI, H. SANE, A. SPERANZON, ET AL., *Scalable approach to uncertainty quantification and robust design of interconnected dynamical systems*, Annual Reviews in Control, 35 (2011), pp. 77–98.
- [19] D. BARBER AND C. M. BISHOP, *Ensemble learning in Bayesian neural networks*, Nato ASI Series F Computer and Systems Sciences, 168 (1998), pp. 215–238.
- [20] M. BATTY, *Digital twins*, 2018.
- [21] N. BENES, A. KASMAN, AND K. YOUNG, *On decompositions of the KdV 2-soliton*, Journal of Nonlinear Science, 16 (2006), pp. 179–200.
- [22] L. BERGER AND M. MARINACCI, *Model uncertainty in climate change economics: A review and proposed framework for future research*, Environmental and Resource Economics, 77 (2020), pp. 475–501.

- [23] M. BETANCOURT, *A conceptual introduction to Hamiltonian Monte Carlo*, arXiv preprint arXiv:1701.02434, (2017).
- [24] D. M. BLEI, A. KUCUKELBIR, AND J. D. MCAULIFFE, *Variational inference: A review for statisticians*, Journal of the American Statistical Association, 112 (2017), pp. 859–877.
- [25] C. BLUNDELL, J. CORNEBISE, K. KAVUKCUOGLU, AND D. WIERSTRA, *Weight uncertainty in neural network*, in International Conference on Machine Learning, PMLR, 2015, pp. 1613–1622.
- [26] L. BONATI, G. PICCINI, AND M. PARRINELLO, *Deep learning the slow modes for rare events sampling*, Proceedings of the National Academy of Sciences, 118 (2021), p. e2113533118.
- [27] J. BRADBURY, R. FROSTIG, P. HAWKINS, M. J. JOHNSON, C. LEARY, D. MACLAURIN, G. NECULA, A. PASZKE, J. VANDERPLAS, S. WANDERMAN-MILNE, AND Q. ZHANG, *JAX: composable transformations of Python+NumPy programs*, 2018.
- [28] A. P. BRNA, R. C. BROWN, P. M. CONNOLLY, S. B. SIMONS, R. E. SHIMIZU, AND M. AGUILAR-SIMON, *Uncertainty-based modulation for lifelong learning*, Neural Networks, 120 (2019), pp. 129–142.
- [29] 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 (2016), pp. 3932–3937.
- [30] J. A. BUCKLEW AND J. BUCKLEW, *Introduction to rare event simulation*, vol. 5, Springer, 2004.
- [31] S. CAI, J. LIANG, Q. GAO, C. XU, AND R. WEI, *Particle image velocimetry based on a deep learning motion estimator*, IEEE Transactions on Instrumentation and Measurement, 69 (2019), pp. 3538–3554.
- [32] S. CAI, Z. MAO, Z. WANG, M. YIN, AND G. E. KARNIADAKIS, *Physics-informed neural networks (PINNs) for fluid mechanics: A review*, Acta Mechanica Sinica, 37 (2021), pp. 1727–1738.
- [33] S. CAI, Z. WANG, F. FUEST, Y. J. JEON, C. GRAY, AND G. E. KARNIADAKIS, *Flow over an espresso cup: inferring 3-D velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks*, Journal of Fluid Mechanics, 915 (2021).
- [34] S. CAI, Z. WANG, S. WANG, P. PERDIKARIS, AND G. E. KARNIADAKIS, *Physics-informed neural networks for heat transfer problems*, Journal of Heat Transfer, 143 (2021), p. 060801.
- [35] S. CAI, S. ZHOU, C. XU, AND Q. GAO, *Dense motion estimation of particle images via a convolutional neural network*, Experiments in Fluids, 60 (2019), pp. 1–16.

- [36] G. C. CALAFIORE, C. NOVARA, AND C. POSSIERI, *A time-varying SIRD model for the COVID-19 contagion in Italy*, Annual Reviews in Control, 50 (2020), pp. 361–372.
- [37] J. CALDEIRA AND B. NORD, *Deeply uncertain: Comparing methods of uncertainty quantification in deep learning algorithms*, Machine Learning: Science and Technology, 2 (2020), p. 015002.
- [38] J. Q. CANDELA AND O. WINTHER, *Incremental Gaussian processes*, Advances in Neural Information Processing Systems, 15 (2002).
- [39] R. CARUANA, *Multitask learning: A knowledge-based source of inductive bias*¹, in Proceedings of the Tenth International Conference on Machine Learning, Citeseer, 1993, pp. 41–48.
- [40] R. CARUANA, *Multitask learning*, Machine learning, 28 (1997), pp. 41–75.
- [41] A. CHANDRA, M. CURTI, K. TIELS, E. A. LOMONOVA, AND D. M. TARTAKOVSKY, *Physics-informed neural networks for modelling anisotropic and bi-anisotropic electromagnetic constitutive laws through indirect data*, in 2022 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2022, pp. 1451–1459.
- [42] A. CHARALAMPOPOULOS AND T. SAPSIS, *Uncertainty quantification of turbulent systems via physically consistent and data-informed reduced-order models*, Physics of Fluids, 34 (2022).
- [43] B. CHARPENTIER, O. BORCHERT, D. ZÜGNER, S. GEISLER, AND S. GÜNNEMANN, *Natural posterior network: Deep Bayesian uncertainty for exponential family distributions*, arXiv preprint arXiv:2105.04471, (2021).
- [44] B. CHARPENTIER, D. ZÜGNER, AND S. GÜNNEMANN, *Posterior network: Uncertainty estimation without OOD samples via density-based pseudo-counts*, Advances in Neural Information Processing Systems, 33 (2020), pp. 1356–1367.
- [45] F. CHEN, D. SONDAK, P. PROTOPAPAS, M. MATTHEAKIS, S. LIU, D. AGARWAL, AND M. DI GIOVANNI, *NeuroDiffEq: A Python package for solving differential equations with neural networks*, Journal of Open Source Software, 5 (2020), p. 1931.
- [46] P. CHEN, T. MENG, Z. ZOU, J. DARBON, AND G. E. KARNIADAKIS, *Leveraging Hamilton-Jacobi PDEs with time-dependent Hamiltonians for continual scientific machine learning*, in 6th Annual Learning for Dynamics & Control Conference, PMLR, 2024, pp. 1–12.
- [47] —, *Leveraging multitime Hamilton–Jacobi PDEs for certain scientific machine learning problems*, SIAM Journal on Scientific Computing, 46 (2024), pp. C216–C248.
- [48] R. T. CHEN, Y. RUBANOVA, J. BETTENCOURT, AND D. K. DUVENAUD, *Neural ordinary differential equations*, Advances in Neural Information Processing Systems, 31 (2018).

- [49] Y. CHEN, B. HOSSEINI, H. OWHADI, AND A. M. STUART, *Solving and learning nonlinear PDEs with Gaussian processes*, Journal of Computational Physics, 447 (2021), p. 110668.
- [50] Y. CHEN, H. OWHADI, AND F. SCHÄFER, *Sparse Cholesky factorization for solving nonlinear PDEs via Gaussian processes*, Mathematics of Computation, (2024).
- [51] Y.-C. CHEN, P.-E. LU, C.-S. CHANG, AND T.-H. LIU, *A time-dependent SIR model for COVID-19 with undetectable infected persons*, IEEE Transactions on Network Science and Engineering, 7 (2020), pp. 3279–3294.
- [52] Z. CHEN, Y. LIU, AND H. SUN, *Physics-informed learning of governing equations from scarce data*, Nature Communications, 12 (2021), p. 6136.
- [53] Z. CHEN AND D. XIU, *On generalized residual network for deep learning of unknown dynamical systems*, Journal of Computational Physics, 438 (2021), p. 110362.
- [54] C.-A. CHENG AND B. BOOTS, *Incremental variational sparse Gaussian process regression*, Advances in Neural Information Processing Systems, 29 (2016).
- [55] M. CHUA, D. KIM, J. CHOI, N. G. LEE, V. DESHPANDE, J. SCHWAB, M. H. LEV, R. G. GONZALEZ, M. S. GEE, AND S. DO, *Tackling prediction uncertainty in machine learning for healthcare*, Nature Biomedical Engineering, 7 (2023), pp. 711–718.
- [56] Y. CHUNG, I. CHAR, H. GUO, J. SCHNEIDER, AND W. NEISWANGER, *Uncertainty toolbox: An open-source library for assessing, visualizing, and improving uncertainty quantification*, arXiv preprint arXiv:2109.10254, (2021).
- [57] K. CIOSEK, V. FORTUIN, R. TOMIOKA, K. HOFMANN, AND R. TURNER, *Conservative uncertainty estimation by fitting prior networks*, in International Conference on Learning Representations, 2019.
- [58] M. CRANMER, *Interpretable machine learning for science with PySR and SymbolicRegression.jl*, arXiv preprint arXiv:2305.01582, (2023).
- [59] L. CSATÓ AND M. OPPER, *Sparse on-line Gaussian processes*, Neural Computation, 14 (2002), pp. 641–668.
- [60] S. CUOMO, V. S. DI COLA, F. GIAMPAOLO, G. ROZZA, M. RAISSI, AND F. PICCIALLI, *Scientific machine learning through physics-informed neural networks: Where we are and what’s next*, Journal of Scientific Computing, 92 (2022), p. 88.
- [61] A. DAMIANOU AND N. LAWRENCE, *Deep Gaussian processes*, in Artificial Intelligence and Statistics, 2013, pp. 207–215.
- [62] F. D’ANGELO AND V. FORTUIN, *Repulsive deep ensembles are Bayesian*, Advances in Neural Information Processing Systems, 34 (2021), pp. 3451–3465.

