

NUMERICAL METHODS FOR MATHEMATICAL MODELING WITH
APPLICATIONS TO FLUID AND MOLECULAR DYNAMICS

by

Sarah Helfert

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Applied Mathematics

Charlotte

2026

Approved by:

Dr. Xingjie Helen Li

Dr. Taufiquar Khan

Dr. Duan Chen

Dr. Anthony Brizendine

ABSTRACT

SARAH HELFERT. NUMERICAL METHODS FOR MATHEMATICAL
MODELING WITH APPLICATIONS TO FLUID AND MOLECULAR
DYNAMICS. (Under the direction of DR. XINGJIE HELEN LI)

This dissertation aims to advance the mathematical foundations for scientific machine learning approaches to problems in fluid dynamics, molecular simulation, and multi-scale systems. In this work, I bridge rigorous numerical analysis with modern machine learning architectures, guided by a breadth of projects ranging from classical analysis to novel model development. These projects aim to address the question regarding the efficacy of machine learning-based models over classical mathematical methods by enhancing the understanding and interconnections between these two approaches for fluid and molecular dynamics modeling, facilitating the development of new methods that integrate the advantages of both.

Together with my advisor Dr. Xingjie Helen Li, we develop numerical analysis for a range of classical schemes and novel coarse-graining approaches for solving stochastic differential equations. While novel machine learning approaches perform exceptionally well on benchmark examples, a rigorous theoretical framework is still needed to understand precisely why such models achieve their observed robustness, accuracy and efficiency. We provide a baseline study that will be applied to the novel models in future work. In this work, we presented a new benchmark equation to evaluate SDE solvers that better captures the overall performance of classical numerical schemes than popular choices of benchmark equations. This was completed by comparing the time evolution and asymptotic limits of statistical moments of exact solutions to the solutions given by the solvers. This analysis provides insight into whether various solvers are stable or biased over long timescales.

With the same goal of understanding numerical performance and modeling error in the classical setting, we also developed error analysis for coarse-grained models

of SDEs, which are cheap-to-simulate surrogate models that reduce the number of variables of interest while still capturing important system behavior. I inspect a novel coarse-graining method that achieves a 10-fold speedup with the same accuracy for the 1D benchmark examples. My error analysis includes calculating system statistics and analyzing the performance of the model in the long and short terms.

Throughout internships at Pacific Northwest National Laboratory, I also explore common failure modes that arise when applying machine learning models to problems in fluid dynamics. And so, we present a range of approaches for improving the performance of physics-informed neural networks (PINNs), physics-informed deep operator networks (PI-DeepONets), and physics-informed Kolmogorov Arnold Networks. These projects include a novel stacked multifidelity architecture for improving PINNs on chaotic dynamical systems, domain decomposition KANs for problems with complex geometries, and theory-based adaptive loss weighting to improve the accuracy and efficiency of physics-informed deep operator networks.

ACKNOWLEDGEMENTS

I extend my gratitude to my advisor, Dr. Xingjie Helen Li, for her support and guidance throughout my graduate education. I also thank Dr. Taufiqar Khan, Dr. Duan Chen, and Dr. Anthony Brizendine for their guidance and willingness to serve on my PhD committee. My gratitude also goes to Dr. Amanda Howard and Dr. Panos Stinis at Pacific Northwest National Lab for their mentorship through various internships.

I acknowledge the support of GASP funding, the Edward C. Giles Dissertation Year Fellowship, and Dr. Li's NSF grants: NSF DMS-2513795 and NSF CAREER award DMS-1847770. I acknowledge the support of the NSF Mathematical Sciences Graduate Internship for funding one of the projects within this dissertation.

TABLE OF CONTENTS

LIST OF TABLES	xi
LIST OF FIGURES	xiii
LIST OF ABBREVIATIONS	xxii
CHAPTER 1: INTRODUCTION	1
CHAPTER 2: NUMERICAL STABILITY REVISITED FOR CLASSICAL SDE INTEGRATORS	4
2.1. Stochastic Differential Equations	8
2.2. Overview of numerical schemes	9
2.3. Asymptotic moment stability for Geometric Brown Motion	11
2.3.1. Evolution of moments for geometric Brownian motion	11
2.3.2. Asymptotic moment stability regions for various schemes	12
2.4. Derivation and properties of benchmark problem	17
2.4.1. Reduction of cases	18
2.4.2. Equilibrium distribution	20
2.4.3. Evolution of moments	23
2.5. Statistical stability properties of numerical schemes	25
2.5.1. Asymptotic moment stability	26
2.5.2. Euler-Maruyama (EM)	28
2.5.3. Milstein (Mil)	29
2.5.4. Stochastic Heun (SH)	31
2.5.5. 3-stage Runge-Kutta	32

2.5.6.	Summary of results	34
2.6.	Numerical verification	37
2.6.1.	Asymptotic accuracy of moments for methods considered	37
2.6.2.	Pre-asymptotic moment evolution	39
2.6.3.	Numerical comparison in a nonlinear case	41
2.7.	Conclusions	46
CHAPTER 3:	MODELING ERROR ANALYSIS OF COARSE-GRAINED STOCHASTIC SYSTEMS	47
3.1.	Coarse-graining of linear Langevin SDEs	50
3.1.1.	Approach 0	52
3.1.2.	Approach 1	52
3.1.3.	Approach 2	53
3.2.	Statistical Properties	53
3.2.1.	Error Metrics	56
3.3.	Cases of Analysis	59
3.4.	Numerical Experiments	60
3.4.1.	2D example	60
3.4.2.	40-dimensional spring-mass system	61
3.5.	Conclusions	65

CHAPTER 4: LEADING ORDER ESTIMATES FOR RELAXATION TIME ERROR	67
4.1. Relaxation time of a linear Langevin SDE and coarse graining approaches	67
4.1.1. Non-perturbative bounds on the ordering of relaxation time	69
4.2. Relaxation time error bound: Approach 1	69
4.3. Relaxation time error bound: Approach 2	78
4.4. Conclusions	80
CHAPTER 5: PRELIMINARIES ON SCIENTIFIC MACHINE LEARNING METHODS	82
5.1. Physics-informed neural networks	82
5.2. Physics-informed deep operator networks	84
5.3. Kolmogorov-Arnold Networks	86
5.4. Conclusions	88
CHAPTER 6: STACKED MULTIFIDELITY PHYSICS-INFORMED NEURAL NETWORKS	89
6.1. When PINNs fail to train	90
6.2. Methods	91
6.2.1. Multifidelity PINNs	91
6.2.2. Stacking PINNs	93
6.3. Results with PINNs	95
6.3.1. Pendulum	95
6.3.2. Multiscale problems	96
6.3.3. Wave equation	98

6.3.4. 2D wave equation	101
6.4. Conclusions	102
CHAPTER 7: DOMAIN DECOMPOSITION FOR SCIENTIFIC MACHINE LEARNING METHODS	108
7.1. Finite basis Kolmogorov-Arnold Networks: domain decomposition for data-driven and physics-informed problems	112
7.1.1. Methods	113
7.1.2. Data-driven results	115
7.1.3. Physics-informed results	118
7.2. Adaptive Domain Decomposition for finite basis Physics-informed Neural Networks	123
7.2.1. Methodology	126
7.2.2. Results	138
7.2.3. Conclusions	152
7.3. Model Hyperparameters	155
CHAPTER 8: EFFICIENT AND ACCURATE ADAPTIVE LOSS WEIGHTING FOR PHYSICS-INFORMED DEEP OPERATOR NETWORKS	157
8.1. The Neural Tangent Kernel and Conjugate Kernel	159
8.2. Results	161
8.2.1. Wave equation	161
8.2.2. Burgers equation	162
8.2.3. Stokes flow	165
8.3. Conclusions	172
CHAPTER 9: DISCUSSION AND FUTURE WORK	177

REFERENCES

LIST OF TABLES

TABLE 2.1: A summary of the strong and weak convergence orders of the methods studied. The first two rows correspond to results proved in [1], while the latter two rows correspond to results proved in [2, 3].	11
TABLE 2.2: A summary of the results of in Sections 2.5.2–2.5.5, i.e. the asymptotic accuracy results for the first and second moments as predicted by the methods studied.	34
TABLE 6.1: Relative ℓ_2 errors and network sizes for the multiscale problem in Sec. 6.3.2. All stacking PINNs begin with a single fidelity network with size 3×32 . For the stacking cases, we report the first stacking level that has a relative ℓ_2 error less than the minimum single fidelity error, and the relative ℓ_2 error after the tenth stacking level.	98
TABLE 6.2: Schedule for increasing c with stacking steps for each of the cases considered for the wave equation case. The right column has the relative ℓ_2 error for the final stacking step. Note that the relative error is calculated with respect to the exact solution with $c = 2$ for cases 1, 2, and 3, and $c = 4$ for Case 4.	100
TABLE 7.1: Relative ℓ_2 errors for data-driven test 2.	118
TABLE 7.2: Relative ℓ_2 errors for physics-informed test 2.	121
TABLE 7.3: Relative L2 errors and number of collocation points active in exactly 2 subdomains for the 1D Sine equation with $\omega_1 = 1$ and $\omega_2 = 45$ across a range of overlap widths $\delta = 0.1, 0.2, 0.3$ and 30 uniform subdomains. The second row denotes a case where more collocation points were placed in the overlapping regions with an enrichment factor of 4.0, showing that more collocation points does not substantially boost accuracy.	143
TABLE 7.4: Relative L2 errors and number of collocation points active in exactly 2 subdomains for the 2D Helmholtz equation with $a_1 = 1$ and $a_2 = 4$ across a range of overlap widths $\delta = 0.09, 0.14, 0.19, 0.24$ and 25 uniform subdomains. The second row denotes a case where more collocation points were placed in the overlapping regions with an enrichment factor of 3.0, showing that more collocation points does not substantially boost accuracy.	145

TABLE 7.5: Relative L2 errors and number of collocation points active in exactly 2 subdomains for the 2D Poisson equation with multiscale features with $\omega_i = [1, 2, 4]$ across a range of overlap widths $\delta = 0.09, 0.14, 0.19, 0.24$ and 25 uniform subdomains. An oversampling factor of $\alpha = 4.0$ is used.	145
TABLE 7.6: Relative L2 errors for a standard PINN, FBPINN, and ADD-FBPINN on the 2D Helmholtz equation. Results are shown for a range of wave numbers to highlight the ability of this method to capture both high-frequency isotropic cases and high-frequency anisotropic cases.	148
TABLE 7.7: Relative L2 errors for a standard PINN, FBPINN, and ADD-FBPINN on the 2D Poisson equation with multiscale features. Results are shown for a range of frequencies.	150
TABLE 7.8: Relative L2 errors for a standard PINN, FBPINN, and ADD-FBPINN on the 2D Poisson equation with multiscale features. Results are shown for a range of frequencies with training difficulty increasing for each case.	150
TABLE 7.9: Relative L2 errors for a standard PINN, FBPINN, and ADD-FBPINN on Burgers equation. Results are shown for two test cases, one where the shock is located at $x = 0$ (Equation 7.32), and one where the shock is located at $x = -0.25$ (Equation 7.33).	152
TABLE 7.10: Ablation Study Parameters	155
TABLE 7.11: PINN Parameters	156
TABLE 7.12: Fixed DD FBPINN Parameters	156
TABLE 7.13: ADD-FBPINN Parameters	156
TABLE 7.14: ADD-FBPINN POUnet Parameters	156
TABLE 8.1: Results for Burgers equation for a range of viscosities ν . The mean and standard deviation of the relative ℓ_2 error are over 500 initial conditions in the test set. Average run times are computed using an NVIDIA A100 GPU.	169

LIST OF FIGURES

- FIGURE 2.1: **Top row:** 1st moment stability regions (boundaries shown as lines) and 2nd moment stability regions (pink shaded regions) for the schemes considered; in each case, the stability region for second moments is strictly contained inside the stability region for first moments. **Bottom row:** A comparison of 2nd moment stability regions among all schemes plotted on the same axes. For asymptotic first moments, we observe EM and Mil are more stable than the higher-order schemes in many cases. For asymptotic second moments, RK3 is the most stable scheme when $\eta \leq 0.5$ or $\eta > 1.145$; and EM is the most stable when $0.5 < \eta < 1.145$. 35
- FIGURE 2.2: Asymptotic accuracy of the first and second moments obtained using the various numerical methods with respect to the step size h . Errors were computed as the absolute value of the difference between the exact moment in the limit $T \rightarrow \infty$ and the numerical moment at final simulation time $T = 1000$. **Column 1:** The absolute error in the first moment for the SH and RK3 schemes only, showing the $O(h)$ convergence predicted; EM and Mil are not shown as they exactly recover the first moment. **Column 2:** The relative error of the asymptotic second moment for each scheme. For the case where $\eta = 0.1$ (first row), the higher-order schemes perform better, and SH reaches machine precision for small h , with the accumulation of floating point errors appearing to limit further gains. For the case of larger η (second row), all schemes give similar error, with EM outperforming the higher-order schemes as h gets closer to 0. 38
- FIGURE 2.3: A comparison of the evolution of absolute first moment error for the various discretization schemes up to time $T = 20$ for our benchmark equation. The first row depicts the case when $\eta = 0.1$ for step sizes $h = 0.001, 0.005, 0.025$, and the second row depicts when $\eta = 1.41$ with the same range of step sizes. In agreement with our analytical results, we see that the EM and Mil recover asymptotically exact first moments, whereas RK3 and SH are asymptotically biased. 39
- FIGURE 2.4: A comparison of the evolution of the relative error in second moments by the discretization schemes versus the exact evolution up to time $T = 20$. The first row depicts the case when $\eta = 0.1$ for step sizes $h = 0.001, 0.005, 0.025$, while the second row depicts the case when $\eta = 1.41$ with the same range of step sizes. At late times, we see substantial accuracy gains from higher-order schemes only when η is small. 40

FIGURE 2.5: Examples of the particular drift and diffusion coefficients used in the nonlinear example (2.20) (left column), along with the equilibrium measures computed by numerical integration (right column). The top row shows the case where $\eta \approx 0.05633$; while the bottom row shows the case where $\eta \approx 1.1267$. We see from the equilibrium measures that larger values of η leads to significant skew in the distribution.

42

FIGURE 2.6: Numerical errors in the asymptotic means for discretizations of the SDE with non-affine coefficients (2.20), plotted against time step size. The difference between the sample mean at time $T = 16$ taken over 4×10^4 trajectories against the equilibrium mean determined by numerical integration is plotted. We see that the accuracy decreases linearly with step size, as predicted by our analytic results for the benchmark equation.

44

FIGURE 2.7: A comparison of the evolution of the first moment for discretizations of the nonlinear problem (2.20) for the sets of parameters producing small and large values of η . The plots compare the means predicts with a fine step size $h = 0.01$ and a coarse step size $h = 1$, averaged over 4×10^4 sample trajectories. We see that the low-order schemes appear to predict the first moment evolution more consistently across a wide variation of time steps, and there is evidence that SH is unstable for the large value of η and large time-step, consistent with our stability analysis on the simpler benchmark.

45

FIGURE 3.1: Error functions for a 2D system with initial mean in equilibrium ($\mu_0 = (0, 0)$) plotted against $\lambda \in [1.1, 1000]$. Top Row: Approach 1 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance. Bottom row: Approach 2 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance.

62

FIGURE 3.2: Error functions for a 2D system with initial mean outside of equilibrium ($\mu_0 = (1, 0)$) plotted against $\lambda \in [1.1, 1000]$. Top Row: Approach 1 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance. Bottom row: Approach 2 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance.

62

FIGURE 3.3: Error functions plotted against time lags in the range of $t \in [0, 10]$ for a case where the initial mean is outside of equilibrium $\mu_0 = (1, 0)$. Here, $\lambda = 1.1$. Top Row: Approach 1 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance. Bottom row: Approach 2 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance.	63
FIGURE 3.4: Error functions plotted against time lags in the range of $t \in [0, 10]$ for a case where the initial mean is outside of equilibrium $\mu_0 = (1, 0)$. Here, $\lambda = 1000$. Top Row: Approach 1 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance. Bottom row: Approach 2 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance.	63
FIGURE 3.5: Top row: Eigenvalue gap (left) and theta values (right) plotted against values of k_2 . Bottom row: Errors plotted against k_2 , for initial mean out of equilibrium $\mu_0 = \mathbf{0}$	66
FIGURE 5.1: Physics-informed neural network	85
FIGURE 6.1: PINN results s_1 (left) and s_2 (right) as a function of time for the pendulum problem for ten random initial seedings. In each case, the PINN solution decays and does not agree well with the exact solution.	91
FIGURE 6.2: Multifidelity physics-informed neural network	93
FIGURE 6.3: Stacking multifidelity physics-informed neural network	94
FIGURE 6.4: Stacking PINN results for an illustrative example of s_1 (top) and s_2 (bottom) as a function of time for the pendulum problem up to nine stacking steps.	97
FIGURE 6.5: Pendulum relative ℓ_2 training errors after each stacking step for ten initial seedings.	97
FIGURE 6.6: Stacking PINN training for the pendulum problem.	97
FIGURE 6.7: Single fidelity	103
FIGURE 6.8: Stacking PINN	103
FIGURE 6.9: Relative ℓ_2 errors	103

- FIGURE 6.10: Results for the multiscale problem. (a) Single fidelity results for a variety of network sizes. (b) Multifidelity stacking results. (c) Relative ℓ_2 errors for the stacking PINN. The black dashed line in (c) is the lowest error from the single fidelity training for the 5×64 network. For the stacking PINN, step 0 is the single fidelity step and step 1 is the first multifidelity step. 103
- FIGURE 6.11: Results for the stacking PINN for the wave equation for Case 1. The top left figure is the exact solution. The results from training steps 0, 2, and 14 are in the middle column. Their respective absolute errors are in the right column. 104
- FIGURE 6.12: Results for the stacking PINN for the wave equation for Case 3. The left column has the exact solution for the value of c used for a given stacking step. The results from training steps 0, 2, and 10 are in the middle column. Their respective absolute errors are in the right column. 105
- FIGURE 6.13: Relative ℓ_2 errors for the stacking PINNs for the wave equations for Case 1 (circles), Case 2 (triangles), and Case 3 (squares). The relative ℓ_2 is calculated using the exact solution for the wave equation with $c = 2$. 105
- FIGURE 6.14: $t = 0.6$ 106
- FIGURE 6.15: $t = 1$ 106
- FIGURE 6.16: Results for the 2D wave equation at two different times, $t = 0.6$ (a) and $t = 1$ (b). 106
- FIGURE 6.17: Relative ℓ_2 error for the 2D wave equation calculated at 125,000 evenly spaced grid points in the training domain. 107
- FIGURE 7.2: Results with $L = 2, 8$, and 32 subdomains for 7.10. (a) Training data and plot of exact $f(x)$. (b) Loss curves (7.9). (c) Plot of the outputs $f(x)$ and $\sum_{j=1}^L \omega_j(x) \mathcal{K}_j(x)$. (d) Pointwise errors $f(x) - \sum_{j=1}^L \omega_j(x) \mathcal{K}_j(x)$. (e) Scaling results for Test 1 with L subdomains. 116

- FIGURE 7.3: Results for Test 1 with noisy training data; (a) Example training data and plot of exact $f(x)$ with 9.6% mean relative noise. (b) Loss curves for an example training with 9.6% mean relative noise. (c) Plot of the outputs $f(x)$ and $\sum_{j=1}^L \omega_j(x) \mathcal{K}_j(x)$ with 9.6% mean relative noise. (d) Pointwise errors $f(x) - \sum_{j=1}^L \omega_j(x) \mathcal{K}_j(x)$ with 9.6% mean relative noise. (e) Relative ℓ_2 error of the KANs and FBKANs with respect to the magnitude of the noise added to the training data. 117
- FIGURE 7.4: Training loss curves for the data-driven test 2. 118
- FIGURE 7.5: Reference solution (left), predicted solutions (middle) and relative errors (right) for KAN-1 and FBKAN-1 for data-driven test 2. Note that the error plots have different color scales. 119
- FIGURE 7.6: Reference solution (left), predicted solutions (middle) and relative errors (right) for KAN-2 and FBKAN-2 for data-driven test 2. Note that the error plots have different color scales. 120
- FIGURE 7.7: Results for physics-informed test 1 with $L = 8$. The FBKAN clearly outperforms the KAN, reaching a lower loss with fewer grid points. 121
- FIGURE 7.8: Results for physics-informed test 2 with KAN-1 and FBKAN-1, and $a_1 = 1, a_2 = 4, L = 4$. The solution along the line $x = 0$ is given in the bottom left subfigure. Note that the error plots have different color scales. 122
- FIGURE 7.9: Results for physics-informed test 2 with KAN-1 and FBKAN-1, and $a_1 = 4, a_2 = 4, L = 4$. The solution along the line $x = 0$ is given in the bottom left subfigure. Note that the error plots have different color scales. 123
- FIGURE 7.10: Results for physics-informed test 2 with KAN-1 and FBKAN-1 and $a_1 = a_2 = 6$, with $L = 4$ and $L = 16$. The solution along the line $x = 0$ is given in the middle left subfigure. Note that the error plots have different color scales. 124
- FIGURE 7.11: Results for physics-informed test 3 with $c = \sqrt{2}$ and $L = 4$. Note that the error plots have different color scales. 125
- FIGURE 7.12: Results for physics-informed test 3 with $c = 2$ and $L = 4$. Note that the error plots have different color scales. 126

- FIGURE 7.13: Predicted solution by the initial PINN for the Helmholtz equation (Eq. 7.28) with $a_1 = 1, a_2 = 4$ (left) and learned subdomains by the POUnet (right). In this figure, the prediction on the left is used as data for the POUnet, which outputs the subdomains on the right. 130
- FIGURE 7.14: Comparison of the subdomains learned by a POUnet for the Helmholtz equation with no added penalty terms to enforce wider overlaps (left), and the subdomains learned by a POUnet with Laplacian penalty term (right). The Laplacian penalty term has a weight of $\lambda_{\text{Lap}} = 0.0001$. The subdomains with Laplacian penalty have a wider transition between subdomains, which aids in downstream FBPINN training. 134
- FIGURE 7.15: Graphical abstract of the ADD-FBPINN 136
- FIGURE 7.16: Predicted and exact results for the 1D Sine ODE with 2000 base points (left) and with 2000 base points + oversampling in the overlap (right). 143
- FIGURE 7.17: Collocation points used to solve the 1D Sine ODE with oversample factor of $\alpha = 4.0$. Collocation points are plotted along the exact solution. Blue points denote points in only one subdomain, and red points denote collocation points active in two subdomains. 144
- FIGURE 7.18: Predicted and exact results for the 1D Sine ODE with 2000 base points and an overlap width of $\delta = 0.1$ (left) and $\delta = 0.3$ (right). The wider overlap results in a more accurate solution. 144
- FIGURE 7.19: Distribution of collocation points for the 2D Helmholtz equation ablation study with an overlap width of $\delta = 0.09$. 10000 base collocation points are sampled over the entire domain using Hammersley sampling. Then, extra points are added to regions where 2 or more subdomains are active with an oversampling factor of $\alpha = 4.0$. 146
- FIGURE 7.20: Results for the 2D Helmholtz equation with $a_1 = 3$ and $a_2 = 8$. On the left, we see the predicted, exact, and absolute error for both Stage 0 (initial PINN), and Stage 1 (ADD-FBPINN). On the right, we see the learned subdomains used as the domain decomposition in Stage 1. 148

FIGURE 7.21: Results for the 2D Helmholtz equation with $a_1 = 1$ and $a_2 = 9$. On the left, we see the predicted, exact, and absolute error for both Stage 0 (initial PINN), and Stage 1 (ADD-FBPINN). On the right, we see the learned subdomains used as the domain decomposition in Stage 1. 149

FIGURE 7.22: Results for the 2D Poisson equation with multiscale features for Case 3 with $\omega = [1, 2, 4, 8]$. On the left, we have predicted, exact, and absolute error for both Stage 0 (baseline PINN) and Stage 1 (ADD-FBPINN). On the right, we see the learned domain decomposition used in Stage 1. In this case, the subdivision is still uniform, but the adaptive method is able to learn the ideal overlap width. 151

FIGURE 7.23: Results for the 2D Poisson equation with multiscale features for Case 3 with $\omega = [1, 2, 4, 8, 16]$. . On the left, we have predicted, exact, and absolute error for both Stage 0 (baseline PINN) and Stage 1 (ADD-FBPINN). On the right, we see the learned domain decomposition used in Stage 1. In this case, the subdivision favors more subdomains with sharper transitions. 152

FIGURE 7.24: Subdomains learned for various cases of Burgers equation by a POUnet. 153
 (a) Subdomains learned for Burgers Equation 7.32 with sharp gradient at $x = 0.5$ -

subfigure((b)). Subdomains learned for Burgers Equation 7.33 with sharp gradient at $x = -0.25$. 153

FIGURE 7.25: Predicted and exact solution and absolute error for Burgers equation comparing fixed domain decomposition with a standard FBPINN to that of an ADD-FBPINN. 154

FIGURE 7.26: Comparison between the fixed domain decomposition subdomains and the learned subdomains for Burgers equation with shock located at $x = -0.25$. This case highlights the ability of the adaptive domain decomposition to perform feature-aligned domain decomposition automatically. 155
 (a) Fixed domain decomposition used for Burgers Equation 7.32 with sharp gradient at $x = -0.25$ -

subfigure((b)). Subdomains learned for Burgers Equation 7.33 with sharp gradient at $x = -0.25$. 155

- FIGURE 8.1: Wave equation: Relative testing errors at 500 initial conditions, averaged over 5 different network initializations, and average elapsed time to run on a NVIDIA A100 GPU. 162
- FIGURE 8.2: Results using a DeepONet to solve the wave equation. Top Row: Exact solution (left), Predicted solution using the CK (middle), distribution of weights throughout the space-time domain (right). Middle row: Predicted solution using the NTK ($\alpha = 0.5$) (middle) and distribution of weights throughout the space-time domain. Bottom row: Eigenvalues of the sub-matrices of the NTK and CK corresponding to various loss terms (residual, initial conditions, and boundary conditions, respectively) calculated at the end of training. 162
- FIGURE 8.3: Spectrum of various kernels at regular intervals during training for the wave equation solved with a PI-DeepONet 163
- FIGURE 8.4: Burgers Equation: Testing errors on 500 initial conditions, with results for 5 randomly initialized networks 165
- FIGURE 8.5: Burger's equation spectrum at the end of training with $\nu = 0.01$ for the sub-matrices of the NTK or CK corresponding to the residual, initial condition, or boundary condition loss terms, respectively. 166
- FIGURE 8.6: Burger's equation spectrum at the end of training with $\nu = 0.001$ for the sub-matrices of the NTK or CK corresponding to the residual, initial condition, or boundary condition loss terms, respectively. 166
- FIGURE 8.7: Burger's equation spectrum at the end of training with $\nu = 0.0001$ for the sub-matrices of the NTK or CK corresponding to the residual, initial condition, or boundary condition loss terms, respectively. 166
- FIGURE 8.8: Eigenvalues of kernels during training for Burgers equation with viscosity $\nu = 0.01$. The eigenvalues in all cases converge quickly to stabilized value, indicating accurate and efficient training. 167
- FIGURE 8.9: Eigenvalues of kernels during training for Burgers equation with viscosity $\nu = 0.0001$. The eigenvalues during training are more dispersed and stabilize slower, if at all, when compare to other cases. This highlights a difficulty in training. 167

FIGURE 8.10: Sample results for Burgers equation with $\nu = 0.0001$	168
FIGURE 8.11: Burgers Equation: Testing errors on 500 initial conditions for $\nu = 0.0001$, with results for 5 randomly initialized networks. This plot includes errors for when the combination of NTK and CK is applied.	168
FIGURE 8.12: Stokes flow sample results exact, predicted, and absolute error for a representative boundary condition from the testing set using the NTK with $\alpha = 0.5$.	172
FIGURE 8.13: Stokes flow sample results exact, predicted, and absolute error for a representative boundary condition from the testing set using the CK with $\alpha = 0.5$.	173
FIGURE 8.14: Stokes flow: Box plots of testing errors over 100 various obstacles over 5 network runs. The points denoted by red crosses are outliers, which extend past 1.5x the inter-quartile range of the box.	174
FIGURE 8.15: Stokes flow: Mean testing errors at 500 testing ellipse sizes, averaged over 5 different network initializations, and average elapsed time to run on a NVIDIA A100 GPU. The NTK/CK case computes NTK weights for the residual loss terms, and CK weights for the boundary loss terms. All cases have $\alpha = 1$.	174
FIGURE 8.16: Stokes flow: Mean testing errors at 500 testing ellipse sizes, averaged over 5 different network initializations, and average elapsed time to run on a NVIDIA A100 GPU. The NTK/CK case computes NTK weights for the residual loss terms, and CK weights for the boundary loss terms. All cases have $\alpha = 0.5$.	175
FIGURE 8.17: Stokes flow. Top row: Spectrum of residual kernels with respect to u, v, c using the NTK and CK for a case in which the DeepONet fails to reach a solution. Bottom row: Spectrum of residual kernels with respect to u, v, c using the NTK and CK for a case in which the DeepONet trains successfully.	175
FIGURE 8.18: Eigenvalues during training for Stokes' flow	176
FIGURE 8.19: Stokes flow sample results for the NTK with $\alpha = 1$ and weight distribution for the residual kernels	176

LIST OF ABBREVIATIONS

- ADD-FBPINN An acronym for adaptive domain decomposition finite basis physics-informed neural network.
- CK An acronym for the conjugate kernel of a neural network.
- DeepONet An acronym for deep operator network.
- EM An acronym for the Euler Maruyama numerical method.
- FBPINN An acronym for finite basis physics-informed neural network.
- GBM An acronym for geometric Brownian motion.
- KAN An acronym for Kolmogorov-Arnold network.
- Mil An acronym for the Milstein numerical method.
- NTK An acronym for the neural tangent kernel of a neural network.
- ODE An acronym for ordinary differential equations.
- PDF An acronym for probability density function.
- PI-DeepONet An acronym for physics-informed deep operator network.
- PINN An acronym for physics-informed neural network.
- RK3 An acronym for the 3rd order Runge Kutta numerical method.
- SciML An acronym for scientific machine learning.
- SDE An acronym for stochastic differential equations.
- SH An acronym for the stochastic heun numerical method.

CHAPTER 1: INTRODUCTION

Many problems in the physical sciences demand mathematical models that can simulate and predict system behavior, such as solid materials modeling for nanomaterials manufacturing [4], ice sheet friction modeling for climate science [5], and particle diffusion in porous materials [6, 7]. As such, a major focus in the field of applied mathematics is the development of mathematical models with significant applications in mind and accompanying rigorous numerical analysis to guarantee their reliability and effectiveness. Numerical methods designed for the simulation, prediction, and analysis of physical systems range from classical numerical schemes, such as time integrators, to scientific machine learning approaches, such as physics-informed neural networks [8] and operator networks [9]. Machine learning models have been shown to effectively approximate solutions to equations that model fluid dynamics [10, 11]. However, the performance of these methods is not yet well understood, the computational cost is often high, and they require careful hyperparameter tuning. In this work, I address the recent debate regarding the efficacy of machine learning-based models over classical mathematical methods [12, 13, 14] through a suite of projects that examine the capabilities and shortcomings of both classical and modern methods for problems in fluid and molecular dynamics. Rather than arguing for one approach over the other, we explore the possibilities for each field to benefit from the other, including novel machine learning methods that are guided by classical error analysis and classical numerical techniques applied to improve the accuracy or efficiency of machine learning approaches.

In addition to making the case that mathematical modeling benefits from applying a variety of methods both classical and modern, the results of the seven projects in this

dissertation support the importance of three unifying pillars that must be balanced in order to create trustworthy mathematical modeling frameworks: numerical stability, modeling error, and computational cost. Numerical stability refers to the accuracy of a numerical scheme that approximates the solution to a system. Modeling error is introduced when a reduction is made to the full dynamics of a system in order to make the approximation of a solution more efficient. Computational cost refers to the computational resources required to solve and predict system behavior. These three facets are crucial to modeling frameworks regardless of numerical method used or application domain, which is highlighted in the variety of projects presented here.

With numerical stability, modeling error, and computational cost being three necessary quantities of interest when building trustworthy mathematical modeling frameworks, this dissertation follows the exploration of each via classical numerical schemes for stochastic differential equations, and then extends similar exploration to the machine learning setting. Chapters 2, 3 and 4 explore these modeling pillars in detail in the context of solving stochastic differential equations (SDEs) via classical methods, where techniques for analysis are well-known. We then pivot to the modern machine learning setting, beginning in Chapter 5 with a preliminary description of the field of scientific machine learning (SciML) and the models studied in this work. Chapters 6, 8, and 7 explore the importance of these pillars in the machine learning setting, exposing how our proposed modeling framework transcends method choice and application domain. Specifically, each chapter that follows is organized in the following manner:

- Chapter 2 focuses on a novel benchmark equation for the analysis of the performance of classical numerical SDE integrators.
- Chapter 3 discusses the use of the transition probability law as a short and long time error metric for the modeling error introduced when a model reduction is introduced via coarse-graining.

- Chapter 4 extended the study of coarse-graining approaches in the previous chapter with a leading first order estimate bound on the relaxation time of the full and resolved dynamics, and then uses this analysis to guide the selection of which variables to resolve and which to leave unresolved when making a reduction via coarse-graining.
- A review of prominent scientific machine learning methods for fluids and molecular dynamics is presented in Chapter 5. Common failure modes and best practices are discussed.
- 6 presents the use of a novel stacked multifidelity architecture to improve the accuracy of physics-informed neural networks and deep operator networks when the system dynamics are chaotic due to the occurrence of a fixed point, showcasing how techniques from classical numerical modeling can improve machine learning approaches.
- Chapter 7 presents another study on the application of classical techniques to improve mathematical models based on machine learning techniques. I explore the use of classical domain decomposition techniques for Kolmogorov-Arnold networks (KANs) and develop an adaptive domain decomposition technique for finite basis PINNs (FBPINNs).
- In Chapter 8, I present the use of a theory-based, adaptive loss weighting scheme used to improve the accuracy and efficiency of training physics-informed deep operator networks.
- Finally, I conclude with a discussion regarding the implications of this work on fluid and molecular dynamics modeling and provide areas for future work in Chapter 9.

CHAPTER 2: NUMERICAL STABILITY REVISITED FOR CLASSICAL SDE INTEGRATORS

This chapter is based on the work published in *Numerical Stability Revisited: A Family of Benchmark Problems for the Analysis of Explicit Stochastic Differential Equation integrators* by Thomas Hudson, Xingjie Li, and Sarah Helfert [15]. In this work, we explore four classical numerical integrators for SDEs and provide a novel benchmark equation with which to analyze their performance. The novel benchmark we propose improves upon the standard benchmark, geometric Brownian motion, because it better captures the overall performance of the schemes in a nonlinear setting. This work highlights the importance of carefully selecting benchmark equations and parameters for numerical methods that highlight various application contexts, because lower-order schemes can outperform higher-order schemes in some cases. This is a crucial insight that can offer improved analysis in the machine learning setting as well, as shown in later projects. Further, this project highlights the nuances of capturing the numerical error that is introduced when we approximate the solution to a system, which is the first essential pillar of trustworthy mathematical modeling frameworks. In later chapters, we introduce studies on modeling error and efficiency as equally crucial pieces of modeling frameworks.