- [63] J. DARBON AND G. P. LANGLOIS, *On Bayesian posterior mean estimators in imaging sciences and Hamilton–Jacobi partial differential equations*, Journal of Mathematical Imaging and Vision, 63 (2021), pp. 821–854.
- [64] J. DARBON, G. P. LANGLOIS, AND T. MENG, *Overcoming the curse of dimensionality for some Hamilton–Jacobi partial differential equations via neural network architectures*, Research in the Mathematical Sciences, 7 (2020), pp. 1–50.
- [65] J. DARBON AND T. MENG, *On some neural network architectures that can represent viscosity solutions of certain high dimensional Hamilton–Jacobi partial differential equations*, Journal of Computational Physics, 425 (2021), p. 109907.
- [66] A. DAW, M. MARUF, AND A. KARPATNE, *PID-GAN: A GAN framework based on a physics-informed discriminator for uncertainty quantification with physics*, in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 237–247.
- [67] M. DE FLORIO, Z. ZOU, D. E. SCHIAVAZZI, AND G. E. KARNIADAKIS, *Quantification of total uncertainty in the physics-informed reconstruction of CVSim-6 physiology*, ArXiv, (2024).
- [68] M. V. DE HOOP, N. B. KOVACHKI, N. H. NELSEN, AND A. M. STUART, *Convergence rates for learning linear operators from noisy data*, SIAM/ASA Journal on Uncertainty Quantification, 11 (2023), pp. 480–513.
- [69] R. DE NEUFVILLE, K. HODOTA, J. SUSSMAN, AND S. SCHOLTES, *Real options to increase the value of intelligent transportation systems*, Transportation Research Record, 2086 (2008), pp. 40–47.
- [70] P. DELLAPORTAS AND D. A. STEPHENS, *Bayesian analysis of errors-in-variables regression models*, Biometrics, (1995), pp. 1085–1095.
- [71] S. DESAI, M. MATTHEAKIS, H. JOY, P. PROTOPAPAS, AND S. J. ROBERTS, *One-shot transfer learning of physics-informed neural networks*, in ICML 2022 2nd AI for Science Workshop, 2022.
- [72] N. DEWOLF, B. D. BAETS, AND W. WAEGEMAN, *Valid prediction intervals for regression problems*, Artificial Intelligence Review, 56 (2023), pp. 577–613.
- [73] J. V. DILLON, I. LANGMORE, D. TRAN, E. BREVDO, S. VASUDEVAN, D. MOORE, B. PATTON, A. ALEMI, M. HOFFMAN, AND R. A. SAUROUS, *TensorFlow Distributions*, arXiv preprint arXiv:1711.10604, (2017).
- [74] L. DINH, J. SOHL-DICKSTEIN, AND S. BENGIO, *Density estimation using Real NVP*, in International Conference on Learning Representations, 2017.
- [75] E. DONG, H. DU, AND L. GARDNER, *An interactive web-based dashboard to track COVID-19 in real time*, The Lancet Infectious Diseases, 20 (2020), pp. 533–534.

- [76] B. EASTMAN, L. PODINA, AND M. KOHANDEL, *A PINN approach to symbolic differential operator discovery with sparse data*, in *The Symbiosis of Deep Learning and Differential Equations II*, 2022.
- [77] M. R. EBERS, K. M. STEELE, AND J. N. KUTZ, *Discrepancy modeling framework: Learning missing physics, modeling systematic residuals, and disambiguating between deterministic and random effects*, *SIAM Journal on Applied Dynamical Systems*, 23 (2024), pp. 440–469.
- [78] S. EBRAHIMI, M. ELHOSEINY, T. DARRELL, AND M. ROHRBACH, *Uncertainty-guided continual learning in Bayesian neural networks—Extended abstract*, in *Proc. IEEE Conf. Comput. Vis. Pattern Recognition (CVPR)*, 2018.
- [79] L. C. EVANS, *Partial Differential Equations*, vol. 19, American Mathematical Society, 2 ed., 2010.
- [80] L. C. EVANS AND P. E. SOUGANIDIS, *Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations*, *Indiana University Mathematics Journal*, 33 (1984), pp. 773–797.
- [81] G. FABBRI, F. GOZZI, AND A. SWIECH, *Stochastic optimal control in infinite dimension*, *Probability and Stochastic Modelling*. Springer, (2017).
- [82] J. FAN AND Y. K. TRUONG, *Nonparametric regression with errors in variables*, *The Annals of Statistics*, (1993), pp. 1900–1925.
- [83] U. FASEL, J. N. KUTZ, B. W. BRUNTON, AND S. L. BRUNTON, *Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control*, *Proceedings of the Royal Society A*, 478 (2022), p. 20210904.
- [84] S. FORT, H. HU, AND B. LAKSHMINARAYANAN, *Deep ensembles: A loss landscape perspective*, *arXiv preprint arXiv:1912.02757*, (2019).
- [85] V. FORTUIN, *Priors in bayesian deep learning: A review*, *International Statistical Review*, 90 (2022), pp. 563–591.
- [86] J. N. FUHG, I. KALOGERIS, A. FAU, AND N. BOUKLAS, *Interval and fuzzy physics-informed neural networks for uncertain fields*, *Probabilistic Engineering Mechanics*, 68 (2022), p. 103240.
- [87] J. C. FYFE, V. V. KHARIN, B. D. SANTER, J. N. COLE, AND N. P. GILLETT, *Significant impact of forcing uncertainty in a large ensemble of climate model simulations*, *Proceedings of the National Academy of Sciences*, 118 (2021), p. e2016549118.
- [88] Y. GAL, *Uncertainty in deep learning*, University of Cambridge, 1 (2016).
- [89] Y. GAL AND Z. GHAHRAMANI, *Dropout as a Bayesian approximation: Representing model uncertainty in deep learning*, in *International Conference on Machine Learning*, 2016, pp. 1050–1059.

- [90] Y. GAO AND M. K. NG, *Wasserstein generative adversarial uncertainty quantification in physics-informed neural networks*, Journal of Computational Physics, 463 (2022), p. 111270.
- [91] J. R. GARDNER, G. PLEISS, D. BINDEL, K. Q. WEINBERGER, AND A. G. WILSON, *GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration*, in Advances in Neural Information Processing Systems, 2018.
- [92] S. GARG AND S. CHAKRABORTY, *VB-DeepONet: A Bayesian operator learning framework for uncertainty quantification*, Engineering Applications of Artificial Intelligence, 118 (2023), p. 105685.
- [93] T. GARIPOV, P. IZMAILOV, D. PODOPRIKHIN, D. P. VETROV, AND A. G. WILSON, *Loss surfaces, mode connectivity, and fast ensembling of DNNs*, in Advances in Neural Information Processing Systems, 2018, pp. 8789–8798.
- [94] J. GAWLIKOWSKI, C. R. N. TASSI, M. ALI, J. LEE, M. HUMT, J. FENG, A. KRUSPE, R. TRIEBEL, P. JUNG, R. ROSCHER, ET AL., *A survey of uncertainty in deep neural networks*, Artificial Intelligence Review, 56 (2023), pp. 1513–1589.
- [95] A. GELMAN, J. B. CARLIN, H. S. STERN, AND D. B. RUBIN, *Bayesian data analysis*, Chapman and Hall/CRC, 1995.
- [96] N. GENEVA AND N. ZABARAS, *Multi-fidelity generative deep learning turbulent flows*, Foundations of Data Science, 2 (2020), pp. 391–428.
- [97] M. GERMAIN, K. GREGOR, I. MURRAY, AND H. LAROCHELLE, *MADE: Masked autoencoder for distribution estimation*, in International Conference on Machine Learning, PMLR, 2015, pp. 881–889.
- [98] R. G. GHANEM AND P. D. SPANOS, *Stochastic Finite Elements: A Spectral Approach*, Dover Publications, 2003.
- [99] A. GIRARD, C. RASMUSSEN, J. Q. CANDELA, AND R. MURRAY-SMITH, *Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting*, Advances in Neural Information Processing systems, 15 (2002).
- [100] L. J. GLEESER, *Estimation in a multivariate “errors in variables” regression model: Large sample results*, The Annals of Statistics, (1981), pp. 24–44.
- [101] X. GLOROT AND Y. BENGIO, *Understanding the difficulty of training deep feedforward neural networks*, in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
- [102] E. GOAN AND C. FOOKES, *Bayesian neural networks: An introduction and survey*, Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018, (2020), pp. 45–87.