SDE models are important for modeling a wide range of real-world phenomena, and combine both deterministic and random effects into a model describing the evolution of degrees of freedom in time. In many cases, SDE models are ergodic, entailing that the law of the process converges to an equilibrium distribution independently of the initial condition. In practice, ergodicity is a desirable property for real-world applications, with the equilibrium distribution representing a form of thermalised

steady-state for a system.

To simulate and predict the behavior of SDE models, numerical integration algorithms are needed, and a wide range of algorithms have therefore been developed to simulate SDE models. Even when an SDE is ergodic, it has long been recognized that a numerical discretization of the evolution may not reproduce the equilibrium distribution, leading to biased long-time statistics. For example, when the drift and diffusion coefficients are only *locally Lipschitz*, explicit schemes can fail to be ergodic and therefore yield biased asymptotic statistics, even when the underlying SDE is itself geometrically ergodic [16, 17, 18, 19]. On the other hand, when the coefficients are globally Lipschitz, explicit schemes admit strong and weak convergence guarantees and can be proved to reproduce the equilibrium distribution when time step size h tends to 0 [2, 1, 20]. Moreover, for a fixed simulation time T , higher-order schemes are provably more accurate than lower-order schemes as the time step tends to zero. However, the associated error bounds typically involve prefactors that depend on T , so these results do not by themselves imply that a higher-order method will consistently outperform a lower-order one in the long-time limit for a fixed time-step size. Motivated by this, we are interested here in comparing the long-time behavior of explicit schemes of a range of orders in the regime where $T \rightarrow \infty$ while the time step h remains moderately large, thereby focusing on the ability of the schemes we consider to approximate the invariant measure and the resulting asymptotic statistics.

We note at this stage that where discretizations of SDEs are used as a method to generate samples of a particular distribution with a known density, Monte Carlo algorithms often use an accept-reject or Metropolis step to counteract the bias inherent in the numerical discretization [21, 22]. On the other hand, when we are interested not only in sampling the equilibrium distribution but also in the dynamics of the process, performing accept-reject steps on entire trajectory segments may be costly, as they involve discarding expensive-to-compute trajectories of high-dimensional systems of

SDEs. As an alternative, we ask how accurately and stably numerical schemes are able to capture asymptotic statistical properties of the underlying SDE model. If we are able to quantify the errors committed, and these are at an acceptable level for the required application, we may as a result be able to save extra computational work. Furthermore, stochasticity and stiffness appear in a wide range of physical and chemical systems, and the time-step size is often severely restricted to maintain numerical stability and accuracy when integrating stiff stochastic differential equations with fast diffusion process [23, 24, 25].

Classically, the stability of SDE schemes has been measured through an approach known as mean-square stability analysis [26, 27, 28, 29]. These approaches have tended to use geometric Brownian motion (GBM) as a benchmark, i.e.

$$dq_t = -q_t dt + \eta q_t dW_t,$$

extending the concept of linear stability from deterministic ordinary differential equations (ODEs) [27, 1]. In this work, we revisit the underlying assumptions of this stability analysis, and propose a more generic benchmark SDE problem for the study of statistical accuracy and stability properties, namely the one-dimensional equation

$$dq_t = -q_t dt + (1 + \eta q_t) dW_t.$$

Here, η is a real parameter which controls the relative strength of the multiplicative noise and the stiffness of the deterministic drift. By non-dimensionalising, we show that this benchmark arises generically from a general first-order SDE in one dimension with affine coefficients. While the study of the GBM problem benefits from a simple exact expression for solution trajectories in terms of Brownian motion, we will argue below that since it represents a less generic situation, it does not necessarily reflect the performance of numerical integration schemes in a realistic nonlinear setting.

Despite their superficial similarity, as a stark illustration of the contrast between the two problems, the equilibrium distribution for GBM is always concentrated at $x = 0$ when it exists, while our proposed benchmark includes a wide range of algebraic tail behavior for the resulting equilibrium distribution.

At this stage, we briefly comment on our reasons for selecting a one-dimensional rather than a multi-dimensional test case. In higher dimensions, a non-dimensionalisation strategy similar to that which we pursue below to derive our benchmark will lead to a much richer parametric representation, which is therefore more complex to analyze. These challenges are already recognised in the literature on mean-squared stability; see for example [30, 28], and further discussion the complexities faced in higher-dimensional cases is provided when deriving our benchmark below.

After deriving the benchmark problem, we use it to analyze the long-time performance of some well-established explicit integration schemes. Our focus on explicit schemes is motivated by the need for integrators which avoid costly implicit steps in high dimensional problems. While there is no simple exact formula for solutions of our test problem, we will see that it is nevertheless simple enough to allow for the explicit computation of both the equilibrium distribution and the evolution of the low-order moments of the law as the process evolves. We choose the latter as metrics for assessing the accuracy of the schemes, reflecting the sorts of statistics which are often of interest to practitioners.

Along with a discussion of the long-time statistical accuracy of the numerical methods considered, we obtain results characterising the range of time-steps for which the methods are stable in the sense that they preserve the boundedness of first and second moments as the number of time steps simulated tends to infinity. This is a feature which we term *statistical stability* to distinguish it from other differing notions [27, 31, 1]. The ranges of time step we find provide us with a meaningful way to compare the stability of methods as the parameter η varies, allowing us to represent-

ing a range of relative strengths for the multiplicative noise and drift. In modeling contexts where long trajectories are important in order to enable the observation of phenomena which would otherwise be out of reach, understanding and predicting the stability of methods are important as a way to allow for increased time-step sizes, which trade accuracy for physical insight which would otherwise not be available.

Finally, to verify the analytical results we obtain, we conclude by performing a range of numerical experiments, and discuss the consequences of our results for a more general application to a nonlinear example.¹

In Section 2.1, we present the preliminaries on stochastic differential equations (SDEs) that are the focus of this work. In Section 2.2, we review the numerical schemes we study, and give an overview of established results on their local accuracy properties. In Section 2.4, we use a nondimensionalisation argument to derive our test problem from a general linear SDE, discussing various exactly computable properties of the resulting equation. In Section 2.5, we derive expressions for the evolution of discrete statistical moments under various numerical integrations, comparing to the continuous counterparts for the benchmark problem, verifying the results numerically and studying the corresponding stability regions. In Section 2.6.3, we demonstrate the utility of our results in the context of a more realistic setting of an autonomous equation where the drift and diffusion coefficients are both nonlinear.

2.1 Stochastic Differential Equations

This work focuses on approximating the solution to stochastic differential equations of the following form

$$dq_t = f(q_t)dt + g(q_t)dW_t, \quad t \in [0, T]. \quad (2.1)$$

In this work, we assume the solution is subject to a deterministic initial condition x_0 .

¹Accompanying codes can be found at https://github.com/XingjieHelenLi/NumSDE_Revisit.

For the purposes of comparing the numerical schemes we study, we will discretize the simulation time interval into subintervals of fixed length $h = T/N$, and define a discrete time grid $t_n = nh$ for $n = 0, \dots, N$. For the different schemes we discuss, x_n will always be used to refer to the approximation of the solution generated at time t_n . For convenience in describing the numerical schemes below, we also introduce the notation

$$f_n := f(x_n), \quad \text{and} \quad f'_n := f'(x_n),$$

and use analogous expressions g_n , g'_n and g''_n for the relevant functions evaluated at x_n . We also introduce sequences of independent identically distributed Gaussian random variables ΔW_n and $\Delta \tilde{W}_n$ with mean zero and variance h , i.e. $\Delta \tilde{W}_n, \Delta W_n \sim \mathcal{N}(0, h)$.

2.2 Overview of numerical schemes

We consider the stochastic initial value problem for the scalar autonomous Itô SDE given by

$$dq_t = f(q_t)dt + g(q_t)dW_t, \quad t \in [0, T]. \quad (2.2)$$

We will assume throughout that the solution is subject to a deterministic initial condition, x_0 .

For the purposes of comparing the numerical schemes we study, we will discretize the simulation time interval into subintervals of fixed length $h = T/N$, and define a discrete time grid $t_n = nh$ for $n = 0, \dots, N$. For the different schemes we discuss, x_n will always be used to refer to the approximation of the solution generated at time t_n . For convenience in describing the numerical schemes below, we also introduce the notation

$$f_n := f(x_n), \quad \text{and} \quad f'_n := f'(x_n),$$

and use analogous expressions g_n , g'_n and g''_n for the relevant functions evaluated at x_n . We also introduce sequences of independent identically distributed Gaussian random

variables ΔW_n and $\Delta \tilde{W}_n$ with mean zero and variance h , i.e. $\Delta \tilde{W}_n, \Delta W_n \sim \mathcal{N}(0, h)$.

We select four classical explicit numerical schemes to study with a range of levels of accuracy. In particular, we consider:

1. The Euler–Maruyama (EM) method [32], [1, §9.1]:

$$x_{n+1} = x_n + f_n h + g_n \Delta W_n. \quad (\text{EM})$$

2. The Milstein (Mil) method [1, §10.3]:

$$x_{n+1} = x_n + f_n h + g_n \Delta W_n + \frac{1}{2} g'_n g_n (\Delta W_n^2 - h). \quad (\text{Mil})$$

3. The Stochastic Heun (SH) method [27]:

$$x_{n+1} = x_n + \frac{1}{2} [F_1 + F_2] h + \frac{1}{2} [G_1 + G_2] \Delta W_n, \quad (\text{SH})$$

where setting $F(x) := f(x) - \frac{1}{2} g'(x) g(x)$, the coefficients in the relation above are defined to be

$$\begin{aligned} F_1 &= F(x_n), & G_1 &= g(x_n), \\ F_2 &= F(x_n + F_1 h + G_1 \Delta W_n), & G_2 &= g(x_n + F_1 h + G_1 \Delta W_n). \end{aligned}$$

4. The improved 3-stage Runge-Kutta (RK3) method [2, 27]:

$$\begin{aligned} x_{n+1} &= x_n + \frac{1}{4} (F_1 + 3F_3) h + \frac{1}{4} [G_1 + 3G_3] \Delta W_n \\ &\quad + \frac{1}{2\sqrt{3}} \left(f'_n g_n - g'_n f_n - \frac{1}{2} g''_n g_n^2 \right) h \Delta \tilde{W}_n, \end{aligned} \quad (\text{RK3})$$

Table 2.1: A summary of the strong and weak convergence orders of the methods studied. The first two rows correspond to results proved in [1], while the latter two rows correspond to results proved in [2, 3].

Method	Strong order	Weak order
Euler-Maruyama (EM)	0.5	1
Milstein (Mil)	1	1
Stochastic Heun (SH)	1	2
3-stage Runge-Kutta (RK3)	1	2

where we again denote $F(x) := f(x) - \frac{1}{2}g'(x)g(x)$, and the coefficients are

$$\begin{aligned}
F_1 &= F(x_n), & G_1 &= g(x_n), \\
F_2 &= F(x_n + \tfrac{1}{3}F_1h + \tfrac{1}{3}G_1\Delta W_n), & G_2 &= g(x_n + \tfrac{1}{3}F_1h + \tfrac{1}{3}G_1\Delta W_n), \\
F_3 &= F(x_n + \tfrac{2}{3}F_2h + \tfrac{2}{3}G_2\Delta W_n), & G_3 &= g(x_n + \tfrac{2}{3}F_2h + \tfrac{2}{3}G_2\Delta W_n).
\end{aligned}$$

Established results on the strong and weak orders of these schemes are summarized in Table 2.1, and the accuracy in terms of time-stepping h is summarized in Table 2.2. As we can see, these schemes range significantly across the convergence order that they exhibit.

2.3 Asymptotic moment stability for Geometric Brown Motion

In this section, we provide analysis of the stability of the schemes we study for GBM as a benchmark, to later be compared with our novel benchmark.

2.3.1 Evolution of moments for geometric Brownian motion

To provide a clear comparison between the previous GBM benchmark

$$dq_t = -q_t dt + \eta q_t dW_t$$

used to define MS-stability and our benchmark (2.10), we compute the analytic and numerical first and second moments of various schemes for the GBM in this section. Taking the GBM from a deterministic initial position $x(0) = x_0$ and assuming that

$\eta^2 < 2$, we have

$$dq_t = -q_t dt + \eta q_t dW_t.$$

In this case, the Fokker-Planck equation for the probability density $p(x, t)$ of the state is given by

$$\partial_t p = \partial_x (xp + \partial_x (\frac{1}{2}\eta^2 x^2 p)).$$

As in Section 2.4.3, we derive an equation for the first moment by taking time derivatives and integrating by parts, giving

$$\frac{d}{dt}\mu^{(1)}(t) = \int_{-\infty}^{\infty} x \partial_t p dx = \int_{-\infty}^{\infty} x \partial_x (xp) dx + \frac{\eta^2}{2} \int_{-\infty}^{\infty} x \partial_x^2 (x^2 p) dx = -\mu^{(1)}(t);$$

this equation has analytic solution $\mu^{(1)}(t) = \mu^{(1)}(0)e^{-t}$. Similarly, the second moment satisfies

$$\frac{d}{dt}\mu^{(2)}(t) = \int_{-\infty}^{\infty} x^2 \partial_t p dx = \int_{-\infty}^{\infty} x^2 \partial_x (xp) dx + \frac{\eta^2}{2} \int_{-\infty}^{\infty} x^2 \partial_x^2 (x^2 p) dx = -(2 - \eta^2)\mu^{(2)}(t)$$

with solution $\mu^{(2)}(t) = \mu^{(2)}(0)e^{(-2+\eta^2)t}$. Since the initial condition is assumed to be deterministic, $p(x, 0) = \delta_{x_0}(x)$, and we have the following analytical expressions and asymptotic values of the first and second moments:

$$\mu^{(1)}(t) = x_0 e^{-t} \rightarrow 0 \quad \text{and} \quad \mu^{(2)}(t) = x_0^2 e^{-(2-\eta^2)t} \rightarrow 0 \quad \text{as} \quad t \rightarrow \infty.$$

2.3.2 Asymptotic moment stability regions for various schemes

In this section, we compute the asymptotic first and second-moment stability regions for the geometric Brownian motion under the four numerical schemes of interest.

2.3.2.1 Euler Maruyama (EM)

Taking expectations and using the scheme definition (EM), the first moment evolves according to

$$\mu_{n+1}^{(1)} = \mathbb{E}[x_{n+1}] = \mathbb{E}[(1-h)x_n + \eta x_n h^{1/2} \Delta W_n] = (1-h)\mu_n^{(1)}, \quad (2.3)$$

and thus EM is first-moment asymptotically stable for $|1-h| < 1$, i.e., $0 < h < 2$.

Following a similar approach for the second moment, we have

$$\begin{aligned} \mu_{n+1}^{(2)} &= \mathbb{E}[(x_n^2)(1-h + \eta h^{1/2} \Delta W_t)^2] \\ &= \mathbb{E}[X_n^2] \left((1-h)^2 + 2\eta h^{1/2}(1-h)\mathbb{E}[\Delta W_n] + \eta^2 h \mathbb{E}[(\Delta W_n)^2] \right) \\ &= \mu_n^{(2)} \left((1-h)^2 + \eta^2 h \right), \end{aligned} \quad (2.4)$$

which leads to the stability requirement that

$$0 < h < 2 - \eta^2.$$

These stability requirements are identical to those found for the benchmark (2.10) in Section 2.5. In addition, the EM scheme for the GBM satisfies

$$\text{EM:} \quad \mu_\infty^{(1)} = 0, \quad \mu_\infty^{(2)} = 0$$

which show that EM scheme recovers the exact analytic values of asymptotic first and second moments for GBM.

2.3.2.2 Milstein method (Mil)

We see a similar phenomenon for the Mil method applied to the GBM. The Mil updates step for GBM is

$$X_{n+1} = X_n \left(1 - h + \eta \Delta W_n + \frac{1}{2} \eta^2 ((\Delta W_n)^2 - h) \right).$$

Taking the first moment, we have

$$\mu_{n+1}^{(1)} = \mu_n^{(1)}(1 - h). \quad (2.5)$$

This leads to the stability condition for the first moment $0 < h < 2$.

Taking the second moment, the recursion is

$$\mu_{n+1}^{(2)} = \mu_n^{(2)} \left(1 + (\eta^2 - 2)h + \left(1 + \frac{1}{2}\eta^4\right)h^2 \right), \quad (2.6)$$

which results in the asymptotic stability condition for the second moment to be

$$0 < h < \frac{2 - \eta^2}{1 + \frac{1}{2}\eta^4}.$$

Again, the first and second stability conditions for GBM under Mil are identical to the stability conditions we found for the benchmark (2.10). In addition for Mil, we have

$$\text{Mil:} \quad \mu_{\infty}^{(1)} = 0, \quad \mu_{\infty}^{(2)} = 0,$$

again, the Mil scheme recovers the exact asymptotic first and second moments without bias for GBM.

2.3.2.3 Stochastic Heun (SH)

The evolution of the first moment under SH reads

$$\mu_{n+1}^{(1)} = \left(1 - h + \frac{1}{2} \left(1 + \frac{1}{2} \eta^2\right)^2 h^2\right) \mu_n^{(1)} = \left(1 - h + \frac{1}{2} h^2 + \frac{1}{2} \eta^2 h^2 + \frac{1}{8} \eta^4 h^2\right) \mu_n^{(1)}, \quad (2.7)$$

which gives the stability region

$$0 < h < \frac{2}{\left(1 + \frac{1}{2} \eta^2\right)^2} = \frac{8}{\eta^4 + 4\eta^2 + 4}$$

Similarly, we have the recurrence relation for the second moment:

$$\begin{aligned} \mu_{n+1}^{(2)} = & \left(1 + (\eta^2 - 2)h + \left(2 - \eta^2 - \frac{1}{4}\eta^4\right)h^2\right. \\ & \left. + \left(\frac{1}{4}\eta^6 + \frac{3}{4}\eta^4 - 1\right)h^3 + \left(\frac{1}{64}\eta^8 + \frac{1}{8}\eta^6 + \frac{3}{8}\eta^4 + \frac{1}{2}\eta^2 + \frac{1}{4}\right)h^4\right) \mu_n^{(2)}, \end{aligned}$$

which yields the stability condition

$$\left|1 + (\eta^2 - 2)h + \left(2 - \eta^2 - \frac{1}{4}\eta^4\right)h^2 + \left(\frac{1}{4}\eta^6 + \frac{3}{4}\eta^4 - 1\right)h^3 + \left(\frac{1}{64}\eta^8 + \frac{1}{8}\eta^6 + \frac{3}{8}\eta^4 + \frac{1}{2}\eta^2 + \frac{1}{4}\right)h^4\right| < 1.$$

These stability conditions for the first and second moments under SH are remain the same when compared to the results for the benchmark (2.10). Moreover, we have the asymptotic result that

$$\text{SH:} \quad \mu_{\infty}^{(1)} = 0, \quad \mu_{\infty}^{(2)} = 0;$$

so the exact analytic values of asymptotic moments are recovered without bias for GBM under SH discretization scheme.

2.3.2.4 3-stage Runge-Kutta (RK3)

For RK3, we have for the first moment

$$\begin{aligned}\mu_{n+1}^{(1)} &= \left(1 - h + \left(\frac{1}{2} - \frac{1}{8}\eta^4\right)h^2 - \left(\frac{1}{48}\eta^6 + \frac{1}{8}\eta^4 + \frac{1}{4}\eta^2 + \frac{1}{6}\right)h^3\right)\mu_n^{(1)} \\ &= \left(1 - h + \frac{1}{8}h^2(4 - \eta^4) - \frac{1}{48}h^3(2 + \eta^2)^3\right)\mu_n^{(1)}.\end{aligned}\tag{2.8}$$

which indicates the first-moment is asymptotically stable when

$$\left|1 - h + \frac{1}{8}h^2(4 - \eta^4) - \frac{1}{48}h^3(2 + \eta^2)^3\right| < 1.$$

Next for the second moment, we have

$$\mu_{n+1}^{(2)} = R^{(2)}(h, \eta) \mu_n^{(2)},$$

where $R^{(2)}$ is a polynomial in h of the form

$$R^{(2)}(h, \eta) = a_0 + a_1h + a_2h^2 + a_3h^3 + a_4h^4 + a_5h^5 + a_6h^6,$$

with coefficients

$$\begin{aligned}a_0 &= 1, \quad a_1 = \eta^2 - 2, \quad a_2 = \frac{1}{4}\eta^4 - 2\eta^2 + 2, \\ a_3 &= -\frac{1}{8}\eta^6 + \frac{3}{2}\eta^2 - \frac{4}{3}, \quad a_4 = \frac{9}{64}\eta^8 + \frac{7}{24}\eta^6 - \frac{3}{8}\eta^4 - \frac{1}{2}\eta^2 + \frac{7}{12}, \\ a_5 &= \frac{1}{48}\eta^{10} + \frac{5}{32}\eta^8 + \frac{5}{12}\eta^6 + \frac{5}{12}\eta^4 - \frac{1}{6}, \\ a_6 &= \frac{1}{2304}\eta^{12} + \frac{1}{192}\eta^{10} + \frac{5}{192}\eta^8 + \frac{5}{72}\eta^6 + \frac{5}{48}\eta^4 + \frac{1}{12}\eta^2 + \frac{1}{36}.\end{aligned}$$

Again, stability for the second moment requires that $|R^{(2)}(h, \eta)| < 1$. Both stability conditions for the first and second moments for GBM under RK3 are still the same

compared to the benchmark (2.10). Meanwhile, we have

$$\text{RK3:} \quad \mu_{\infty}^{(1)} = 0, \quad \mu_{\infty}^{(2)} = 0,$$

which shows that RK3 recovers the exact second moment without bias for GBM.

For each of the schemes studied, the above computations show that the stability requirements for the GBM benchmark take the same form as for our benchmark equation (2.10) in terms of the parameter η and timestep h . A stark difference however is that all schemes are asymptotically exact in terms of the first and second moment for GBM, with no bias. In part, this highlights the particularly special feature of GBM, which is that the drift and diffusion both vanish at the same point, and hence when it exists, the equilibrium measure is concentrated. As argued in Section 2.4.1, this behavior is rather special when considering an SDE with affine coefficients, highlighting the need for an alternative benchmark.

2.4 Derivation and properties of benchmark problem

To analytically compare the methods (EM), (Mil), (SH) and (RK3), we focus on a test problem where f and g are time-independent and affine in the spatial variable, i.e.

$$f(\xi) = -\alpha - \beta\xi \quad \text{and} \quad g(\xi) = \gamma + \delta\xi.$$

In other words, we consider the SDE

$$d\xi_{\tau} = -(\alpha + \beta\xi_{\tau})d\tau + (\gamma + \delta\xi_{\tau})dW_{\tau}, \tag{2.9}$$

which is the most general form of autonomous SDE in one variable with coefficients which are affine in ξ_{τ} , driven by a single Brownian motion. This problem can be seen as the linearised form of the general nonlinear autonomous SDE (2.2) about some fixed spatial point.

In this section, we perform what amounts to a non-dimensionalisation of this equation, which allows us to reduce the equation to a single parameter family in the most general case. We also discuss properties of the resulting reduced equation.

2.4.1 Reduction of cases

We proceed by rescaling and shifting time and space coordinates from (ξ, τ) to (x, t) . To do so, we define $\xi_\tau := Aq_t + B$, and we introduce the time rescaling $\tau = Tt$. Here A , B and T are real parameters to be chosen, and $T > 0$ is positive, so that the direction of time is preserved. Substituting appropriately, and using the properties of Brownian motion, we obtain the following equation in the new coordinates

$$dq_t = -\left(TA^{-1}(\alpha + \beta B) + \beta Tq_t\right)dt + T^{1/2}A^{-1}(\gamma + \delta B + \delta Aq_t)dW_t.$$

If $\beta \neq 0$, then we can set $B = -\alpha/\beta$ and $T = 1/|\beta|$ and obtain

$$dq_t = -\frac{\beta}{|\beta|}q_tdt + \left(\frac{\beta\gamma - \alpha\delta}{A\beta|\beta|^{1/2}} + \frac{\delta}{|\beta|^{1/2}}q_t\right)dW_t.$$

Next, as long as $\beta\gamma - \alpha\delta \neq 0$, we can set $A = \frac{\beta\gamma - \alpha\delta}{\beta|\beta|^{1/2}}$ and $\eta := \frac{\delta}{|\beta|^{1/2}}$ to obtain

$$dq_t = \pm q_tdt + (1 + \eta q_t)dW_t, \tag{2.10}$$

where the sign of the term in front of the drift term is opposite to that of β . Equation (2.10) provides the most generic dimensionless form of (2.9) in this sense, but there are other possible cases:

- If $\beta \neq 0$ but $\beta\gamma - \alpha\delta = 0$, then we can rescale to obtain geometric Brownian motion

$$dq_t = \pm q_tdt + \eta q_t dW_t, \tag{2.11}$$

where again $\eta = \frac{\delta}{|\beta|^{1/2}}$.

- If $\beta = 0$ but $\delta \neq 0$, then we can set $B = -\gamma/\delta$, $T = |\delta|^{-1/2}$ and $A = -\alpha|\delta|^{1/2}$ to obtain

$$dq_t = dt \pm q_t dW_t. \quad (2.12)$$

- Finally, we could have $\beta = 0$ and $\delta = 0$, resulting in an equation which is simply a translation of a Brownian motion.

In the existing literature on the numerical analysis of SDEs with multiplicative noise, geometric Brownian motion (2.11) has been used as a common test equation for numerical methods. A likely reason for the use of geometric Brownian motion is the availability of an analytical solution; the derivation above shows however that both this case and the case where (2.12) is the resulting equation form sets of zero measure in the four-dimensional space of parameters α , β , γ and δ . As such, we choose to focus on consideration of (2.10) for $\eta \in$ throughout the remainder of this work. We choose also to only consider the case with drift term being $-x_t dt$ in (2.10), which guarantees ergodicity of the dynamics with respect to an equilibrium distribution; this is discussed further in the following section.

As discussed in the introduction, we highlight some complexities which would arise in the more general case where we consider an SDE in d dimensions, which motivate our choice to focus on a one-dimensional case. Consider the general d -dimensional system of SDEs where the drift and diffusion are affine, i.e.

$$d\xi_\tau^i = \left(\alpha^i + \sum_{j=1}^d \beta^{ij} \xi_\tau^j \right) d\tau + \sum_{j=1}^d \left(\gamma^{ij} + \sum_{k=1}^d \delta^{ijk} \xi_\tau^k \right) dW_\tau^j,$$

where α^i , β^{ij} , γ^{ij} and δ^{ijk} are real coefficients of the system for $i, j, k \in \{1, \dots, d\}$. In this case, the number of independent coefficients is $d + 2d^2 + d^3$. In higher dimensions,

a general redefinition of the spatial variables takes the form

$$\xi_\tau^i := A^i + \sum_{j=1}^d B^{ij} x_t^j, \quad i = 1, \dots, d,$$

along with the time rescaling $\tau = Tt$. This provides us with the flexibility to scale out $d^2 + d + 1$ parameters in the most generic case, but under such a rescaling, there would remain $d^3 + d^2 - 1$ independent dimensionless parameters governing the system. Even in two dimensions, there are therefore 11 possible cases (although we note further reductions may be possible through finding other equivalences). This rapid growth in the number of cases with dimension justifies our choice to focus on a one-dimensional cases where there is exactly 1 dimensionless variable for simplicity. Note that in the case where $d = 1$, the above argument predicts that we have exactly one free dimensionless parameter, just as derived above.

For comparison, the particular case considered in [28] corresponds to a reduction of the two-dimensional case with the further requirement that β is symmetric, γ^{ij} is zero, $\delta^{ijk} = 0$ for $k = 2$, and $i, j \in \{1, 2\}$. As in the case of geometric Brownian motion in one dimension, this is clearly not a case that will arise generically from a linearisation of nonlinear drift and diffusion about a fixed spatial point, but the analysis remains challenging: there are 9 parameters that can be reduced to 2 by scaling in space and time. This conclusion can effectively be seen from the result of Theorem 1 in [28], which relies upon two independent conditions holding in order to guarantee stability.

2.4.2 Equilibrium distribution

Since our focus is on asymptotic statistical properties of the dynamics and the accurate recovery of these statistics by numerical schemes, we now consider the ergodicity properties of our reduced equation (2.10). There are a range of established approaches which can be used to demonstrate that the SDE is ergodic with respect

to an equilibrium distribution. One such approach is provided by the results of [17], which in turn relies upon the methodology introduced in [33, 34, 35]. The Fokker–Planck equation which governs the probability density function p of solutions to the model problem (2.10) is the PDE

$$\partial_t p = \partial_x \left(xp + \partial_x \left(\frac{1}{2}(1 + \eta x)^2 p \right) \right). \quad (2.13)$$

In particular, this approach to showing ergodicity requires verification a *minorisation condition* for the transition kernel of the process, and the construction of a *Lyapunov function*. In the case of the equation under consideration, the minorisation condition can be shown to hold as a consequence of Hörmander’s theorem [36, 34], and $V(x) = \sqrt{1 + x^2}$ serves as an appropriate Lyapunov function. However, a detailed verification of all conditions required to establish ergodicity is beyond the scope of the present work.

Instead, we focus here on formally computing the equilibrium distribution for use as a comparator for asymptotic statistics. When the SDE is ergodic, there exists a steady-state solution p_∞ of (2.13) which is Lebesgue integrable on the real line such that, for most sufficiently regular initial distributions, the probability density p_t converges to p_∞ as $t \rightarrow +\infty$. Because p_∞ is time independent, the steady state of (2.13) becomes

$$0 = \partial_x \left(xp_\infty + \partial_x \left(\frac{1}{2}(1 + \eta x)^2 p_\infty \right) \right) = \partial_x \left(\left(\eta + (\eta^2 + 1)x \right) p_\infty + \frac{1}{2}(1 + \eta x)^2 \partial_x p_\infty \right),$$

which holds wherever p_∞ is supported. We note that the coefficient of the highest order terms in this equation vanish when $x = -\frac{1}{\eta}$, and this reflects the compact support of the solution p_∞ when $\eta \neq 0$. Integrating once and rearranging, we have

$$\frac{K}{(1 + \eta x)^2} = \frac{2\eta + 2(\eta^2 + 1)x}{(1 + \eta x)^2} p_\infty + \partial_x p_\infty,$$

where K is some constant. Multiplying through by the integrating factor

$$I(x) := \exp \left(\int^x \frac{2\eta + 2(\eta^2 + 1)u}{(1 + \eta u)^2} du \right) = \exp \left(\frac{2}{\eta^2} \frac{1}{1 + \eta x} \right) (1 + \eta x)^{2 + \frac{2}{\eta^2}},$$

we have that

$$K \exp \left(\frac{2}{\eta^2} \frac{1}{1 + \eta x} \right) (1 + \eta x)^{\frac{2}{\eta^2}} = \partial_x (I(x) p_\infty(x)).$$

We note that the term on the right hand side is not integrable in a neighborhood of $x = -\frac{1}{\eta}$, so we must have that $K = 0$. Integrating again it follows that

$$p_\infty(x) = \frac{C}{I(x)} = C \exp \left(-\frac{2}{\eta^2} \frac{1}{1 + \eta x} \right) (1 + \eta x)^{-2 - \frac{2}{\eta^2}},$$

for some coefficient C , and to normalize appropriately, we can integrate. For now, assuming that $\eta > 0$ and making the change of variable $y = \frac{2}{\eta^2} \frac{1}{1 + \eta x}$, we can express the integral using the Gamma function as

$$\int_{-\frac{1}{\eta}}^{\infty} \exp \left(-\frac{2}{\eta^2} \frac{1}{1 + \eta x} \right) (1 + \eta x)^{-2 - \frac{2}{\eta^2}} dx = 2^{-\frac{2}{\eta^2}} \eta^{-1 + \frac{4}{\eta^2}} \Gamma\left(\frac{2}{\eta^2}\right).$$

The density of the properly normalized form of the equilibrium distribution is therefore

$$p_\infty(x) = \begin{cases} \frac{2^{\frac{2}{\eta^2}} \eta^{1 - \frac{4}{\eta^2}}}{\Gamma(\frac{2}{\eta^2})} \exp \left(-\frac{2}{\eta^2} \frac{1}{1 + \eta x} \right) (1 + \eta x)^{-2 - \frac{2}{\eta^2}} & x > -\frac{1}{\eta}, \\ 0 & x \leq -\frac{1}{\eta}. \end{cases} \quad (2.14)$$

If $\eta < 0$, then we obtain the same form of the equilibrium distribution, but x and η are replaced by $-x$ and $-\eta$ in the definition (2.14) above. In the special case where $\eta = 0$, the SDE (2.10) has an additive noise term, and the distribution becomes a standard Gaussian with zero mean and variance $\frac{1}{2}$.

We note that the resulting distribution (2.14) is in fact a translated form of the inverse Gamma distribution, which is commonly parametrised with probability distribution function

$$\tilde{p}(y; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{1}{y^{\alpha+1}} \exp(-\beta/y),$$

where $\alpha = 1 + \frac{2}{\eta^2}$ and $\beta = \frac{1}{\eta^2}$, and y is related to x through the transformation $y = 1 + \eta x$. Due to the algebraic decay of the tail of this distribution as $x \rightarrow +\infty$, we note that the k th moment of the equilibrium distribution is only well-defined when $x^k p_\infty(x)$ is Lebesgue integrable, which requires

$$k - 2 - \frac{2}{\eta^2} < -1, \quad \text{i.e.} \quad |\eta| < \sqrt{\frac{2}{k-1}}.$$

When they exist, we can also explicitly calculate that the first and second moments of the equilibrium distribution. These are

$$\mu_\infty^{(1)} = \int_{-\infty}^{\infty} x p_\infty dx = 0 \quad \text{and} \quad \mu_\infty^{(2)} = \int_{-\infty}^{\infty} x^2 p_\infty dx = \frac{1}{2 - \eta^2}; \quad (2.15)$$

explicit representations as rational functions of η can also be computed, but our focus will be on the first and second moments. We note that, when the second moment is finite, we can infer the value of η up to its sign, and hence identify the equation governing the dynamics within the class of reduced equations (2.10).