- [103] R. GONZÁLEZ-GARCÍA, R. RICO-MARTÍNEZ, AND I. G. KEVREKIDIS, *Identification of distributed parameter systems: A neural net based approach*, Computers & Chemical Engineering, 22 (1998), pp. S965–S968.
- [104] I. GOODFELLOW, Y. BENGIO, AND A. COURVILLE, *Deep Learning*, vol. 1, MIT Press, 2016.
- [105] I. GOODFELLOW, J. POUGET-ABADIE, M. MIRZA, B. XU, D. WARDEFARLEY, S. OZAIR, A. COURVILLE, AND Y. BENGIO, *Generative adversarial networks*, Communications of the ACM, 63 (2020), pp. 139–144.
- [106] S. N. GOSLING, G. R. MCGREGOR, AND J. A. LOWE, *The benefits of quantifying climate model uncertainty in climate change impacts assessment: An example with heat-related mortality change estimates*, Climatic Change, 112 (2012), pp. 217–231.
- [107] S. GOSWAMI, M. YIN, Y. YU, AND G. E. KARNIADAKIS, *A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials*, Computer Methods in Applied Mechanics and Engineering, 391 (2022), p. 114587.
- [108] R. B. GRAMACY AND H. K. LEE, *Adaptive design and analysis of supercomputer experiments*, Technometrics, 51 (2009), pp. 130–145.
- [109] A. GRAVES, *Practical variational inference for neural networks*, in Advances in Neural Information Processing Systems, 2011, pp. 2348–2356.
- [110] C. GRIGO AND P.-S. KOUTSOURELAKIS, *A physics-aware, probabilistic machine learning framework for coarse-graining high-dimensional systems in the Small Data regime*, Journal of Computational Physics, 397 (2019), p. 108842.
- [111] L. GUO, H. WU, X. YU, AND T. ZHOU, *Monte Carlo fPINNs: Deep learning method for forward and inverse problems involving high dimensional fractional partial differential equations*, Computer Methods in Applied Mechanics and Engineering, 400 (2022), p. 115523.
- [112] L. GUO, H. WU, AND T. ZHOU, *Normalizing field flows: Solving forward and inverse stochastic differential equations using physics-informed flow models*, Journal of Computational Physics, 461 (2022), p. 111202.
- [113] W. HACKBUSCH, *Multi-grid methods and applications*, vol. 4, Springer Science & Business Media, 2013.
- [114] E. HAGHIGHAT AND R. JUANES, *SciANN: A keras/tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks*, Computer Methods in Applied Mechanics and Engineering, 373 (2021), p. 113552.
- [115] E. J. HALL, S. TAVERNIERS, M. A. KATSOULAKIS, AND D. M. TARTAKOVSKY, *GINNs: Graph-informed neural networks for multiscale physics*, Journal of Computational Physics, 433 (2021), p. 110192.
- [116] B. HAMZI AND H. OWHADI, *Learning dynamical systems from data: A simple cross-validation perspective, part i: Parametric kernel flows*, Physica D: Nonlinear Phenomena, 421 (2021), p. 132817.

- [117] J. HAN, A. JENTZEN, AND W. E, *Solving high-dimensional partial differential equations using deep learning*, Proceedings of the National Academy of Sciences, 115 (2018), pp. 8505–8510.
- [118] C. R. HARRIS, K. J. MILLMAN, S. J. VAN DER WALT, R. GOMMERS, P. VIRTANEN, D. COURNAPEAU, E. WIESER, J. TAYLOR, S. BERG, N. J. SMITH, ET AL., *Array programming with NumPy*, Nature, 585 (2020), pp. 357–362.
- [119] B. HE, B. LAKSHMINARAYANAN, AND Y. W. TEH, *Bayesian deep ensembles via the neural tangent kernel*, Advances in Neural Information Processing Systems, 33 (2020), pp. 1010–1022.
- [120] Y. HE AND D. XIU, *Numerical strategy for model correction using physical constraints*, Journal of Computational Physics, 313 (2016), pp. 617–634.
- [121] O. HENNIGH, S. NARASIMHAN, M. A. NABIAN, A. SUBRAMANIAM, K. TANGSALI, Z. FANG, M. RIETMANN, W. BYEON, AND S. CHOUDHRY, *NVIDIA SimNet™: An AI-accelerated multi-physics simulation framework*, in International Conference on Computational Science, Springer, 2021, pp. 447–461.
- [122] C. F. HIGHAM AND D. J. HIGHAM, *Deep learning: An introduction for applied mathematicians*, SIAM Review, 61 (2019), pp. 860–891.
- [123] J. HO, A. JAIN, AND P. ABBEEL, *Denoising diffusion probabilistic models*, Advances in Neural Information Processing Systems, 33 (2020), pp. 6840–6851.
- [124] M. D. HOFFMAN, D. M. BLEI, C. WANG, AND J. PAISLEY, *Stochastic variational inference*, The Journal of Machine Learning Research, 14 (2013), pp. 1303–1347.
- [125] M. D. HOFFMAN, A. GELMAN, ET AL., *The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo*, J. Mach. Learn. Res., 15 (2014), pp. 1593–1623.
- [126] L. HOFFMANN AND C. ELSTER, *Deep ensembles from a Bayesian perspective*, arXiv preprint arXiv:2105.13283, (2021).
- [127] R. HU AND M. LAURIERE, *Recent developments in machine learning methods for stochastic control and games*, arXiv preprint arXiv:2303.10257, (2023).
- [128] D. Z. HUANG, K. XU, C. FARHAT, AND E. DARVE, *Learning constitutive relations from indirect observations using deep neural networks*, Journal of Computational Physics, 416 (2020), p. 109491.
- [129] G. HUANG, Y. LI, G. PLEISS, Z. LIU, J. E. HOPCROFT, AND K. Q. WEINBERGER, *Snapshot Ensembles: Train 1, Get M for Free*, in International Conference on Learning Representations, 2017.
- [130] M. F. HUBER, *Recursive Gaussian process: On-line regression and learning*, Pattern Recognition Letters, 45 (2014), pp. 85–91.

- [131] E. HÜLLERMEIER AND W. WAEGEMAN, *Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods*, Machine Learning, 110 (2021), pp. 457–506.
- [132] T. M. INC., *Matlab version: 9.13.0 (r2022b)*, 2022.
- [133] P. IZMAILOV, S. VIKRAM, M. D. HOFFMAN, AND A. G. G. WILSON, *What are Bayesian neural network posteriors really like?*, in International Conference on Machine Learning, PMLR, 2021, pp. 4629–4640.
- [134] A. D. JAGTAP AND G. E. KARNIADAKIS, *Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations*, Communications in Computational Physics, 28 (2020), pp. 2002–2041.
- [135] A. D. JAGTAP, E. KHARAZMI, AND G. E. KARNIADAKIS, *Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems*, Computer Methods in Applied Mechanics and Engineering, 365 (2020), p. 113028.
- [136] X. JIANG, X. WANG, Z. WEN, E. LI, AND H. WANG, *Practical uncertainty quantification for space-dependent inverse heat conduction problem via ensemble physics-informed neural networks*, International Communications in Heat and Mass Transfer, 147 (2023), p. 106940.
- [137] X. JIANG, X. WANG, Z. WEN, AND H. WANG, *Resolution-independent generative models based on operator learning for physics-constrained Bayesian inverse problems*, Computer Methods in Applied Mechanics and Engineering, 420 (2024), p. 116690.
- [138] P. JIN, S. MENG, AND L. LU, *MIONet: Learning multiple-input operators via tensor product*, SIAM Journal on Scientific Computing, 44 (2022), pp. A3490–A3514.
- [139] L. V. JOSPIN, H. LAGA, F. BOUSSAID, W. BUNTINE, AND M. BEN-NAMOUN, *Hands-on Bayesian neural networks—A tutorial for deep learning users*, IEEE Computational Intelligence Magazine, 17 (2022), pp. 29–48.
- [140] S. JUNEJA AND P. SHAHABUDDIN, *Rare-event simulation techniques: An introduction and recent advances*, Handbooks in Operations Research and Management Science, 13 (2006), pp. 291–350.
- [141] H. D. KABIR, A. KHOSRAVI, M. A. HOSEN, AND S. NAHAVANDI, *Neural network-based uncertainty quantification: A survey of methodologies and applications*, IEEE access, 6 (2018), pp. 36218–36234.
- [142] G. E. KARNIADAKIS, I. G. KEVREKIDIS, L. LU, P. PERDIKARIS, S. WANG, AND L. YANG, *Physics-informed machine learning*, Nature Reviews Physics, 3 (2021), pp. 422–440.
- [143] S. KARUMURI, R. TRIPATHY, I. BILIONIS, AND J. PANCHAL, *Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks*, Journal of Computational Physics, 404 (2020), p. 109120.

- [144] K. KASHINATH, M. MUSTAFA, A. ALBERT, J. WU, C. JIANG, S. ES-MAEILZADEH, K. AZIZZADENESHELI, R. WANG, A. CHATTOPADHYAY, A. SINGH, ET AL., *Physics-informed machine learning: Case studies for weather and climate modelling*, Philosophical Transactions of the Royal Society A, 379 (2021), p. 20200093.
- [145] R. E. KASS, *The validity of posterior expansions based on Laplace's method*, Bayesian and Likelihood Methods in Statistics and Econometrics, (1990), pp. 473–488.
- [146] R. KASTNER, J. MATAI, AND S. NEUENDORFFER, *Parallel programming for FPGAs*, arXiv preprint arXiv:1805.03648, (2018).
- [147] D. KAVETSKI, G. KUCZERA, AND S. W. FRANKS, *Bayesian analysis of input uncertainty in hydrological modeling: 1. theory*, Water Resources Research, 42 (2006).
- [148] —, *Bayesian analysis of input uncertainty in hydrological modeling: 2. application*, Water Resources Research, 42 (2006).
- [149] M. KHAN, D. NIELSEN, V. TANGKARATT, W. LIN, Y. GAL, AND A. SRIVASTAVA, *Fast and scalable bayesian deep learning by weight-perturbation in Adam*, in International Conference on Machine Learning, PMLR, 2018, pp. 2611–2620.
- [150] E. KHARAZMI, M. CAI, X. ZHENG, Z. ZHANG, G. LIN, AND G. E. KARNIADAKIS, *Identifiability and predictability of integer-and fractional-order epidemiological models using physics-informed neural networks*, Nature Computational Science, 1 (2021), pp. 744–753.
- [151] E. KHARAZMI, Z. ZHANG, AND G. E. KARNIADAKIS, *hp-VPINNs: Variational physics-informed neural networks with domain decomposition*, Computer Methods in Applied Mechanics and Engineering, 374 (2021), p. 113547.
- [152] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).
- [153] D. P. KINGMA, T. SALIMANS, R. JOZEFOWICZ, X. CHEN, I. SUTSKEVER, AND M. WELLING, *Improved variational inference with inverse autoregressive flow*, Advances in Neural Information Processing Systems, 29 (2016).
- [154] D. P. KINGMA AND M. WELLING, *Auto-Encoding Variational Bayes*, in 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, 2014.
- [155] J. KIRKPATRICK, R. PASCANU, N. RABINOWITZ, J. VENESS, G. DESJARDINS, A. A. RUSU, K. MILAN, J. QUAN, T. RAMALHO, A. GRABSKA-BARWINSKA, D. HASSABIS, C. CLOPATH, D. KUMARAN, AND R. HADSELL, *Overcoming catastrophic forgetting in neural networks*, Proceedings of the National Academy of Sciences, 114 (2017), pp. 3521–3526.

- [156] E. KIYANI, K. SHUKLA, G. E. KARNIADAKIS, AND M. KARTTUNEN, *A framework based on symbolic regression coupled with eXtended physics-informed neural networks for gray-box learning of equations of motion from data*, Computer Methods in Applied Mechanics and Engineering, 415 (2023), p. 116258.
- [157] I. KOBYZEV, S. J. PRINCE, AND M. A. BRUBAKER, *Normalizing flows: An introduction and review of current methods*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 43 (2020), pp. 3964–3979.
- [158] A.-K. KOPETZKI, B. CHARPENTIER, D. ZÜGNER, S. GIRI, AND S. GÜNNEMANN, *Evaluating robustness of predictive uncertainty estimation: Are Dirichlet-based models reliable?*, in International Conference on Machine Learning, PMLR, 2021, pp. 5707–5718.
- [159] A. KORYAGIN, R. KHUDOROZKOV, AND S. TSIMFER, *PyDEns: A python framework for solving differential equations with neural networks*, arXiv preprint arXiv:1909.11544, (2019).
- [160] N. B. KOVACHKI, Z. LI, B. LIU, K. AZIZZADENESHELI, K. BHATTACHARYA, A. M. STUART, AND A. ANANDKUMAR, *Neural operator: Learning maps between function spaces with applications to PDEs.*, J. Mach. Learn. Res., 24 (2023), pp. 1–97.
- [161] V. KULESHOV, N. FENNER, AND S. ERMON, *Accurate uncertainties for deep learning using calibrated regression*, in International Conference on Machine Learning, PMLR, 2018, pp. 2796–2804.
- [162] I. LAGARIS, A. LIKAS, AND D. FOTIADIS, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE Transactions on Neural Networks, 9 (1998), pp. 987–1000.
- [163] B. LAKSHMINARAYANAN, A. PRITZEL, AND C. BLUNDELL, *Simple and scalable predictive uncertainty estimation using deep ensembles*, in Advances in Neural Information Processing Systems, 2017, pp. 6402–6413.
- [164] J. LAMPINEN AND A. VEHTARI, *Bayesian approach for neural networks—review and case studies*, Neural Networks, 14 (2001), pp. 257–274.
- [165] S. LANTHALER, S. MISHRA, AND G. E. KARNIADAKIS, *Error estimates for DeepONets: A deep learning framework in infinite dimensions*, Transactions of Mathematics and Its Applications, 6 (2022), p. tnac001.
- [166] J. LAO, C. SUTER, I. LANGMORE, C. CHIMISOV, A. SAXENA, P. SOUNTSOV, D. MOORE, R. A. SAUROUS, M. D. HOFFMAN, AND J. V. DILLON, *tfp.mcmc: Modern Markov chain Monte Carlo tools built for modern hardware*, arXiv preprint arXiv:2002.01184, (2020).
- [167] J. LEE, Y. BAHRI, R. NOVAK, S. S. SCHOENHOLZ, J. PENNINGTON, AND J. SOHL-DICKSTEIN, *Deep neural networks as Gaussian Processes*, in International Conference on Learning Representations, 2018, p. 17.

- [168] K. LEE, N. TRASK, AND P. STINIS, *Structure-preserving sparse identification of nonlinear dynamics for data-driven modeling*, in Mathematical and Scientific Machine Learning, PMLR, 2022, pp. 65–80.
- [169] W. T. LEUNG, G. LIN, AND Z. ZHANG, *NH-PINN: Neural homogenization-based physics-informed neural network for multiscale problems*, Journal of Computational Physics, 470 (2022), p. 111539.
- [170] D. LEVI, L. GISPAN, N. GILADI, AND E. FETAYA, *Evaluating and calibrating uncertainty prediction in regression tasks*, Sensors, 22 (2022), p. 5540.
- [171] Z. LI, N. KOVACHKI, K. AZIZZADENESHELI, B. LIU, K. BHATTACHARYA, A. STUART, AND A. ANANDKUMAR, *Fourier neural operator for parametric partial differential equations*, arXiv preprint arXiv:2010.08895, (2020).
- [172] Z. LI, H. ZHENG, N. KOVACHKI, D. JIN, H. CHEN, B. LIU, K. AZIZZADENESHELI, AND A. ANANDKUMAR, *Physics-informed neural operator for learning partial differential equations*, ACM/JMS Journal of Data Science, 1 (2024), pp. 1–27.
- [173] G. LIN, C. MOYA, AND Z. ZHANG, *B-DeepONet: An enhanced Bayesian DeepONet for solving noisy parametric PDEs using accelerated replica exchange SGLD*, Journal of Computational Physics, 473 (2023), p. 111713.
- [174] G. LIN, Y. WANG, AND Z. ZHANG, *Multi-variance replica exchange SGM-CMC for inverse and forward problems via Bayesian PINN*, Journal of Computational Physics, 460 (2022), p. 111173.
- [175] K. LINKA, A. SCHÄFER, X. MENG, Z. ZOU, G. E. KARNIADAKIS, AND E. KUHL, *Bayesian Physics Informed Neural Networks for real-world nonlinear dynamical systems*, Computer Methods in Applied Mechanics and Engineering, 402 (2022), p. 115346.
- [176] Q. LIU AND D. WANG, *Stein variational gradient descent: A general purpose bayesian inference algorithm*, Advances in Neural Information Processing Systems, 29 (2016).
- [177] A. LOQUERCIO, M. SEGU, AND D. SCARAMUZZA, *A general framework for uncertainty estimation in deep learning*, IEEE Robotics and Automation Letters, 5 (2020), pp. 3153–3160.
- [178] L. LU, P. JIN, G. PANG, Z. ZHANG, AND G. E. KARNIADAKIS, *Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators*, Nature Machine Intelligence, 3 (2021), pp. 218–229.
- [179] L. LU, X. MENG, S. CAI, Z. MAO, S. GOSWAMI, Z. ZHANG, AND G. E. KARNIADAKIS, *A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data*, Computer Methods in Applied Mechanics and Engineering, 393 (2022), p. 114778.
- [180] L. LU, X. MENG, Z. MAO, AND G. E. KARNIADAKIS, *DeepXDE: A deep learning library for solving differential equations*, SIAM Review, 63 (2021), pp. 208–228.