2.4.3 Evolution of moments

In addition to finding the equilibrium distribution, our benchmark equation is simple enough that we are also able to use the Fokker–Planck equation (2.13) to find closed form expressions for the evolution of the moments of the distribution of independent solutions to (2.10) over time. In particular, consider the evolution of the

first and second moments of the position distribution, defining

$$\mu^{(1)}(t) := \int_{-\infty}^{\infty} xp \, dx \quad \text{and} \quad \mu^{(2)}(t) := \int_{-\infty}^{\infty} x^2 p \, dx.$$

We can now use the Fokker–Planck equation (2.13) to derive equations for these moments. For example, we have

$$\frac{d\mu^{(1)}}{dt} = \int_{-\infty}^{\infty} x \partial_t p \, dx = \int_{-\infty}^{\infty} x \partial_x \left(xp + \partial_x \left(\frac{1}{2}(1 + \eta x)^2 p \right) \right) dx.$$

Formally integrating by parts, we find that the first moment satisfies the ODE

$$\frac{d\mu^{(1)}}{dt} = - \int_{-\infty}^{\infty} xp \, dx - \int_{-\infty}^{\infty} \partial_x \left(\frac{1}{2}(1 + \eta x)^2 p \right) dx = -\mu^{(1)},$$

and a similar argument allows us to show that the second moment satisfies

$$\frac{d\mu^{(2)}}{dt} = -(2 - \eta^2)\mu^{(2)} + 2\eta\mu^{(1)} + 1.$$

If the initial condition is deterministic, so that $p(x, 0) = \delta_{x_0}(x)$, the corresponding initial conditions for these moment equations are $\mu^{(1)}(0) = x_0$ and $\mu^{(2)}(0) = x_0^2$. Solving the first equation, we find that

$$\mu^{(1)}(t) = x_0 e^{-t}. \tag{2.16}$$

As expected, $\mu^{(1)}(t) \rightarrow 0$ as $t \rightarrow \infty$, and the first moment tends to the first moment of the equilibrium distribution, as expected from the discussion of ergodicity given above.

For the second moment, we can substitute the expression for the first moment and

solve, yielding the solutions

$$\mu^{(2)}(t) = \begin{cases} x_0^2 e^{-(2-\eta^2)t} + \frac{1}{2-\eta^2} (1 - e^{-(2-\eta^2)t}) + \frac{2\eta x_0}{1-\eta^2} (e^{-t} - e^{-(2-\eta^2)t}), & \eta \neq \pm 1, \\ x_0^2 e^{-t} + 1 - e^{-t} \pm 2x_0 t e^{-t}, & \eta = \pm 1. \end{cases} \quad (2.17)$$

We observe that the long-time limit of $\mu^{(2)}$ exists only in the case when $\eta^2 < 2$, which corresponds to the case where the second moment of the equilibrium distribution is finite, and in this case $\mu^{(2)}(t) \rightarrow \frac{1}{2-\eta^2}$ as $t \rightarrow \infty$, which is exactly the value given in (2.15).

Even in the case where $\mu^{(2)}$ blows up as $t \rightarrow \infty$, we observe that we can still determine the value of η (up to sign) by observing the exponential growth rate:

$$\lim_{t \rightarrow \infty} \frac{\log \mu^{(2)}(t)}{t} = \eta^2 - 2.$$

This demonstrates that first and second moment information are sufficient to recover the η parameter which fixes the particular reduced problem (2.10).

2.5 Statistical stability properties of numerical schemes

We now turn our focus to the asymptotic stability properties of the numerical schemes we consider. To study the asymptotic properties of the numerical schemes considered across a range of time-step sizes h , we define the j th moment at step n of the numerical scheme to be

$$\mu_n^{(j)} := \mathbb{E}[(x_n)^j],$$

where x_n is the position predicted by one of the schemes described in Section 2.2 at time $t_n = nh$.

By considering each of the schemes in turn, we will show that we can derive recurrence relations for these moments. We will focus on the cases where $j = 1$ and $j = 2$, since as we have argued above, these are the moments which together can be used

to characterise the reduced SDE (2.10) via the method of moments. Moreover, in a general nonlinear setting, it is common to focus on wide-sense (or weakly) stationary processes; these are the class of processes which have time-invariant mean and time autocovariance functions, rather than being strictly stationary (in which case all finite time-marginals are time-invariant). Wide-sense stationary processes are often considered since the mean, variance and autocovariance are the most important and readily-measurable statistics for applications, and being wide-sense stationary is equivalent to being strictly stationary for Gaussian processes. The latter are a common model for many data-generating processes in practice; see for example the discussion in Section 4.18 of [37].

2.5.1 Asymptotic moment stability

For the purposes of our analysis, we define the following notions of asymptotic stability and accuracy of a numerical discretization as applied to a particle SDE problem.

For a given time step h , a numerical discretization scheme of an SDE is said to be *asymptotically stable for the j th moment* if

$$\limsup_{n \rightarrow \infty} \left| \mu_n^{(j)} - \mathbb{E}[(X_{t_n})^j] \right| < +\infty.$$

A numerical discretization is said to be *asymptotically $O(h^n)$ accurate for the j th moment* if for all h sufficiently small,

$$\limsup_{n \rightarrow \infty} \left| \mu_n^{(j)} - \mathbb{E}[(X_{t_n})^j] \right| = O(h^n).$$

Clearly, the latter notion of asymptotic accuracy for the j th moment necessarily requires that there exist some possible values of the time step h for which the scheme

is also asymptotically stable for the same moment.

Before proceeding further, we comment on the distinction between the definition of asymptotic moment stability given above and the existing condition of mean-squared (MS) stability, which was first defined in [27]. In fact, our definition is simply a generalisation of the notion of MS-stability: MS-stability was introduced on the basis of geometric Brownian motion as a benchmark, and corresponds to the requirement that $\mu_n^{(2)} \rightarrow 0$ as $n \rightarrow \infty$ for fixed step size h . The choice of this condition was motivated by the fact that it is convenient to understand a test problem in which all sample paths concentrate at 0 as $t \rightarrow \infty$, as they do in the case of the benchmark ordinary differential equation $dq_t = -q_t dt$. In the case of geometric Brownian motion as studied in [28], we have that $\lim_{n \rightarrow \infty} \mathbb{E}[(X_{t_n})^2] = 0$. In this case, under our definition, an integrator applied to the same equations with fixed timestep h is both stable and $o(1)$ accurate for second moments if and only if it is MS-stable.

By refining our understanding of stability and accuracy via the definition above, we can consider the discretization of SDEs where j th moments tend to a stable non-zero value as $t \rightarrow \infty$. As explained in the introduction, this is often a desirable property of SDEs used in applications, where some form of ergodicity is of interest. Moreover, as we have already seen in the discussion of our benchmark given in Section 2.4.2, there are examples of relatively simple SDEs in which $\mathbb{E}[(X_{t_n})^j]$ does not have a finite limiting value, despite the SDE being ergodic. Our definition also covers such cases, by requiring only that the limit supremum of the difference between moments computed from the integrator and from the true SDE remain under control.

Over the course of the following subsections, we derive conditions under which the schemes described in Section 2.2 are asymptotically stable for first and second moments when applied to the model problem (2.10). A corresponding analysis for the more standard GBM benchmark can be found in Appendix 2.3.

2.5.2 Euler-Maruyama (EM)

We begin by deriving a recurrence relation for the moments as they evolve under the EM scheme. Taking expectations and using the scheme definition (EM) in this case, we find that

$$\mu_{n+1}^{(1)} = \mathbb{E}[x_{n+1}] = \mathbb{E}[(1-h)x_n + (1+\eta x_n)\Delta W_n] = (1-h)\mu_n^{(1)},$$

where we have used the fact that x_n and $\Delta W_n \sim \mathcal{N}(0, h)$ are independent, and hence we can solve to obtain $\mu_n^{(1)} = (1-h)^n x_0$. For the expectation to vanish in the long-time limit, we therefore require that

$$|1-h| < 1, \quad \text{i.e.} \quad 0 < h < 2.$$

As we should expect, this is the classical stability region for the explicit Euler method. When the scheme is asymptotically first moment stable, we have that $\mu_n^{(1)} \rightarrow 0$ as $n \rightarrow \infty$, so there is no asymptotic bias in the scheme.

Following a similar approach for the second moment using independence and the properties of a standard normal random variable to deduce that the cross terms vanish, we have

$$\begin{aligned} \mu_{n+1}^{(2)} &= \mathbb{E}[(x_{n+1})^2] = \mathbb{E}[(1-h)^2 x_n^2] + \mathbb{E}[(1+\eta x_n)^2 (\Delta W_n)^2] \\ &= (1-h)^2 \mu_n^{(2)} + h(1+2\eta\mu_n^{(1)} + \eta^2 \mu_n^{(2)}) \\ &= \left((1-h)^2 + h\eta^2\right) \mu_n^{(2)} + 2h\eta\mu_n^{(1)} + h. \end{aligned}$$

Employing the standard linear stability analysis for discrete dynamical systems, the

fixed point is stable when

$$\left| \frac{\partial}{\partial \mu_n^{(2)}} \left[\left((1-h)^2 + h\eta^2 \right) \mu_n^{(2)} + 2h\eta\mu_n^{(1)} + h \right] \right| = |(1-h)^2 + h\eta^2| < 1;$$

this reduces to the requirement that

$$0 < h < 2 - \eta^2. \quad (2.18)$$

Note that this is (as expected) strictly more restrictive than the requirement that $0 < h < 2$ when only considering the asymptotic stability of the first moment.

If the scheme is asymptotically first and second moment stable under the conditions obtained above, then $\mu_n^{(2)} \rightarrow \mu_\infty^{(2)}$ which solves

$$\mu_\infty^{(2)} = \left((1-h)^2 + h\eta^2 \right) \mu_\infty^{(2)} + h \quad \text{and hence} \quad \mu_\infty^{(2)} = \frac{1}{2 - \eta^2 - h} = [X_{t_n}^2] + O(h).$$

The asymptotic second moment is therefore asymptotically first-order accurate for the second moment.

2.5.3 Milstein (Mil)

For the Milstein scheme applied to test equation (2.10), the relevant coefficients are

$$f_n = -x_n, \quad g_n = 1 + \eta x_n \quad \text{and} \quad \frac{1}{2}[g'g]_n = \frac{1}{2}\eta(1 + \eta x_n),$$

so

$$x_{n+1} = (1-h)x_n + (1+\eta x_n)\Delta W_n + \frac{1}{2}\eta(1+\eta x_n)(\Delta W_n^2 - h).$$

Taking expectations, the latter terms on the right-hand side vanish, and as with the Euler-Maruyama scheme, we have

$$\mu_{n+1}^{(1)} = (1-h)\mu_n^{(1)}.$$

The same step-size restriction therefore applies for asymptotic stability of the first moment, i.e. we require $0 < h < 2$. Similarly, when asymptotically first moment stable, there is no error in the scheme; the limit point is $\mu_\infty^{(1)} = 0$.

To analyze the second moment, we find that after squaring, the expectations of the cross terms again vanish, and we obtain

$$\begin{aligned}\mu_{n+1}^{(2)} &= (1-h)^2 \mathbb{E}[x_n^2] + \mathbb{E}[(1+\eta x_n)^2][\Delta W_n^2] + \frac{1}{4}\eta^2 \mathbb{E}[(1+\eta x_n)^2] \mathbb{E}[(\Delta W_n^2 - h)^2] \\ &= (1-h)^2 \mu_n^{(2)} + h \left(1 + 2\eta \mu_n^{(1)} + \eta^2 \mu_n^{(2)}\right) + \frac{1}{2}h^2 \eta^2 \left(1 + 2\eta \mu_n^{(1)} + \eta^2 \mu_n^{(2)}\right).\end{aligned}$$

Once more, assuming that $\mu_n^{(1)} \rightarrow 0$, the system has a stable fixed point under the condition that

$$\begin{aligned}\left| \frac{\partial}{\partial \mu_n^{(2)}} \left((1-h)^2 \mu_n^{(2)} + h \left(1 + 2\eta \mu_n^{(1)} + \eta^2 \mu_n^{(2)}\right) + \frac{1}{2}h^2 \eta^2 \left(1 + 2\eta \mu_n^{(1)} + \eta^2 \mu_n^{(2)}\right) \right) \right| \\ = \left| (1-h)^2 + h\eta^2 + \frac{1}{2}h^2 \eta^4 \right| = \left| 1 - (2-\eta^2)h + (1+\frac{1}{2}\eta^4)h^2 \right| < 1.\end{aligned}$$

For $h > 0$, this condition can be further reduced to

$$0 < h < \frac{2-\eta^2}{1+\frac{1}{2}\eta^4}. \quad (2.19)$$

Note that the latter condition for the stability of the second moment is more restrictive than for EM; compare (2.18). When the scheme is asymptotically second moment stable, the fixed point $\mu_\infty^{(2)}$ must solve

$$\mu_\infty^{(2)} = \left((1-h)^2 + h\eta^2 + \frac{1}{2}h^2 \eta^4 \right) \mu_\infty^{(2)} + h + \frac{1}{2}h^2 \eta^2,$$

which has solution

$$\mu_\infty^{(2)} = \frac{1 + \frac{1}{2}h\eta^2}{2 - \eta^2 - h(1 + \frac{1}{2}\eta^4)} = \frac{1}{2 - \eta^2} + O(h),$$

and hence the scheme is asymptotically first-order accurate for the second moment.

2.5.4 Stochastic Heun (SH)

To analyze the SH method, we note that the auxiliary function used in the scheme $F := f - \frac{1}{2}g'g$ is

$$F(x) = -x - \frac{1}{2}\eta(1 + \eta x).$$

Working through the algebra, and taking expectations, we find that

$$\mu_{n+1}^{(1)} = \left(1 - h + \frac{1}{8}h^2(2 + \eta^2)^2\right) \mu_n^{(1)} + \frac{1}{8}\eta h^2(2 + \eta^2).$$

In this case, the criterion for first moment stability is

$$\left|1 - h + \frac{1}{8}h^2(2 + \eta^2)^2\right| < 1, \quad \text{or} \quad 0 < h < \frac{8}{(2 + \eta^2)^2}.$$

When stable, the limiting fixed point $\mu_\infty^{(1)}$ is

$$\mu_\infty^{(1)} = \frac{\frac{1}{8}\eta h(2 + \eta^2)}{1 - \frac{1}{8}h(2 + \eta^2)^2} = O(h).$$

As such, the SH scheme is asymptotically biased, i.e. it has a first-order accurate asymptotic first moment.

For the second moment, working through the algebra, we obtain the recurrence relation:

$$\begin{aligned} \mu_{n+1}^{(2)} = & \left(h - \frac{1}{2}h^2(2 + \eta^2) + \frac{1}{4}h^3(1 + \eta^2)^2 + \frac{1}{64}h^4\eta^2(2 + \eta^2)^2 \right) \\ & + \left(2h\eta - \frac{1}{4}h^2\eta(10 + 3\eta^2) + \frac{1}{4}h^3\eta(2 + 5\eta^2 + 2\eta^4) + \frac{1}{32}h^4\eta(2 + \eta^2)^3 \right) \mu_n^{(1)} \\ & + \left(1 - h(2 - \eta^2) - \frac{1}{4}h^2((2 + \eta^2)^2 - 12) \right. \\ & \quad \left. + \frac{1}{4}h^3(\eta^6 + 3\eta^4 - 4) + \frac{1}{64}h^4(2 + \eta^2)^4 \right) \mu_n^{(2)}. \end{aligned}$$

This recurrence relation is stable if the first moment converges, and if in addition

$$\left| 1 - h(2 - \eta^2) - \frac{1}{4}h^2((2 + \eta^2)^2 - 12) + \frac{1}{4}h^3(\eta^6 + 3\eta^4 - 4) + \frac{1}{64}h^4(2 + \eta^2)^4 \right| < 1.$$

In this case, the stability region does not have simple closed form expression, and indeed we find that the SH method is in fact stable beyond the region in which the SDE has an asymptotically stable second moment; see Section 2.6 below for further discussion.

Using the asymptotic first moment obtained above and solving to find the fixed point, we obtain that the second moment is first-order accurate, i.e. that

$$\mu_\infty^{(2)} = \frac{1}{2 - \eta^2} + O(h).$$

2.5.5 3-stage Runge-Kutta

Here, after working through the algebra and taking expectations, we find that

$$\mu_{n+1}^{(1)} = -\frac{1}{24}h^2\eta(2 + 3\eta^2) - \frac{1}{48}h^3\eta(2 + \eta^2)^2 + \left(1 - h + \frac{1}{8}h^2(4 - \eta^4) - \frac{1}{48}h^3(2 + \eta^2)^3\right)\mu_n^{(1)}$$

The first moment is asymptotically stable whenever

$$\left| 1 - h + \frac{1}{8}h^2(4 - \eta^4) - \frac{1}{48}h^3(2 + \eta^2)^3 \right| < 1,$$

and the fixed point is

$$\mu_\infty^{(1)} = -\frac{h\eta(4 + 6\eta^2 + h(2 + \eta^2)^2)}{48 - 6h(4 - \eta^4) + h^2(2 + \eta^2)^3} = O(h).$$

We therefore see that, like the Heun method, the Runge-Kutta scheme is asymptotically biased, but is asymptotically first-order accurate for the first moment.

For the second moment, we have

$$\begin{aligned}
\mu_{n+1}^{(2)} = & \left(h - \frac{1}{2}h^2(2 - \eta^2) + \frac{1}{12}h^3(8 - \eta^4) - \frac{1}{192}h^4(32 + 20\eta^2 - 44\eta^4 - 27\eta^6) \right. \\
& + \frac{1}{288}h^5(2 + \eta^2)^2(2 + 7\eta^2 + 6\eta^4) + \frac{1}{2304}h^6\eta^2(2 + \eta^2)^4 \Big) \\
& + \left(2h\eta - \frac{1}{12}h^2\eta(38 - 9\eta^2) + \frac{1}{24}h^3\eta(56 + 2\eta^2 - 5\eta^4) \right. \\
& - \frac{1}{96}h^4\eta(72 + 44\eta^2 - 50\eta^4 - 27\eta^6) \\
& \left. + \frac{1}{72}h^5\eta(2 + \eta^2)^3(1 + 3\eta^2) + \frac{1}{1152}h^6\eta(2 + \eta^2)^5 \right) \mu_n^{(1)} \\
& + \left(1 - h(2 - \eta^2) + \frac{1}{4}h^2(8 - 8\eta^2 + \eta^4) - \frac{1}{24}h^3(32 - 36\eta^2 + 3\eta^6) \right. \\
& + \frac{1}{192}h^4(2 + \eta^2)^2(28 - 52\eta^2 + 27\eta^4) - \frac{1}{96}h^5(2 + \eta^2)^4(1 - 2\eta^2) \\
& \left. + \frac{1}{2304}h^6(2 + \eta^2)^6 \right) \mu_n^{(2)}.
\end{aligned}$$

The stability criterion in this case is

$$\begin{aligned}
& \left| 1 - h(2 - \eta^2) + \frac{1}{4}h^2(8 - 8\eta^2 + \eta^4) - \frac{1}{24}h^3(32 - 36\eta^2 + 3\eta^6) \right. \\
& \left. + \frac{1}{192}h^4(2 + \eta^2)^2(28 - 52\eta^2 + 27\eta^4) - \frac{1}{96}h^5(2 + \eta^2)^4(1 - 2\eta^2) + \frac{1}{2304}h^6(2 + \eta^2)^6 \right| < 1.
\end{aligned}$$

Again, there is no simple closed form for the solution in this case.