- [181] K. O. LYE, S. MISHRA, AND D. RAY, *Deep learning observables in computational fluid dynamics*, Journal of Computational Physics, 410 (2020), p. 109339.
- [182] Y.-A. MA, T. CHEN, AND E. FOX, *A complete recipe for stochastic gradient MCMC*, in Advances in Neural Information Processing Systems, 2015, pp. 2917–2925.
- [183] D. J. MACKAY, *Information-based objective functions for active data selection*, Neural computation, 4 (1992), pp. 590–604.
- [184] —, *Bayesian methods for backpropagation networks*, in Models of Neural Networks III: Association, Generalization, and Representation, Springer, 1996, pp. 211–254.
- [185] W. J. MADDOX, P. IZMAILOV, T. GARIPPOV, D. P. VETROV, AND A. G. WILSON, *A simple baseline for Bayesian uncertainty in deep learning*, in Advances in Neural Information Processing Systems, 2019, pp. 13153–13164.
- [186] A. J. MAJDA AND D. QI, *Strategies for reduced-order models for predicting the statistical responses and uncertainty quantification in complex turbulent dynamical systems*, SIAM Review, 60 (2018), pp. 491–549.
- [187] S. MALANI, T. S. BERTALAN, T. CUI, J. L. AVALOS, M. BETENBAUGH, AND I. G. KEVREKIDIS, *Some of the variables, some of the parameters, some of the times, with some physics known: Identification with partial information*, Computers & Chemical Engineering, 178 (2023), p. 108343.
- [188] A. MALININ AND M. GALES, *Predictive uncertainty estimation via prior networks*, Advances in Neural Information Processing Systems, 31 (2018).
- [189] S. MANDT, M. D. HOFFMAN, AND D. M. BLEI, *Stochastic gradient descent as approximate Bayesian inference*, The Journal of Machine Learning Research, 18 (2017), pp. 4873–4907.
- [190] Z. MAO, A. D. JAGTAP, AND G. E. KARNIADAKIS, *Physics-informed neural networks for high-speed flows*, Computer Methods in Applied Mechanics and Engineering, 360 (2020), p. 112789.
- [191] A. G. D. G. MATTHEWS, M. ROWLAND, J. HRON, R. E. TURNER, AND Z. GHAHRAMANI, *Gaussian process behaviour in wide deep neural networks*, arXiv preprint arXiv:1804.11271, (2018).
- [192] S. F. MCCORMICK, *Multigrid methods*, SIAM, 1987.
- [193] W. MCENEANEY, *Max-plus methods for nonlinear control and estimation*, Springer Science & Business Media, 2006.
- [194] A. MCHUTCHON AND C. RASMUSSEN, *Gaussian process training with input noise*, Advances in Neural Information Processing Systems, 24 (2011).
- [195] F. MEIER, P. HENNIG, AND S. SCHAAL, *Incremental local Gaussian regression*, Advances in Neural Information Processing Systems, 27 (2014).

- [196] T. MENG, Z. ZOU, J. DARBON, AND G. E. KARNIADAKIS, *HJ-sampler: A Bayesian sampler for inverse problems of a stochastic process by leveraging Hamilton-Jacobi PDEs and score-based generative models*, arXiv preprint arXiv:2409.09614, (2024).
- [197] X. MENG, *Variational inference in neural functional prior using normalizing flows: Application to differential equation and operator learning problems*, Applied Mathematics and Mechanics, 44 (2023), pp. 1111–1124.
- [198] X. MENG, H. BABAEI, AND G. E. KARNIADAKIS, *Multi-fidelity Bayesian neural networks: Algorithms and applications*, Journal of Computational Physics, 438 (2021), p. 110361.
- [199] X. MENG AND G. E. KARNIADAKIS, *A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems*, Journal of Computational Physics, 401 (2020), p. 109020.
- [200] X. MENG, Z. LI, D. ZHANG, AND G. E. KARNIADAKIS, *PPINN: Parareal physics-informed neural network for time-dependent PDEs*, Computer Methods in Applied Mechanics and Engineering, 370 (2020), p. 113250.
- [201] X. MENG, L. YANG, Z. MAO, J. DEL ÁGUILA FERRANDIS, AND G. E. KARNIADAKIS, *Learning functional priors and posteriors from data and physics*, Journal of Computational Physics, 457 (2022), p. 111073.
- [202] S. MIHAI, M. YAQOUB, D. V. HUNG, W. DAVIS, P. TOWAKEL, M. RAZA, M. KARAMANOGLU, B. BARN, D. SHETVE, R. V. PRASAD, ET AL., *Digital twins: A survey on enabling technologies, challenges, trends and future prospects*, IEEE Communications Surveys & Tutorials, 24 (2022), pp. 2255–2291.
- [203] S. MISHRA AND R. MOLINARO, *Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs*, IMA Journal of Numerical Analysis, (2021).
- [204] G. S. MISYRIS, A. VENZKE, AND S. CHATZIVASILEIADIS, *Physics-informed neural networks for power systems*, in 2020 IEEE power & energy society general meeting (PESGM), IEEE, 2020, pp. 1–5.
- [205] J. P. MOLNAR AND S. J. GRAUER, *Flow field tomography with uncertainty quantification using a Bayesian physics-informed neural network*, Measurement Science and Technology, 33 (2022), p. 065305.
- [206] C. MOYA, S. ZHANG, G. LIN, AND M. YUE, *DeepONet-Grid-UQ: A trustworthy deep operator framework for predicting the power grid’s post-fault trajectories*, Neurocomputing, 535 (2023), pp. 166–182.
- [207] Z. NADO, N. BAND, M. COLLIER, J. DJOLONGA, M. W. DUSENBERRY, S. FARQUHAR, Q. FENG, A. FILOS, M. HAVASI, R. JENATTON, ET AL., *Uncertainty baselines: Benchmarks for uncertainty & robustness in deep learning*, arXiv preprint arXiv:2106.04015, (2021).

- [208] K. NATH, X. MENG, D. J. SMITH, AND G. E. KARNIADAKIS, *Physics-informed neural networks for predicting gas flow dynamics and unknown parameters in diesel engines*, Scientific Reports, 13 (2023), p. 13683.
- [209] R. M. NEAL, *Probabilistic Inference Using Markov Chain Monte Carlo Methods*, Department of Computer Science, University of Toronto Toronto, Ontario, Canada, 1993.
- [210] —, *Bayesian Learning for Neural Networks*, PhD thesis, University of Toronto, 1995.
- [211] R. M. NEAL ET AL., *MCMC using Hamiltonian dynamics*, Handbook of Markov Chain Monte Carlo, 2 (2011), p. 2.
- [212] C. NEMETH AND P. FEARNHEAD, *Stochastic gradient Markov chain Monte Carlo*, Journal of the American Statistical Association, 116 (2021), pp. 433–450.
- [213] D. NGUYEN-TUONG, J. PETERS, AND M. SEEGER, *Local Gaussian process regression for real time online model learning*, Advances in Neural Information Processing Systems, 21 (2008).
- [214] G. W. OEHLERT, *A note on the delta method*, The American Statistician, 46 (1992), pp. 27–29.
- [215] A. OLIVIER, M. D. SHIELDS, AND L. GRAHAM-BRADY, *Bayesian neural networks for uncertainty quantification in data-driven materials modeling*, Computer Methods in Applied Mechanics and Engineering, 386 (2021), p. 114079.
- [216] I. OSBAND, J. ASLANIDES, AND A. CASSIRER, *Randomized prior functions for deep reinforcement learning*, Advances in Neural Information Processing Systems, 31 (2018).
- [217] I. OSBAND, Z. WEN, S. M. ASGHARI, V. DWARACHERLA, X. LU, AND B. VAN ROY, *Evaluating high-order predictive distributions in deep learning*, in Uncertainty in Artificial Intelligence, PMLR, 2022, pp. 1552–1560.
- [218] Y. OVADIA, E. FERTIG, J. REN, Z. NADO, D. SCULLEY, S. NOWOZIN, J. DILLON, B. LAKSHMINARAYANAN, AND J. SNOEK, *Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift*, in Advances in Neural Information Processing Systems, 2019, pp. 13991–14002.
- [219] G. PANG, L. LU, AND G. E. KARNIADAKIS, *fPINNs: Fractional physics-informed neural networks*, SIAM Journal on Scientific Computing, 41 (2019), pp. A2603–A2626.
- [220] G. PAPAMAKARIOS, E. NALISNICK, D. J. REZENDE, S. MOHAMED, AND B. LAKSHMINARAYANAN, *Normalizing flows for probabilistic modeling and inference*, Journal of Machine Learning Research, 22 (2021), pp. 1–64.
- [221] G. PAPAMAKARIOS, T. PAVLAKOU, AND I. MURRAY, *Masked autoregressive flow for density estimation*, Advances in Neural Information Processing Systems, 30 (2017).

- [222] G. I. PARISI, R. KEMKER, J. L. PART, C. KANAN, AND S. WERMTER, *Continual lifelong learning with neural networks: A review*, Neural Networks, 113 (2019), pp. 54–71.
- [223] W. S. PARKER, *Ensemble modeling, uncertainty and robust predictions*, Wiley interdisciplinary reviews: Climate change, 4 (2013), pp. 213–223.
- [224] D. V. PATEL AND A. A. OBERAI, *GAN-based priors for quantifying uncertainty in supervised learning*, SIAM/ASA Journal on Uncertainty Quantification, 9 (2021), pp. 1314–1343.
- [225] R. PATEL, I. MANICKAM, M. LEE, AND M. GULIAN, *Error-in-variables modelling for operator learning*, in Mathematical and Scientific Machine Learning, PMLR, 2022, pp. 142–157.
- [226] T. PEARCE, A. BRINTRUP, M. ZAKI, AND A. NEELY, *High-quality prediction intervals for deep learning: A distribution-free, ensembled approach*, in International Conference on Machine Learning, 2018, pp. 4075–4084.
- [227] R. PELLEGRIN, B. BULLWINKEL, M. MATTHEAKIS, AND P. PROTOPAPAS, *Transfer learning with physics-informed neural networks for efficient simulation of branched flows*, arXiv preprint arXiv:2211.00214, (2022).
- [228] M. PENWARDEN, H. OWHADI, AND R. M. KIRBY, *Kolmogorov n -widths for multitask physics-informed machine learning (PIML) methods: Towards robust metrics*, Neural Networks, 180 (2024), p. 106703.
- [229] P. PERDIKARIS, D. VENTURI, AND G. E. KARNIADAKIS, *Multifidelity information fusion algorithms for high-dimensional systems and massive data sets*, SIAM Journal on Scientific Computing, 38 (2016), pp. B521–B538.
- [230] E. PICKERING, S. GUTH, G. E. KARNIADAKIS, AND T. P. SAPSIS, *Discovering and forecasting extreme events via active learning in neural operators*, Nature Computational Science, 2 (2022), pp. 823–833.
- [231] E. PICKERING AND T. P. SAPSIS, *Structure and distribution metric for quantifying the quality of uncertainty: Assessing Gaussian processes, deep neural nets, and deep neural operators for regression*, arXiv preprint arXiv:2203.04515, (2022).
- [232] L. POLOK, V. ILA, M. SOLONY, P. SMRZ, AND P. ZEMCIK, *Incremental block Cholesky factorization for nonlinear least squares in robotics.*, in Robotics: Science and Systems, 2013, pp. 328–336.
- [233] N. G. POLSON AND V. SOKOLOV, *Deep Learning: A Bayesian Perspective*, Bayesian Analysis, 12 (2017), pp. 1275–1304.
- [234] A. F. PSAROS, X. MENG, Z. ZOU, L. GUO, AND G. E. KARNIADAKIS, *Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons*, Journal of Computational Physics, 477 (2023), p. 111902.
- [235] T. QIN, Z. CHEN, J. D. JAKEMAN, AND D. XIU, *Deep learning of parameterized equations with applications to uncertainty quantification*, International Journal for Uncertainty Quantification, 11 (2021).

- [236] J. QUIÑONERO-CANDELA, M. SUGIYAMA, A. SCHWAIGHOFER, AND N. D. LAWRENCE, *Dataset shift in machine learning*, Mit Press, 2008.
- [237] C. RACKAUCKAS, M. INNES, Y. MA, J. BETTENCOURT, L. WHITE, AND V. DIXIT, *Diffeqflux.jl—a julia library for neural differential equations*, arXiv preprint arXiv:1902.02376, (2019).
- [238] C. RACKAUCKAS, Y. MA, J. MARTENSEN, C. WARNER, K. ZUBOV, R. SUPEKAR, D. SKINNER, A. RAMADHAN, AND A. EDELMAN, *Universal differential equations for scientific machine learning*, arXiv preprint arXiv:2001.04385, (2020).
- [239] A. RAGHU, M. RAGHU, S. BENGIO, AND O. VINYALS, *Rapid learning or feature reuse? Towards understanding the effectiveness of MAML*, in 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, OpenReview.net, 2020.
- [240] R. RAHAMAN AND A. H. THIERY, *Uncertainty Quantification and Deep Ensembles*, in Advances in Neural Information Processing Systems, 2021.
- [241] M. RAISSI AND G. E. KARNIADAKIS, *Deep multi-fidelity Gaussian processes*, arXiv preprint arXiv:1604.07484, (2016).
- [242] M. RAISSI, P. PERDIKARIS, AND G. E. KARNIADAKIS, *Machine learning of linear differential equations using Gaussian processes*, Journal of Computational Physics, 348 (2017), pp. 683–693.
- [243] —, *Numerical Gaussian processes for time-dependent and nonlinear partial differential equations*, SIAM Journal on Scientific Computing, 40 (2018), pp. A172–A198.
- [244] 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 (2019), pp. 686–707.
- [245] M. RAISSI, A. YAZDANI, AND G. E. KARNIADAKIS, *Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations*, Science, 367 (2020), pp. 1026–1030.
- [246] A. RANGANATHAN, M.-H. YANG, AND J. HO, *Online sparse Gaussian process regression and its applications*, IEEE Transactions on Image Processing, 20 (2010), pp. 391–404.
- [247] C. RAO, H. SUN, AND Y. LIU, *Physics-informed deep learning for computational elastodynamics without labeled data*, Journal of Engineering Mechanics, 147 (2021), p. 04021043.
- [248] P. REN, Y. XIAO, X. CHANG, P.-Y. HUANG, Z. LI, B. B. GUPTA, X. CHEN, AND X. WANG, *A survey of deep active learning*, ACM Computing Surveys (CSUR), 54 (2021), pp. 1–40.

- [249] L. RICHARD DE NEUFVILLE, S. SCHOLTES, AND T. WANG, *Real options by spreadsheet: Parking garage case example*, Journal of Infrastructure Systems, 12 (2006), pp. 107–111.
- [250] R. RICO-MARTINEZ, K. KRISCHER, I. KEVREKIDIS, M. KUBE, AND J. HUDSON, *Discrete-vs. continuous-time nonlinear signal processing of cu electrodisolution data*, Chemical Engineering Communications, 118 (1992), pp. 25–48.
- [251] H. RITTER, A. BOTEV, AND D. BARBER, *A scalable Laplace approximation for neural networks*, in 6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings, vol. 6, International Conference on Representation Learning, 2018.
- [252] G. O. ROBERTS AND J. S. ROSENTHAL, *Optimal scaling of discrete approximations to Langevin diffusions*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 60 (1998), pp. 255–268.
- [253] U. ROSOLIA, X. ZHANG, AND F. BORRELLI, *Data-driven predictive control for autonomous systems*, Annual Review of Control, Robotics, and Autonomous Systems, 1 (2018), pp. 259–286.
- [254] G. RUBINO, B. TUFFIN, ET AL., *Rare event simulation using Monte Carlo methods*, vol. 73, Wiley Online Library, 2009.
- [255] S. RUDER, *An overview of gradient descent optimization algorithms*, arXiv preprint arXiv:1609.04747, (2016).
- [256] S. H. RUDY, S. L. BRUNTON, J. L. PROCTOR, AND J. N. KUTZ, *Data-driven discovery of partial differential equations*, Science Advances, 3 (2017), p. e1602614.
- [257] D. RULLIÈRE, N. DURRANDE, F. BACHOC, AND C. CHEVALIER, *Nested kriging predictions for datasets with a large number of observations*, Statistics and Computing, 28 (2018), pp. 849–867.
- [258] F. SAHLI COSTABAL, Y. YANG, P. PERDIKARIS, D. E. HURTADO, AND E. KUHL, *Physics-informed neural networks for cardiac activation mapping*, Frontiers in Physics, 8 (2020), p. 42.
- [259] M. SARABIAN, H. BABAEE, AND K. LAKSARI, *Physics-informed neural networks for brain hemodynamic predictions using medical imaging*, IEEE Transactions on Medical Imaging, 41 (2022), pp. 2285–2303.
- [260] F. SCHAÄFER, M. KATZFUSS, AND H. OWHADI, *Sparse Cholesky factorization by Kullback–Leibler minimization*, SIAM Journal on Scientific Computing, 43 (2021), pp. A2019–A2046.
- [261] S. M. SCHENNACH, *Recent advances in the measurement error literature*, Annual Review of Economics, 8 (2016), pp. 341–377.
- [262] M. SCHMIDT AND H. LIPSON, *Distilling free-form natural laws from experimental data*, Science, 324 (2009), pp. 81–85.