Under the assumption that the first moment is stable, it can be checked that the fixed point for this recurrence relation is

$$\mu_\infty^{(2)} = \frac{1}{2 - \eta^2} + O(h);$$

the full expression is omitted due to its length, but we see that this method is asymptotically first-order accurate for the second moment.

Table 2.2: A summary of the results of in Sections 2.5.2–2.5.5, i.e. the asymptotic accuracy results for the first and second moments as predicted by the methods studied.

Method	1st moment	2nd moment
Euler-Maruyama (EM)	Exact	$O(h)$
Milstein (Mil)	Exact	$O(h)$
Stochastic Heun (SH)	$O(h)$	$O(h)$
3-stage Runge-Kutta (RK3)	$O(h)$	$O(h)$

2.5.6 Summary of results

The asymptotic statistical accuracy results we have obtained over the course of the previous sections are summarized in Table 2.2, and a comparison of the stability regions for the first and second moments is provided in Figure 2.1.

Concerning the asymptotic accuracy results in Table 2.2, we recall that the results of [38, 2, 1] provide the strong and weak order of accuracy of various schemes over a fixed time interval $[0, T]$, and were summarized in Table 2.1. While one might assume *a priori* that schemes with higher weak order should give improved convergence of moments since these are expressed as expectations, we see that there are marked discrepancies between the sets of results. These gaps arise due to the time-dependent prefactor which is typically included in convergence estimates when considering a scheme over a fixed time interval $[0, T]$. As $T \rightarrow \infty$, this prefactor can blow up, meaning that the convergence estimates break down, and hence we do not necessarily achieve the same weak accuracy on asymptotic moments the the results on a finite time interval would predict. As such, we have shown the surprising result that selecting a higher-order scheme does not guarantee better approximation regarding the accuracy of asymptotic moments in the long-time simulation, at least when measured purely in terms of the time-step h , and indeed, higher-order schemes can in fact be less numerically stable in recovering these moments in long-term simulations.

Turning to the stability regions shown in Figure 2.1, we see that for first moments, EM and Mil have identical stability regions, both regions being larger than

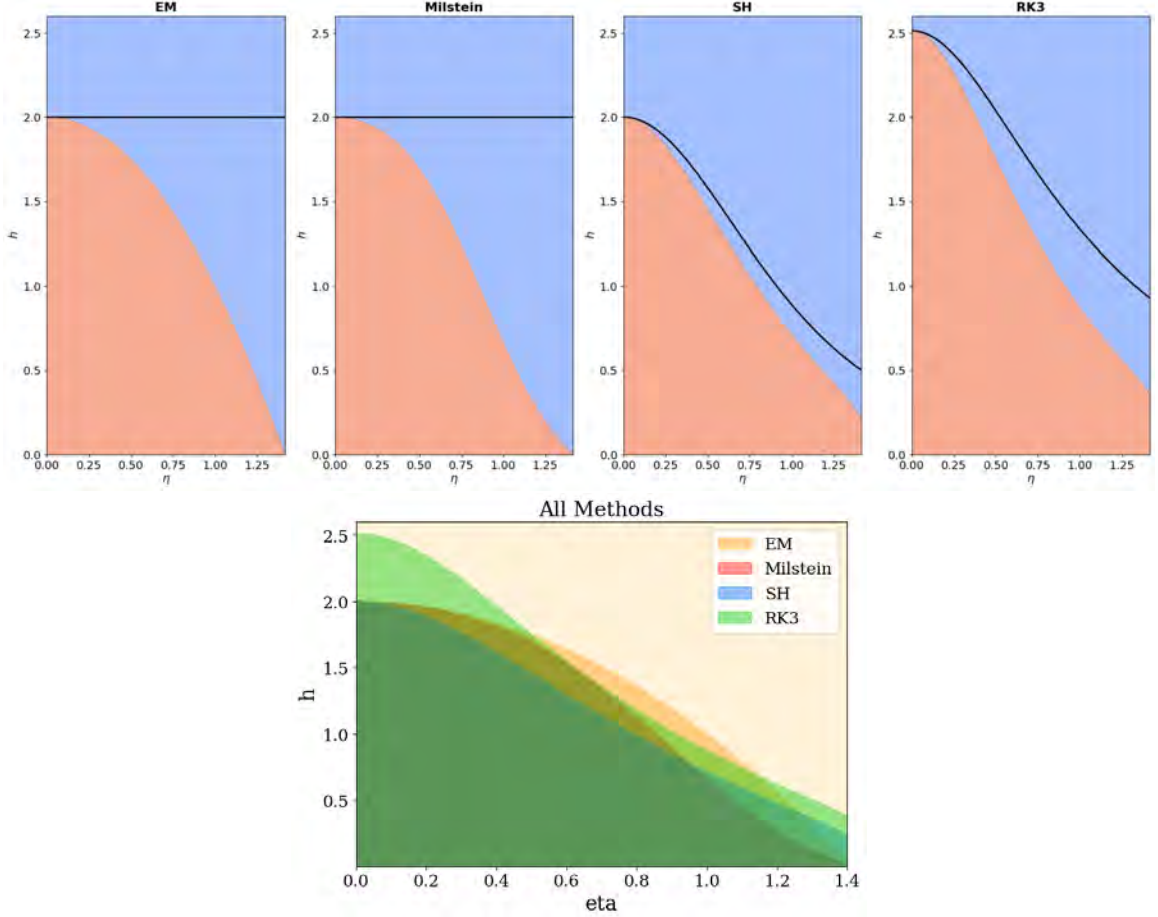


Figure 2.1: **Top row:** 1st moment stability regions (boundaries shown as lines) and 2nd moment stability regions (pink shaded regions) for the schemes considered; in each case, the stability region for second moments is strictly contained inside the stability region for first moments. **Bottom row:** A comparison of 2nd moment stability regions among all schemes plotted on the same axes. For asymptotic first moments, we observe EM and Mil are more stable than the higher-order schemes in many cases. For asymptotic second moments, RK3 is the most stable scheme when $\eta \leq 0.5$ or $\eta > 1.145$; and EM is the most stable when $0.5 < \eta < 1.145$.

SH, whereas RK3 is more stable than EM and Mil for $\eta \lesssim 0.5245$, and less stable if $\eta \gtrsim 0.5245$. This starkly illustrates the fact that when η is reasonably large, the lower-order schemes recover bounded first moments over a much wider range of time-steps h , while the higher-order schemes become unstable in long time simulations. As such, we see there is not always a stability gain to be made from using a higher-order method, in agreement with our analysis.

As our analysis predicts, we see that the regions in parameter space over which second moments are stable lie inside the first moment regions, and these regions coincide as $\eta \rightarrow 0$, which is the case where the diffusion coefficient is no longer position dependent. For 2nd moments, we observe that EM is stable over a strictly larger region than Mil, and RK3 is stable over a strictly larger region than SH. When η is very small or very large, that is $\eta \leq 0.5$ or $\eta \geq 1.145$, RK3 is more stable than EM, however, for $0.5 < \eta < 1.145$, EM is better. For second moments, we only see significant gains in the range of stable step sizes when η approaches $\sqrt{2}$, but we note that the bias observed in the first moments for the higher-order schemes would lead to poor recovery of the variance even in this case.

Since the geometric Brownian motion (GBM) benchmark is more standard in the analysis of SDE schemes than our benchmark (2.10) and was used in the original definition of the notion of MS-stability [27], in Appendix 2.3, we present a parallel analysis for the application of the schemes to our rescaled form of GBM (2.11). In terms of the timestep h and dimensionless parameter η present in this equation, we find that the stability regions take the same form, and so Figure 2.1 also illustrates these regions. We note however that a stark difference between the analysis of schemes applied to our proposed benchmark (2.10) and GBM is that all schemes are asymptotically exact in terms of the first and second moment for GBM. This highlights a special feature of the GBM equation, which is that the drift and diffusion both vanish at the same point, and this means that, when it exists, the equilibrium

measure is concentrated at $x = 0$. As argued in our derivation in Section 2.4.1, this behavior is rather special when considering a general SDE with affine coefficients, and is not always typical in applications. Moreover, it hides the variability in long-term performance; this will become clearer when we consider a nonlinear example equation in Section 2.6.3.

2.6 Numerical verification

To verify and illustrate our analytical results, we performed a range of numerical tests, both on the benchmark example equation, and on a nonlinear equation. These results are presented over the course of this section.

2.6.1 Asymptotic accuracy of moments for methods considered

To verify the asymptotic accuracy results summarized in Table 2.2, we computed the long-time errors of the first and second moments obtained by various numerical methods with respect to the step size h , and compared against the asymptotic moments of the equilibrium distribution: these results are shown in Figure 2.2.

Recall that the reference exact asymptotic moments are given by

$$\mu_{\infty}^{(1)} = 0 \quad \text{and} \quad \mu_{\infty}^{(2)} = \frac{1}{2 - \eta^2},$$

so we set the final simulation time to $T = 1000$ and considered two different parameter values for the dimensionless parameter η where we expect different types of behavior. The values chosen were $\eta = 0.1$ and $\eta = 1.412$, respectively. We see that errors decrease consistently and approximately linearly in most cases as the time step h decreases, following the estimates summarized in Table 2.2. For the small $\eta = 0.1$, we note that the SH method appears to exhibit superlinear convergence rate for the second moment; we hypothesise that this may be due to improved pre-asymptotic performance of this scheme in the case where the diffusion term is close to being position independent. In contrast, for the larger value $\eta = 1.412$, the EM scheme

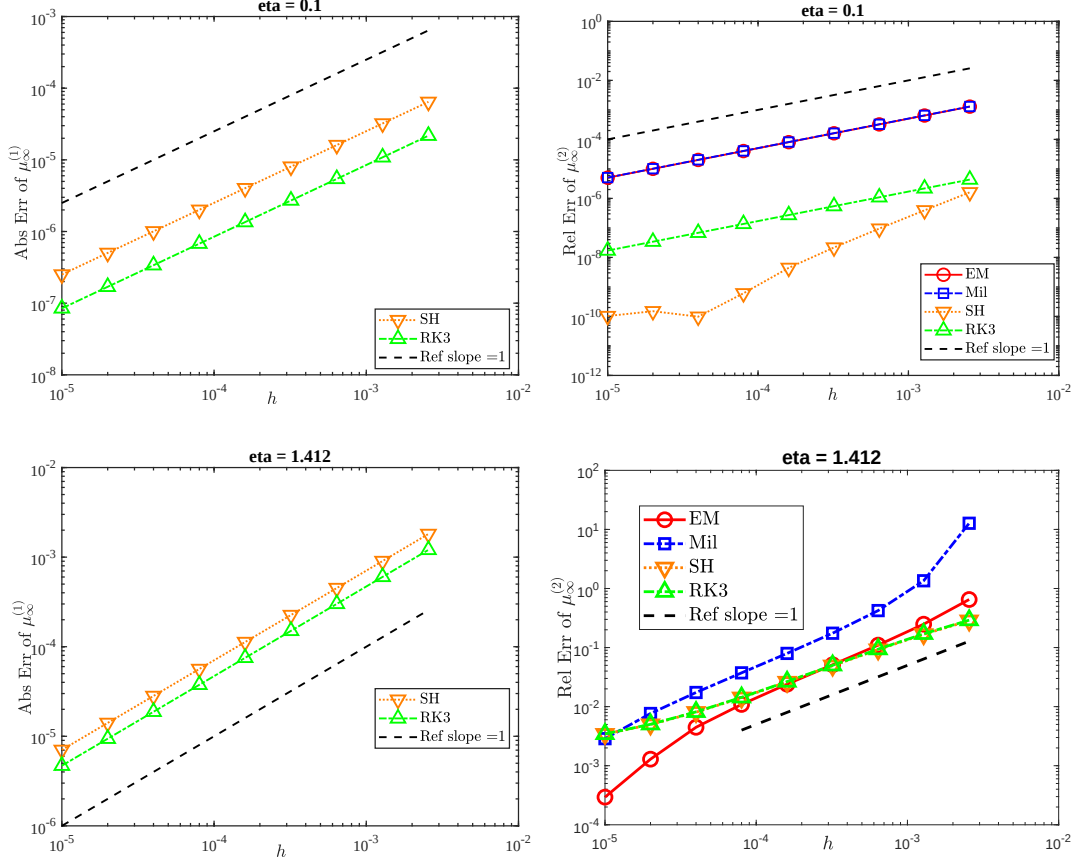


Figure 2.2: Asymptotic accuracy of the first and second moments obtained using the various numerical methods with respect to the step size h . Errors were computed as the absolute value of the difference between the exact moment in the limit $T \rightarrow \infty$ and the numerical moment at final simulation time $T = 1000$. **Column 1:** The absolute error in the first moment for the SH and RK3 schemes only, showing the $O(h)$ convergence predicted; EM and Mil are not shown as they exactly recover the first moment. **Column 2:** The relative error of the asymptotic second moment for each scheme. For the case where $\eta = 0.1$ (first row), the higher-order schemes perform better, and SH reaches machine precision for small h , with the accumulation of floating point errors appearing to limit further gains. For the case of larger η (second row), all schemes give similar error, with EM outperforming the higher-order schemes as h gets closer to 0.

is very competitive with the higher-order schemes in recovering the second moment, but all schemes show similar performance at moderate time step sizes.

2.6.2 Pre-asymptotic moment evolution

To investigate the evolution of error in the regime as the final time T increases, in Figures 2.3 and 2.4, we compare the analytic first and second moments computed using (2.16) and (2.17) with the corresponding discrete moments over time up to $T = 20$. We note that due to our nondimensionalisation leading up to (2.10), $T = 1$ corresponds to a typical relaxation time-scale in the evolution of the equation.

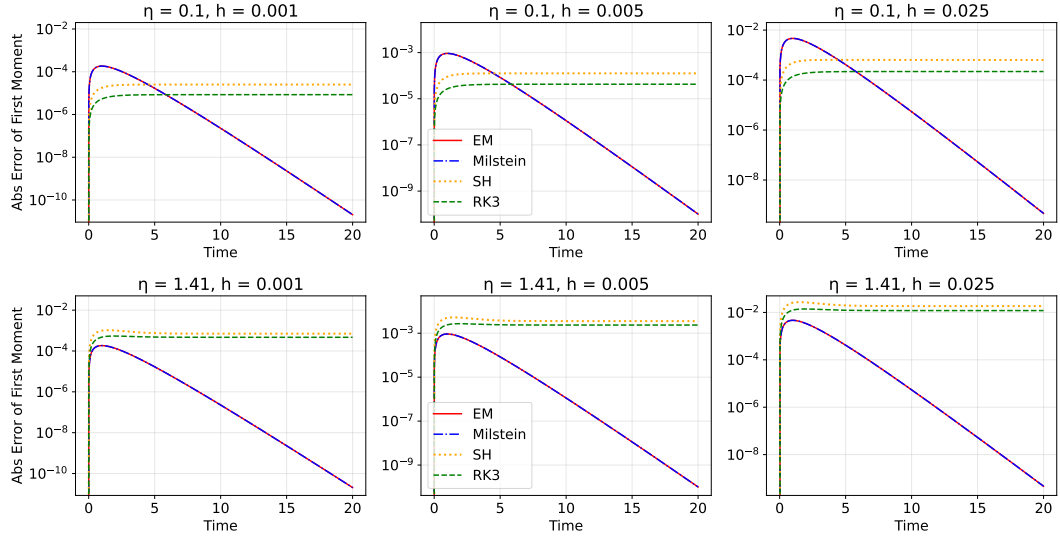


Figure 2.3: A comparison of the evolution of absolute first moment error for the various discretization schemes up to time $T = 20$ for our benchmark equation. The first row depicts the case when $\eta = 0.1$ for step sizes $h = 0.001, 0.005, 0.025$, and the second row depicts when $\eta = 1.41$ with the same range of step sizes. In agreement with our analytical results, we see that the EM and Mil recover asymptotically exact first moments, whereas RK3 and SH are asymptotically biased.

The numerical results agree with our rigorous analysis: in Figure 2.3, we see that for $\eta = 0.1$ and $\eta = 1.41$, the EM and Mil schemes are asymptotically stable and accurate for the first moment, as their absolute errors decay exponentially as T increases for the values of η chosen and a range of time step sizes h . Meanwhile, the RK3 and SH schemes exhibit asymptotic bias, with their discrete first moments stabilising at

a value which deviates from the analytical values at late times. In Figure 2.4, we see similar agreement: second moments tend to approach the correct values with errors behaving linearly in the time step size h .