- [263] D. SEN AND D. SEN, *Use of a modified SIRD model to analyze COVID-19 data*, Industrial & Engineering Chemistry Research, 60 (2021), pp. 4251–4260.
- [264] M. SENSOY, L. KAPLAN, AND M. KANDEMIR, *Evidential deep learning to quantify classification uncertainty*, in Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc., 2018.
- [265] B. SETTLES, *Active learning literature survey*, tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 2009.
- [266] Y. SHIN, J. DARBON, AND G. E. KARNIADAKIS, *On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs*, Communications in Computational Physics, 28 (2020), pp. 2042–2074.
- [267] K. SHUKLA, J. D. TOSCANO, Z. WANG, Z. ZOU, AND G. E. KARNIADAKIS, *A comprehensive and FAIR comparison between MLP and KAN representations for differential equations and operator networks*, Computer Methods in Applied Mechanics and Engineering, 431 (2024), p. 117290.
- [268] K. SHUKLA, Z. ZOU, C. H. CHAN, A. PANDEY, Z. WANG, AND G. E. KARNIADAKIS, *NeuroSEM: A hybrid framework for simulating multiphysics problems by coupling PINNs and spectral elements*, Computer Methods in Applied Mechanics and Engineering, 433 (2025), p. 117498.
- [269] J. SIRIGNANO AND K. SPILIOPOULOS, *DGM: A deep learning algorithm for solving partial differential equations*, Journal of Computational Physics, 375 (2018), pp. 1339–1364.
- [270] J. SOIBAM, I. ASLANIDOU, K. KYPRIANIDIS, AND R. B. FDHILA, *Inverse flow prediction using ensemble PINNs and uncertainty quantification*, International Journal of Heat and Mass Transfer, 226 (2024), p. 125480.
- [271] H. SON AND M. LEE, *A PINN approach for identifying governing parameters of noisy thermoacoustic systems*, Journal of Fluid Mechanics, 984 (2024), p. A21.
- [272] N. STÅHL, G. FALKMAN, A. KARLSSON, AND G. MATHIASON, *Evaluation of uncertainty quantification in deep learning*, in International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer, 2020, pp. 556–568.
- [273] J. STRAHAN, J. FINKEL, A. R. DINNER, AND J. WEARE, *Predicting rare events using neural networks and short-trajectory data*, Journal of Computational Physics, 488 (2023), p. 112152.
- [274] T. D. STROUT, M. HILLEN, C. GUTHEIL, E. ANDERSON, R. HUTCHINSON, H. WARD, H. KAY, G. J. MILLS, AND P. K. HAN, *Tolerance of uncertainty: A systematic review of health and healthcare-related outcomes*, Patient Education and Counseling, 101 (2018), pp. 1518–1537.

- [275] L. SUN, H. GAO, S. PAN, AND J.-X. WANG, *Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data*, Computer Methods in Applied Mechanics and Engineering, 361 (2020), p. 112732.
- [276] K. TANG, X. WAN, AND C. YANG, *DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations*, Journal of Computational Physics, 476 (2023), p. 111868.
- [277] K. TANG, J. ZHAI, X. WAN, AND C. YANG, *Adversarial adaptive sampling: Unify PINN and optimal transport for the approximation of PDEs*, arXiv preprint arXiv:2305.18702, (2023).
- [278] S. TAVERNIERS, E. J. HALL, M. A. KATSOULAKIS, AND D. M. TARTAKOVSKY, *Mutual information for explainable deep learning of multiscale systems*, Journal of Computational Physics, 444 (2021), p. 110551.
- [279] L. TIERNEY AND J. B. KADANE, *Accurate approximations for posterior moments and marginal densities*, Journal of the American Statistical Association, 81 (1986), pp. 82–86.
- [280] R. TIPIREDDY, P. STINIS, AND A. M. TARTAKOVSKY, *Time-dependent stochastic basis adaptation for uncertainty quantification*, arXiv preprint arXiv:2103.03316, (2021).
- [281] J. D. TOSCANO, V. OOMMEN, A. J. VARGHESE, Z. ZOU, N. A. DARYAKENARI, C. WU, AND G. E. KARNIADAKIS, *From PINNs to PIKANs: Recent advances in physics-informed machine learning*, arXiv preprint arXiv:2410.13228, (2024).
- [282] D. TRAN, A. KUCUKELBIR, A. B. DIENG, M. RUDOLPH, D. LIANG, AND D. M. BLEI, *Edward: A library for probabilistic modeling, inference, and criticism*, arXiv preprint arXiv:1610.09787, (2016).
- [283] N. TRASK, A. HUANG, AND X. HU, *Enforcing exact physics in scientific machine learning: A data-driven exterior calculus on graphs*, Journal of Computational Physics, 456 (2022), p. 110969.
- [284] V. TRESP, S. AHMAD, AND R. NEUNEIER, *Training neural networks with deficient data*, Advances in Neural Information Processing Systems, 6 (1993).
- [285] R. K. TRIPATHY AND I. BILIONIS, *Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification*, Journal of Computational Physics, 375 (2018), pp. 565–588.
- [286] P. TSILIFIS, S. GHOSH, AND V. ANDREOLI, *Inverse design under uncertainty using conditional normalizing flows*, in AIAA Scitech 2022 Forum, 2022, p. 0631.
- [287] K. UM, E. J. HALL, M. A. KATSOULAKIS, AND D. M. TARTAKOVSKY, *Causality and Bayesian Network PDEs for multiscale representations of porous media*, Journal of Computational Physics, 394 (2019), pp. 658–678.

- [288] G. M. VAN DE VEN AND A. S. TOLIAS, *Three scenarios for continual learning*, arXiv preprint arXiv:1904.07734, (2019).
- [289] J. VAN GORP, J. SCHOUKENS, AND R. PINTELON, *Learning neural networks with noisy inputs using the errors-in-variables approach*, IEEE Transactions on Neural Networks, 11 (2000), pp. 402–414.
- [290] J. WAHLSTRÖM AND L. ERIKSSON, *Modelling diesel engines with a variable-geometry turbocharger and exhaust gas recirculation by optimization of model parameters for capturing non-linear system dynamics*, Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, 225 (2011), pp. 960–986.
- [291] X. WAN AND G. E. KARNIADAKIS, *Multi-element generalized polynomial chaos for arbitrary probability measures*, SIAM Journal on Scientific Computing, 28 (2006), pp. 901–928.
- [292] X. WAN AND S. WEI, *Coupling the reduced-order model and the generative model for an importance sampling estimator*, Journal of Computational Physics, 408 (2020), p. 109281.
- [293] H. WANG AND D.-Y. YEUNG, *A survey on Bayesian deep learning*, ACM Computing Surveys (CSUR), 53 (2020), pp. 1–37.
- [294] H. WANG, H. ZHAO, AND B. LI, *Bridging multi-task learning and meta-learning: Towards efficient training and effective adaptation*, in International Conference on Machine Learning, PMLR, 2021, pp. 10991–11002.
- [295] L. WANG, J. MI, X. MENG, AND Z. GUO, *A localized mass-conserving lattice Boltzmann approach for non-Newtonian fluid flows*, Communications in Computational Physics, 17 (2015), pp. 908–924.
- [296] L. WANG, X. ZHANG, H. SU, AND J. ZHU, *A comprehensive survey of continual learning: Theory, method and application*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2024).
- [297] N. WANG, H. CHANG, AND D. ZHANG, *Efficient uncertainty quantification for dynamic subsurface flow with surrogate by Theory-guided Neural Network*, Computer Methods in Applied Mechanics and Engineering, 373 (2021), p. 113492.
- [298] S. WANG, H. WANG, AND P. PERDIKARIS, *Learning the solution operator of parametric partial differential equations with physics-informed DeepONets*, Science Advances, 7 (2021), p. eabi8605.
- [299] —, *Improved architectures and training algorithms for deep operator networks*, Journal of Scientific Computing, 92 (2022), p. 35.
- [300] Y. WANG, F. LIU, AND D. E. SCHIAVAZZI, *Variational inference with NoFAS: Normalizing flow with adaptive surrogate for computationally expensive models*, Journal of Computational Physics, 467 (2022), p. 111454.

- [301] M. WELLING AND Y. W. TEH, *Bayesian learning via stochastic gradient Langevin dynamics*, in Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 681–688.
- [302] Z. WEN, I. OSBAND, C. QIN, X. LU, M. IBRAHIMI, V. DWARACHERLA, M. ASGHARI, AND B. VAN ROY, *From predictions to decisions: The importance of joint predictive distributions*, arXiv preprint arXiv:2107.09224, (2021).
- [303] V. D. WILD, S. GHALEBIKESABI, D. SEJDINOVIC, AND J. KNOBLAUCH, *A rigorous link between deep ensembles and (variational) Bayesian methods*, Advances in Neural Information Processing Systems, 36 (2024).
- [304] O. WILES, S. GOWAL, F. STIMBERG, S. ALVISE-REBUFFI, I. K TENA, K. DVIJOTHAM, AND T. CEMGIL, *A fine-grained analysis on distribution shift*, arXiv preprint arXiv:2110.11328, (2021).
- [305] J. WILLARD, X. JIA, S. XU, M. STEINBACH, AND V. KUMAR, *Integrating scientific knowledge with machine learning for engineering and environmental systems*, ACM Computing Surveys, 55 (2022), pp. 1–37.
- [306] C. K. WILLIAMS AND C. E. RASMUSSEN, *Gaussian Processes for Machine Learning*, vol. 2, MIT press Cambridge, MA, 2006.
- [307] A. G. WILSON, *The case for Bayesian deep learning*, arXiv preprint arXiv:2001.10995, (2020).
- [308] A. G. WILSON AND P. IZMAILOV, *Bayesian deep learning and a probabilistic perspective of generalization*, Advances in Neural Information Processing Systems, 33 (2020), pp. 4697–4708.
- [309] N. WINOVICH, K. RAMANI, AND G. LIN, *ConvPDE-UQ: Convolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains*, Journal of Computational Physics, 394 (2019), pp. 263–279.
- [310] W. WRIGHT, *Bayesian approach to neural-network modeling with input uncertainty*, IEEE Transactions on Neural Networks, 10 (1999), pp. 1261–1270.
- [311] T. XIFARA, C. SHERLOCK, S. LIVINGSTONE, S. BYRNE, AND M. GIROLAMI, *Langevin diffusions and the Metropolis-adjusted Langevin algorithm*, Statistics & Probability Letters, 91 (2014), pp. 14–19.
- [312] D. XIU, *Numerical Methods for Stochastic Computations*, Princeton University Press, 2010.
- [313] K. XU AND E. DARVE, *ADCME: Learning spatially-varying physical fields using deep neural networks*, arXiv preprint arXiv:2011.11955, (2020).
- [314] —, *Solving inverse problems in stochastic models using deep neural networks and adversarial training*, Computer Methods in Applied Mechanics and Engineering, 384 (2021), p. 113976.