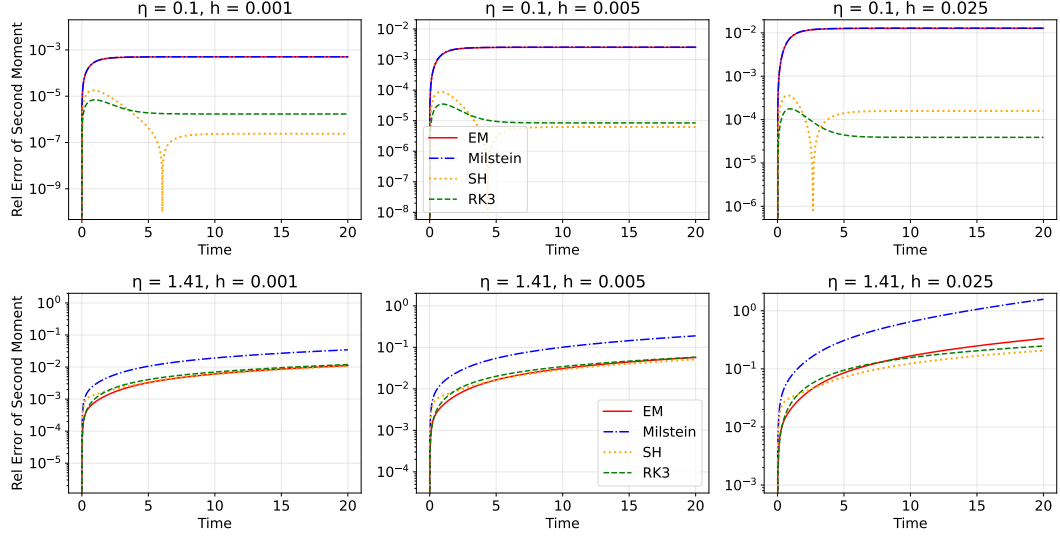


Figure 2.4: A comparison of the evolution of the relative error in second moments by the discretization schemes versus the exact evolution up to time $T = 20$. The first row depicts the case when $\eta = 0.1$ for step sizes $h = 0.001, 0.005, 0.025$, while the second row depicts the case when $\eta = 1.41$ with the same range of step sizes. At late times, we see substantial accuracy gains from higher-order schemes only when η is small.

We note that focusing on time $T = 1$, the higher-order schemes tend to perform better or at least comparably with the low-order schemes; this is to be expected from the existing weak-order results on fixed time intervals. However, at late times where $T \geq 10$, we see significant gains in accuracy from using a higher-order scheme only for second moments and when η is small. In agreement with our analytic results, these experiments therefore allow us to see that improved accuracy over fixed time intervals does not necessary translate to better accuracy in asymptotic statistics, and so the results in Table 2.2 do not contradict the weak order results summarized in Table 2.1.

2.6.3 Numerical comparison in a nonlinear case

As a further illustration of the utility of the benchmark above, we also investigated a numerical example for a one-dimensional SDE with coefficients which are smooth and globally Lipschitz, but not affine, in the general form (2.2). One application area in which advection-diffusion equations for particle transport of this form arise naturally as a homogenised limit are in the study of flows through porous media where the statistical properties medium varies in space [39, 40]. In particular, variation in the size of pores through which fluid can move leads to coupled spatial variation in the flow velocity and molecular diffusion of the particles in the flow. At the single particle scale, this yields variation in the drift and diffusion coefficients, $f(x)$ and $g(x)$. Study of long-time statistics in this case is important for accumulation of pollutants in soils where groundwater is being pumped out of the water table.

The particular case we consider is a stochastic diffusion process on an infinite domain, governed by the SDE

$$dq_t = -A \operatorname{erf}\left(\frac{q_t - x^*}{B}\right) dt + \left(C + D \operatorname{erf}\left(\frac{q_t - x^*}{E}\right)\right) dW_t, \quad (2.20)$$

where we have chosen a drift $f(x) = -A \operatorname{erf}\left(\frac{x - x^*}{B}\right)$ and diffusion $g(x) = C + D \operatorname{erf}\left(\frac{x - x^*}{E}\right)$. The coefficient functions exhibit sublinear growth and are globally Lipschitz, so strong solutions to the SDE exist.

Under the conditions that $A, B > 0$ and $D > C > 0$, it can be shown that this equation is ergodic with respect to an equilibrium distribution with exponentially decaying tails. Moreover, we can Taylor expand the coefficient functions about $x = x^*$ to obtain

$$\begin{aligned} f(x) &= f(x^*) + f'(x^*)(x - x^*) + O(|x - x^*|^2) = \frac{-2A}{B\sqrt{\pi}}(x - x^*) + O(|x - x^*|^2) \quad \text{and} \\ g(x) &= g(x^*) + g'(x^*)(x - x^*) + O(|x - x^*|^2) = C + \frac{2D}{E\sqrt{\pi}}(x - x^*) + O(|x - x^*|^2) \end{aligned}$$

Neglecting terms at quadratic and higher orders, we see that the corresponding linearised benchmark has

$$\eta = \frac{g'(x^*)}{\sqrt{|f'(x^*)|}} = \sqrt{\frac{2BD^2}{AE^2\sqrt{\pi}}}.$$

Note that x^* can be adjusted in the definition of $f(x)$ and $g(x)$ to ensure that the variance of q_t does not overwhelm its mean, thereby reducing sampling bias in Monte Carlo simulations.

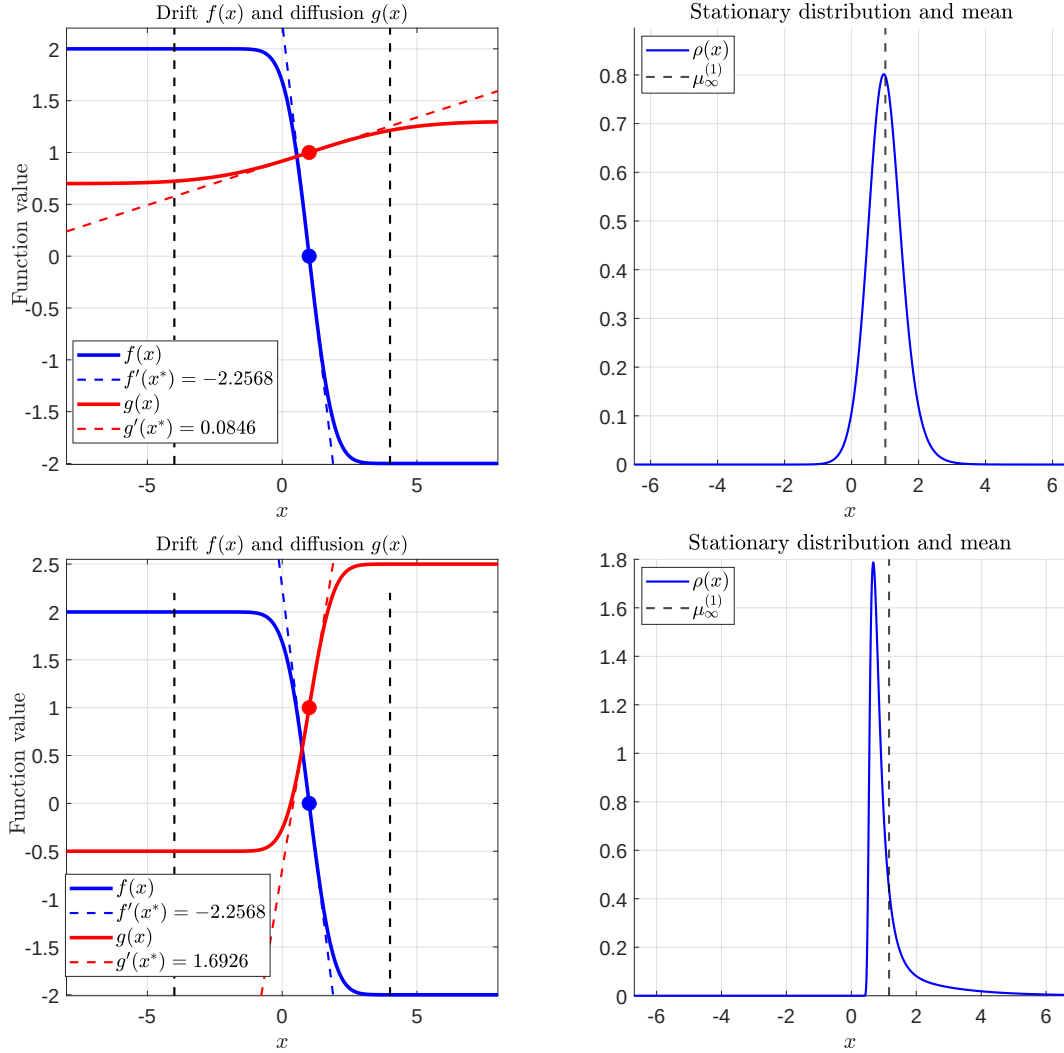


Figure 2.5: Examples of the particular drift and diffusion coefficients used in the non-linear example (2.20) (left column), along with the equilibrium measures computed by numerical integration (right column). The top row shows the case where $\eta \approx 0.05633$; while the bottom row shows the case where $\eta \approx 1.1267$. We see from the equilibrium measures that larger values of η leads to significant skew in the distribution.

For the purposes of numerical testing, we considered two sets of parameters, yielding different values of η :

- $A = 2, B = 1, C = 1, D = 1.5, E = 1$, and $x^* = 1$, yielding the case of large $\eta \approx 1.1267$; and
- $A = 2, B = 1, C = 1, D = 0.3, E = 4$, and $x^* = 1$, yielding the case of small $\eta \approx 0.05633$.

The particular functions f and g in these cases are plotted in Figure 2.5, along with the corresponding equilibrium distributions computed numerically using the formula

$$p_\infty(x) = \frac{1}{Zg(x)^2} \exp\left(\int_{-\infty}^x \frac{2f(y)}{g(y)^2} dy\right) \quad \text{where} \quad Z := \int_{-\infty}^{\infty} \frac{1}{g(x)^2} \exp\left(\int_{-\infty}^x \frac{2f(y)}{g(y)^2} dy\right) dx. \quad (2.21)$$

The density was computed using versions of the trapezoid rule over a uniform grid, using the functionality provided by the `scipy.integrate` package [41].

To investigate whether our late-time asymptotic accuracy and stability results for the benchmark continue to be relevant for this nonlinear example, we simulated 4×10^4 realizations of the dynamics of (2.20) using the various schemes described, and computed a sample estimate of the first moment at time $T = 16$ for a range of step sizes $h \in 0.01 \times 2^{[1:6]} = \{0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64\}$. As a reference value for comparison, we computed the first mean of the equilibrium distribution using a numerical discretization of the exact expression (2.21) using the trapezoid rule. To do so, the equilibrium distribution (2.21) and mean were computed and normalized on the interval $[-100, 100]$ using a uniform grid with step size $\Delta x = 0.0025$. Refinement of the grid allowed us to show this was sufficient to guarantee 5 digit accuracy in the reference values of asymptotic mean. The distance between the sample mean and the reference value for the equilibrium mean obtained by numerical integration is plotted in Figure 2.6.

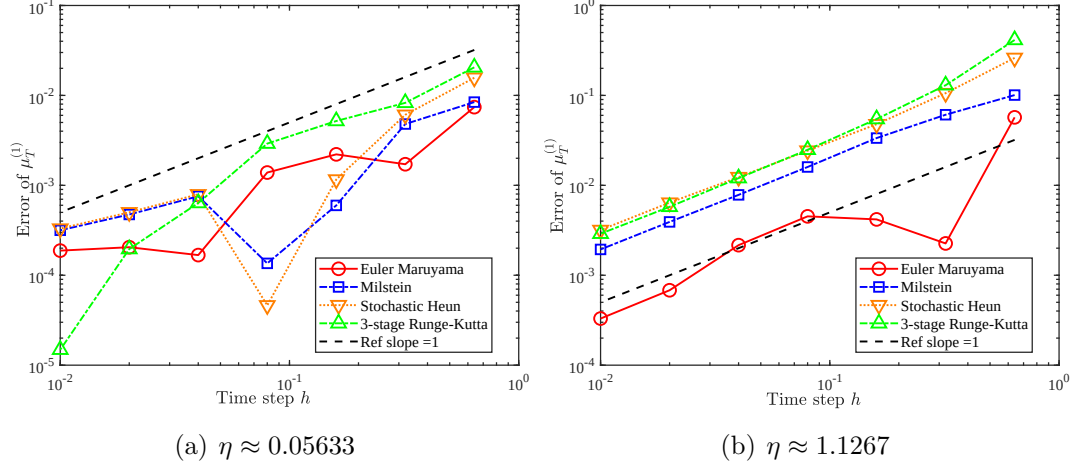


Figure 2.6: Numerical errors in the asymptotic means for discretizations of the SDE with non-affine coefficients (2.20), plotted against time step size. The difference between the sample mean at time $T = 16$ taken over 4×10^4 trajectories against the equilibrium mean determined by numerical integration is plotted. We see that the accuracy decreases linearly with step size, as predicted by our analytic results for the benchmark equation.

We observe first-order convergence rates with respect to the time step size h for all schemes considered. In addition, the EM scheme shows the smallest errors for relatively large time step sizes h in both test cases. This result is consistent with our asymptotic moment accuracy analysis, which indicates that the asymptotic first moments are $O(h)$ -accurate for all schemes, even in this nonlinear setting. We focused on first moments, since the sample error for the statistical estimator of the second moment is controlled by the fourth moment of equilibrium distribution, which is very large in these examples, and eliminating this source of error to make a meaningful comparison would have required excessive computational work.

To assess the evolution of the first moment over time in this example, we compared the mean paths produced by various schemes for (2.20) using both a fine step size $h = 0.01$ and a coarse step size $h = 1$; the results of this comparison are shown in Figure 2.7.

For the case where a small value of η is predicted, we see stable recovery of the mean, which is consistent with the point where $\eta \approx 0.05633$ and $h = 1$ being far

from the boundary of the first-moment stability region for all schemes. The picture is somewhat different in the large η case. The lower-order schemes EM and Mil appear more stable, with less variation in the evolution of the first moment. In contrast, the higher-order scheme SH exhibits growing mean-path errors over time, and seems to be unstable. This mirrors the fact that the point where $h = 1$ and $\eta \approx 1.1267$ lies outside the first moment stability region shown in Figure 2.1. For the RK3 scheme, the same point is inside the first-moment region for RK3, but is much closer to the stability boundary than for the lower-order schemes, and as such, we see much larger variation in the first moment evolution. This demonstrates that our analysis of the benchmark equation (2.10) continues to be of relevance even in cases where the coefficients are nonlinear.

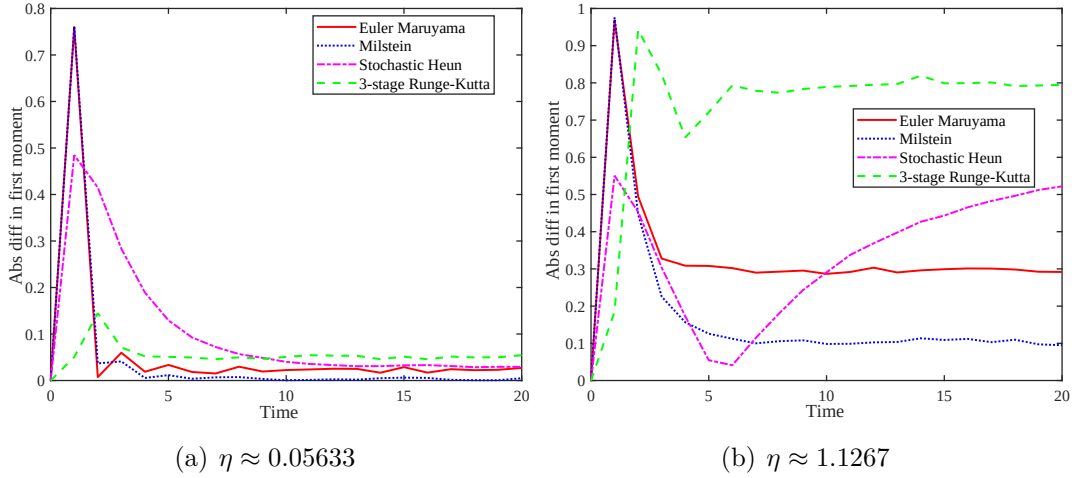


Figure 2.7: A comparison of the evolution of the first moment for discretizations of the nonlinear problem (2.20) for the sets of parameters producing small and large values of η . The plots compare the means predicts with a fine step size $h = 0.01$ and a coarse step size $h = 1$, averaged over 4×10^4 sample trajectories. We see that the low-order schemes appear to predict the first moment evolution more consistently across a wide variation of time steps, and there is evidence that SH is unstable for the large value of η and large time-step, consistent with our stability analysis on the simpler benchmark.

2.7 Conclusions

Motivated by the need by practitioners to capture meaningful statistical averages of solutions to SDEs at late times, we used a nondimensionalisation to derive a family of one-dimensional benchmark problems to evaluate the stability of numerical schemes for SDEs. We argued that in a generic case, the performance of numerical schemes on this benchmark provides more information about the long-time performance of these schemes for a range of applications than the more usual benchmark of geometric Brownian motion. We used our benchmark to present an analysis of the ability of four explicit numerical schemes used for simulating SDEs to recover first and second moments at late times both accurately and stably. We observed that lower-order schemes such as the Euler-Maruyama tend to preserve asymptotic statistical accuracy of both first and second moments better than higher-order schemes, a surprising trend which persists even when testing a realistic nonlinear benchmark problem. Natural next steps for this work would be to expand our analysis to include a range of implicit schemes, and to explore similar benchmarks for higher-order and higher-dimensional SDEs. In the context of this dissertation, this work provides a study on the numerical error introduced by classical time integrators. However, in practice, simply using a time integrator is insufficient for mathematical modeling frameworks, and often a reduction in the representation of a system is required. In the next chapter, we explore the need to also consider the modeling error that is introduced when a reduction in the state variable is made by coarse-graining.