- [315] L. YANG, X. MENG, AND G. E. KARNIADAKIS, *B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data*, Journal of Computational Physics, 425 (2021), p. 109913.
- [316] L. YANG, S. TREICHLER, T. KURTH, K. FISCHER, D. BARAJAS-SOLANO, J. ROMERO, V. CHURAVY, A. TARTAKOVSKY, M. HOUSTON, M. PRABHAT, ET AL., *Highly-scalable, physics-informed GANs for learning solutions of stochastic PDEs*, in 2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS), IEEE, 2019, pp. 1–11.
- [317] L. YANG, D. ZHANG, AND G. E. KARNIADAKIS, *Physics-informed generative adversarial networks for stochastic differential equations*, SIAM Journal on Scientific Computing, 42 (2020), pp. A292–A317.
- [318] M. YANG AND J. T. FOSTER, *Multi-output physics-informed neural networks for forward and inverse PDE problems with uncertainties*, Computer Methods in Applied Mechanics and Engineering, 402 (2022), p. 115041.
- [319] Y. YANG, M. AZIZ BHOURI, AND P. PERDIKARIS, *Bayesian differential programming for robust systems identification under uncertainty*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 476 (2020), p. 20200290.
- [320] Y. YANG, G. KISSAS, AND P. PERDIKARIS, *Scalable uncertainty quantification for deep operator networks using randomized priors*, Computer Methods in Applied Mechanics and Engineering, 399 (2022), p. 115399.
- [321] Y. YANG AND P. PERDIKARIS, *Adversarial uncertainty quantification in physics-informed neural networks*, Journal of Computational Physics, 394 (2019), pp. 136–152.
- [322] —, *Conditional deep surrogate models for stochastic, high-dimensional, and multi-fidelity systems*, Computational Mechanics, 64 (2019), pp. 417–434.
- [323] J. YAO, W. PAN, S. GHOSH, AND F. DOSHI-VELEZ, *Quality of uncertainty quantification for Bayesian neural network inference*, arXiv preprint arXiv:1906.09686, (2019).
- [324] K. YEO, D. E. GRULLON, F.-K. SUN, D. S. BONING, AND J. R. KALAGNANAM, *Variational inference formulation for a model-free simulation of a dynamical system with unknown parameters by a recurrent neural network*, SIAM Journal on Scientific Computing, 43 (2021), pp. A1305–A1335.
- [325] M. YIN, Z. ZOU, E. ZHANG, C. CAVINATO, J. D. HUMPHREY, AND G. E. KARNIADAKIS, *A generative modeling framework for inferring families of biomechanical constitutive laws in data-sparse regimes*, Journal of the Mechanics and Physics of Solids, 181 (2023), p. 105424.
- [326] J. YONG AND X. Y. ZHOU, *Stochastic controls: Hamiltonian systems and HJB equations*, vol. 43, Springer Science & Business Media, 2012.

- [327] A. ZANONI, G. GERACI, M. SALVADOR, K. MENON, A. L. MARSDEN, AND D. E. SCHIAVAZZI, *Improved multifidelity Monte Carlo estimators based on normalizing flows and dimensionality reduction techniques*, Computer Methods in Applied Mechanics and Engineering, 429 (2024), p. 117119.
- [328] E. ZELIKMAN, C. HEALY, S. ZHOU, AND A. AVATI, *CRUDE: calibrating regression uncertainty distributions empirically*, arXiv preprint arXiv:2005.12496, (2020).
- [329] D. ZHANG, L. LU, L. GUO, AND G. E. KARNIADAKIS, *Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems*, Journal of Computational Physics, 397 (2019), p. 108850.
- [330] J. ZHANG AND W. MA, *Data-driven discovery of governing equations for fluid dynamics based on molecular simulation*, Journal of Fluid Mechanics, 892 (2020), p. A5.
- [331] L. ZHANG, S. WANG, F. YUAN, B. GENG, AND M. YANG, *Lifelong language learning with adaptive uncertainty regularization*, Information Sciences, 622 (2023), pp. 794–807.
- [332] S. ZHANG AND G. LIN, *Robust data-driven discovery of governing physical laws with error bars*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 474 (2018), p. 20180305.
- [333] Z. ZHANG, Y. SHIN, AND G. E. KARNIADAKIS, *GFINNs: GENERIC formalism informed neural networks for deterministic and stochastic dynamical systems*, Philosophical Transactions of the Royal Society A, 380 (2022), p. 20210207.
- [334] Z. ZHANG, Z. ZOU, E. KUHL, AND G. E. KARNIADAKIS, *Discovering a reaction–diffusion model for Alzheimer’s disease by combining PINNs with symbolic regression*, Computer Methods in Applied Mechanics and Engineering, 419 (2024), p. 116647.
- [335] W. ZHONG AND H. MEIDANI, *PI-VAE: Physics-informed variational auto-encoder for stochastic differential equations*, Computer Methods in Applied Mechanics and Engineering, 403 (2023), p. 115664.
- [336] X. ZHOU, H. LIU, F. POURPANAH, T. ZENG, AND X. WANG, *A survey on epistemic (model) uncertainty in supervised learning: Recent advances and applications*, Neurocomputing, 489 (2022), pp. 449–465.
- [337] Z. ZHOU, N. ZABARAS, AND D. M. TARTAKOVSKY, *Deep learning for simultaneous inference of hydraulic and transport properties*, Water Resources Research, 58 (2022), p. e2021WR031438.
- [338] A. ZHU, T. BERTALAN, B. ZHU, Y. TANG, AND I. G. KEVREKIDIS, *Implementation and (Inverse Modified) Error Analysis for Implicitly Templated ODE-Nets*, SIAM Journal on Applied Dynamical Systems, 23 (2024), pp. 2643–2669.

- [339] Q. ZHU, W. LI, H. KIM, Y. XIANG, K. WARDEGA, Z. WANG, Y. WANG, H. LIANG, C. HUANG, J. FAN, ET AL., *Know the unknowns: Addressing disturbances and uncertainties in autonomous systems*, in Proceedings of the 39th International Conference on Computer-Aided Design, 2020, pp. 1–9.
- [340] Y. ZHU, N. ZABARAS, P.-S. KOUTSOURELAKIS, AND P. PERDIKARIS, *Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data*, Journal of Computational Physics, 394 (2019), pp. 56–81.
- [341] Y. ZONG, D. BARAJAS-SOLANO, AND A. M. TARTAKOVSKY, *Randomized physics-informed machine learning for uncertainty quantification in high-dimensional inverse problems*, Journal of Computational Physics, 519 (2024), p. 113395.
- [342] ———, *Randomized physics-informed neural networks for Bayesian data assimilation*, arXiv preprint arXiv:2407.04617, (2024).
- [343] Z. ZOU, A. KAHANA, E. ZHANG, E. TURKEL, R. RANADE, J. PATHAK, AND G. E. KARNIADAKIS, *Large scale scattering using fast solvers based on neural operators*, arXiv preprint arXiv:2405.12380, (2024).
- [344] Z. ZOU AND G. E. KARNIADAKIS, *L-HYDRA: Multi-head physics-informed neural networks*, arXiv preprint arXiv:2301.02152, (2023).
- [345] Z. ZOU, T. MENG, P. CHEN, J. DARBON, AND G. E. KARNIADAKIS, *Leveraging viscous Hamilton–Jacobi PDEs for uncertainty quantification in scientific machine learning*, SIAM/ASA Journal on Uncertainty Quantification, 12 (2024), pp. 1165–1191.
- [346] Z. ZOU, X. MENG, AND G. E. KARNIADAKIS, *Correcting model misspecification in physics-informed neural networks (PINNs)*, Journal of Computational Physics, 505 (2024), p. 112918.
- [347] ———, *Uncertainty quantification for noisy inputs–outputs in physics-informed neural networks and neural operators*, Computer Methods in Applied Mechanics and Engineering, 433 (2025), p. 117479.
- [348] Z. ZOU, X. MENG, A. F. PSAROS, AND G. E. KARNIADAKIS, *NeuralUQ: A comprehensive library for uncertainty quantification in neural differential equations and operators*, SIAM Review, 66 (2024), pp. 161–190.