CHAPTER 3: MODELING ERROR ANALYSIS OF COARSE-GRAINED STOCHASTIC SYSTEMS

In this chapter, we expand on the previous project with a study of the need to balance modeling error and computational cost as well as numerical stability. In some cases, applying classical numerical schemes like those in the previous chapter to a high-dimensional system is incredibly costly. This problem is particularly potent in molecular dynamics, where an atomistic model of a system has thousands of interacting degrees of freedom. Rather than adjusting the numerical scheme used, one approach to balance the computational cost is to reduce the full system to a surrogate model, approximate the solution of the surrogate model, and then extrapolate insights back to the full system. This process is known as model reduction, and modeling error quantifies the information lost when the reduction is made. Together with the previous chapter, we highlight the need to balance numerical stability, modeling error and efficiency when modeling materials. In later chapters, we transfer these insights from the classical setting to the modern machine learning setting, while also exploring the additional complexities of modern methods.

Mathematical models that can simulate and predict the behavior of stochastic dynamical systems at the microscopic scale are needed for a range of scientific problems, namely in molecular dynamics [42, 43, 44], climate science [45, 46], and materials science [4, 47]. Often, these systems of interest require the accurate simulation of the interaction of a large number of variables. Modeling the evolution of such systems at high resolutions is extremely costly and infeasible in practice. Coarse-grained models provide a means for reducing the dimensionality of such systems, resulting in approximate models that rely on the contributions of only the most important variables in

a system.

Recent work has shown the capability of coarse-graining for molecular simulations, and numerous papers highlight means for improving existing coarse-graining approaches. Coarse-grained models have been shown to extend the time and spatial scales for which complex systems can be simulated [48]. Data-driven schemes have been applied to improve the selection of coarse-grained variables and reduce overall error [49, 50]. Tools from optimal transport and information theory have also been shown to be used to analyze the performance of such methods [51, 52].

A current area of focus is to provide numerical guarantees on the performance of coarse-grained models. In this work, we study the application of a hierarchy of coarse-grained models in the overdamped linear Langevin setting. The models we study, presented in [53], are derived similarly to how the Generalized Langevin Equations are derived in the Mori-Zwanzig formalism [54]. In particular, the dynamics are reformulated in terms of the coarse-grained dynamics as an integro-differential equation. Then, the integral terms that arise are approximated by a mathematical truncation. The three approaches in this hierarchy are presented as follows: Approach 0 completely omits the integral terms, Approach 1 keeps one term from the Taylor expansions, and Approach 2 keeps 2 terms from the Taylor expansions. The full details of this procedure are presented in Section 3.1.

The primary error quantity used to evaluate these models was the stationary short-time statistics, the equilibrium autocovariance function (ACF), for which the authors established error bounds depending on spectral gaps and the alignment of coarse-graining directions. In particular, this analysis suggested that Approach 1 performs well at short time lags, but underestimates the ACF as time evolves. And, Approach 2 represents the ACF well in the long run, but can overestimate its value at short times. In particular, this was shown to depend on the spectral gap, the ACF error decreases as the spectral gap increases. This is in line with past findings that coarse-grained

models are particularly useful in problems with significant scale separation.

While the equilibrium ACF provides insights into the models behavior when the dynamics reach stationary status, it does not capture the full transition probability law of the dynamics. The transition probability density function (transition PDF) captures the conditional distribution of future states given present states, which is essential for generating predictions [55]. In this work, we extend the analysis of [53] by introducing transition PDF error analysis between coarse-grained models and full dynamics. The ACF error analysis from the reference paper measures how well the coarse-grained models capture correlations between present and future states, however, it does not capture the full transition properties of the dynamics. The transition PDF allows us to compare the full probability law between models. Further, we can utilize established error functions for probability distributions such as the 2-Wasserstein distance[56], the Kullback-Leibler divergence [57], and the total variation distance. These functions pull from various mathematical disciplines, namely optimal transport, information theory, and probability theory respectively, and have clear interpretations. While these metrics have been used in numerical analysis of coarse-graining methods before [51, 52], our work highlights not only the efficiency of our proposed methods, but also that transition PDF analysis via the error functions presented is a viable tool for assessing coarse-graining methods by short-time probability laws.

In this analysis, we make use of the fact that, in the overdamped linear Langevin setting, the full and coarse-grained models are Gaussian processes, which allows for closed form expressions for the 2-Wasserstein distance and Kullback-Leibler divergence between full and reduced models. The 2-Wasserstein distance pulls from tools of optimal transport, and has been used to describe the work required to transform one probability distribution into another [56]. This distance provides a clear depiction of the geometric error in the state space. The Kullback-Leibler divergence is an in-

formation theoretic tool that measures the relative entropy between two distributions [57]. An interpretation of this is that we can measure the information lost by the approximate model from the full model. The total variation distance measures the worst-case discrepancy between the probability laws, which is helpful for bounding the performance of coarse-grained models.

3.1 Coarse-graining of linear Langevin SDEs

Now we present the coarse-graining approaches that are explored in Chapter 3, originally presented in [53]. In this setting, we consider the linear Langevin SDE given by

$$d\mathbf{q}_t = -\mathbf{A}\mathbf{q}_t dt + \sqrt{2\beta^{-1}} d\mathbf{w}_t \quad (3.1)$$

where $\mathbf{A}$ is an $N \times N$ symmetric, positive definite matrix, and $\mathbf{q}$ is a vector in $\mathbb{R}^N$. The inverse temperature is given by $\beta = \frac{1}{k_B T}$ where k_B is the Boltzmann constant. The assumptions on $\mathbf{A}$ ensure that the system is ergodic with respect to the invariant measure corresponding to the Boltzmann-Gibbs equilibrium distribution. This system has the following analytical solution

$$\mathbf{q}_t = e^{-\mathbf{A}t} \mathbf{q}_0 + \int_0^t e^{\mathbf{A}(s-t)} \sqrt{2\beta^{-1}} d\mathbf{w}_s \quad (3.2)$$

and has a mean-zero Gaussian distribution at equilibrium $\mathcal{N}(\mathbf{0}, \beta^{-1} \mathbf{A}^{-1})$, which has the form

$$\mu(d\mathbf{q}) = \frac{1}{Z} \exp\left(-\frac{\beta}{2} \mathbf{q} \cdot (\mathbf{A}\mathbf{q})\right) d\mathbf{q} \quad (3.3)$$

where Z is a normalization constant. This setting is simple yet provides sufficient groundwork to analyze and provide theoretical framework for a hierarchy of coarse-grained models.

We assume 3.1 represents some N -dimensional system of interest to be simplified for predictive modeling. In particular, we want to reduce the degrees of freedom to $n < N$ by selecting n variables of interest, referred to as the resolved variables. We assume we have a linear map $\tilde{\Phi} \in \mathcal{L}(\mathbb{R}^N, \mathbb{R}^n)$ that selects the variables to be included in the coarse-grained model. We can view $\tilde{\Phi}$ as an $n \times N$ matrix. We assume this matrix is full rank. The resolved variables are denoted by $\tilde{\xi} = \tilde{\Phi}\mathbf{q}$.

We will utilize an orthogonal projection to decompose the full dynamics space into a direct sum of the resolve dynamics and unresolved dynamics. To this end, we define

$$\Phi = (\tilde{\Phi}\tilde{\Phi}^*)^{-\frac{1}{2}}\tilde{\Phi}, \quad \xi = \Phi\mathbf{q} = (\tilde{\Phi}\tilde{\Phi}^*)^{-\frac{1}{2}}\tilde{\xi} \quad (3.4)$$

and our projection is given by

$$P = \Phi^*\Phi = \tilde{\Phi}^*(\tilde{\Phi}\tilde{\Phi}^*)^{-1}\tilde{\Phi}. \quad (3.5)$$

It is straightfoward to show that P is an orthogonal projection onto the range of Φ^* , which is a subset of $\mathbb{R}^N$. As such, we assume $\mathbf{I} = P + \Psi^*\Psi$, where $\Psi \in \mathcal{L}(\mathbb{R}^N, \mathbb{R}^m)$ can be viewed as a selector map for the unresolved variables. And so, our resolved variables are $\xi = \Phi\mathbf{q}$, our unresolved variables $\zeta = \Psi\mathbf{q}$, and together we can write $\mathbf{q}_t = \Phi^*\xi_t + \Psi^*\zeta_t$.

And so, with $\mathbf{q}_t$ written in terms of the resolved and unresolved variables, we can rewrite the full dynamics in terms of ξ, ζ as well. The original SDE 3.1 can be written in block format as

$$d \begin{pmatrix} \xi_t \\ \zeta_t \end{pmatrix} = - \begin{pmatrix} \mathbf{A}_0\xi_t + \alpha\zeta_t \\ \alpha^*\xi_t + \mathbf{A}_1\zeta_t \end{pmatrix} dt + \sqrt{2\beta^{-1}} \begin{pmatrix} d\mathbf{w}_{0,t} \\ d\mathbf{w}_{1,t} \end{pmatrix} \quad (3.6)$$

where

$$\begin{aligned}\mathbf{A}_0 &= \Phi \mathbf{A} \Phi^* \in \mathbb{R}^{n \times n} & \mathbf{A}_1 &= \Psi \mathbf{A} \Psi^* \in \mathbb{R}^{m \times m} & \alpha &= \Phi \mathbf{A} \Psi^* \in \mathbb{R}^{n \times m} \\ \mathbf{w}_{0,t} &= \Phi \mathbf{w}_t \in \mathbb{R}^n & \mathbf{w}_{1,t} &= \Psi \mathbf{w}_t \in \mathbb{R}^m.\end{aligned}$$

Solving for the unresolved variables and then making a substitution allows us to write this SDE in terms of only the resolved variables as

$$d\xi_t = - \left(\mathbf{A}_0 \xi_t + \alpha e^{-\mathbf{A}_1 t} \zeta_0 - \int_0^t \alpha e^{\mathbf{A}_1(s-t)} \alpha^* \xi_s ds \right) dt \quad (3.7)$$

$$+ \left(\int_0^t \alpha e^{\mathbf{A}_1(s-t)} \sqrt{2\beta^{-1}} d\mathbf{w}_{1,s} \right) dt + \sqrt{2\beta^{-1}} d\mathbf{w}_{0,t}. \quad (3.8)$$

Note that 3.7 is equivalent to the original SDE 3.1, but the only information we need to know about the unresolved variables is the initial condition. However, this equation is nonlocal in time due to the integral equations that arise. The coarse-grained models are derived through approximating these integral terms to achieved an approximate model that is local in time.

3.1.1 Approach 0

This coarse-grained model approximates the integral terms by omitting them completely. Using the notation defined above, this is given by

$$d\xi_t = - \left(\mathbf{A}_0 \xi_t + \alpha e^{-\mathbf{A}_1 t} \zeta_0 \right) dt + \sqrt{2\beta^{-1}} d\mathbf{w}_{0,t} \quad (3.9)$$

3.1.2 Approach 1

Approach 1 approximates a portion of each integral term via a Taylor expansion. Define $\mathbf{B} := \mathbf{A}_0 - \alpha \mathbf{A}_1^{-1} \alpha^*$ and then Approach 1 is

$$d\xi_t = -(\mathbf{B}\xi_t - \alpha e^{-\mathbf{A}_1 t} \zeta_0)dt + \sqrt{2\beta^{-1}} d\mathbf{w}_{0,t} \quad (3.10)$$

3.1.3 Approach 2

Approach 2 is the most robust approximation, derived again through time rescaling

$$d\xi_t = -\mathbf{C}(\mathbf{B}\xi_t - \alpha e^{-\mathbf{A}_1 t} \zeta_0)dt + \sqrt{2\beta^{-1}}(\alpha \mathbf{A}_1^{-1} d\mathbf{w}_{1,t} + d\mathbf{w}_{0,t})$$

where the rescaling factor $\mathbf{C} = (\mathbf{I} + \alpha(\mathbf{A}_1)^{-2}\alpha^*)^{-1}$.

3.2 Statistical Properties

The key contribution of this work is to extend beyond the ACF error analysis provided in [53] to measure discrepancies in transition probability density functions between the full dynamics and coarse-grained models. In this section, we derive the transition probability density functions for the full dynamics and coarse-grained models. We then define utilize the 2-Wasserstein distance, the Kullback-Leibler Divergence, and the Total Variation distance to measure deviation between the transition PDFs of the true dynamics and coarse-grained models.

In the linear overdamped setting, the solution of the full dynamics is Gaussian. The coarse-grained models are Gaussian for the same reason. Suppose at initial time t_0 , we draw initial position $\mathbf{q}_0$ from the stationary distribution. The general form of the transition PDF for a Gaussian process is known and takes the following form

$$P(\mathbf{q}, t | \mathbf{q}_0, t_0) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma_t)}} \exp\left(-\frac{1}{2}(\mathbf{q} - \mu)^T \Sigma_t^{-1}(\mathbf{q} - \mu)\right). \quad (3.11)$$

And so, we need only know the mean μ and covariance Σ in order to calculate the transition PDF for the full dynamics.

3.2.0.1 Full dynamics transition PDF

The full dynamics are governed by the SDE

$$d\mathbf{q}_t = -\mathbf{A}\mathbf{q}_t dt + \sqrt{2\beta^{-1}} d\mathbf{W}_t \quad (3.12)$$

where we assume $\mathbf{A}$ is symmetric and strictly positive definite. The transition probability density function of the solution is Gaussian, and thus, uniquely determined by the transition mean and covariance of the solution. Because of linear SDE, thus to compute the transition PDF of the solution from time s to time t , one can compute the mean and covariance as follows:

$$\mathbb{E}[\mathbf{q}_t | \mathbf{q}_s] = e^{-\mathbf{A}(t-s)} \mathbf{q}_s \quad (3.13)$$

$$\text{Cov}[\mathbf{q}_t | \mathbf{q}_s] = \beta^{-1} \mathbf{A}^{-1} (\mathbf{I} - e^{-2\mathbf{A}(t-s)}) \quad (3.14)$$

Plugging these into the Gaussian transition PDF formula will give the transition PDF from s to t for the full dynamics. In order to capture only the contributions of ξ , we then project onto the resolved subspace

$$\mathbb{E}[\xi_t | \mathbf{q}_s] = \Phi e^{-\mathbf{A}(t-s)} \mathbf{q}_s \quad (3.15)$$

$$\text{Cov}[\xi_t | \mathbf{q}_s] = \Phi \beta^{-1} \mathbf{A}^{-1} (\mathbf{I} - e^{-2\mathbf{A}(t-s)}) \Phi^* \quad (3.16)$$

.

This is equivalent to using the Gaussian conditional formula and integrating out the effects of ζ . This is what will represent the short time behavior of the full dynamics, to be used as a comparison to the behavior of the coarse-grained models.

3.2.0.2 Approach 0 transition PDF

Approach 0 is given by

$$d\xi_t = -\mathbf{A}_0 \xi_t dt + \sqrt{2\beta^{-1}} d\mathbf{w}_{0,t} \quad (3.17)$$

The transition PDF for this SDE is Gaussian and thus determined by the transition mean and covariance from s to t . Since $\mathbf{A}$ is symmetric, so is $\mathbf{A}_0$, and thus can solve the integral in the covariance calculation exactly:

$$\mathbb{E}[\xi_t | \xi_s] = e^{-\mathbf{A}_0(t-s)} \xi_s = e^{-\mathbf{A}_0(t-s)} \Phi \mathbf{q}_s \quad (3.18)$$

$$\text{Cov}(\xi_t | \xi_s) = \beta^{-1} \mathbf{A}_0^{-1} (\mathbf{I} - e^{-2\mathbf{A}_0(t-s)}) \quad (3.19)$$

3.2.0.3 Approach 1 transition PDF

Approach 1 is given by

$$d\xi_t = -\mathbf{B} \xi_t dt + \sqrt{2\beta^{-1}} d\mathbf{w}_{0,t}, \quad \mathbf{B} = \mathbf{A}_0 - \alpha \mathbf{A}_1^{-1} \alpha^* \quad (3.20)$$

where $\mathbf{B}$ is symmetric because $\mathbf{A}$ is. The mean and covariance are given by

$$\mathbb{E}[\xi_t | \xi_s] = e^{-\mathbf{B}(t-s)} \xi_s = e^{-\mathbf{B}(t-s)} \Phi \mathbf{q}_s \quad (3.21)$$

$$\text{Cov}(\xi_t | \xi_s) = \beta^{-1} \mathbf{B}^{-1} (\mathbf{I} - e^{-2\mathbf{B}(t-s)}) \quad (3.22)$$

3.2.0.4 Approach 2 transition PDF

Approach 2 is given by

$$d\xi_t = -\mathbf{CB}\xi_t dt + \sqrt{2\beta^{-1}\mathbf{C}} d\mathbf{w}_{0,t}, \quad \mathbf{C} = (\mathbf{I} + \alpha\mathbf{A}_1^{-2}\alpha^*)^{-1}, \quad \mathbf{B} = \mathbf{A}_0 - \alpha\mathbf{A}_1^{-1}\alpha^* \quad (3.23)$$

and has transition mean from s to t given by

$$\mathbb{E}[\xi_t | \xi_s] = e^{-\mathbf{CB}(t-s)} \xi_s = e^{-\mathbf{CB}(t-s)} \Phi \mathbf{q}_s \quad (3.24)$$

Since $\mathbf{B}$ and $\mathbf{C}$ are symmetric, $(\mathbf{CB})^* = \mathbf{BC}$. Define

$$F(r) := e^{-\mathbf{CB}r} \mathbf{B}^{-1} e^{-\mathbf{BC}r}.$$

Since $\mathbf{CB}$ commutes with $e^{-\mathbf{CB}r}$,

$$F'(r) = -e^{-\mathbf{CB}r} (\mathbf{CB}\mathbf{B}^{-1} + \mathbf{B}^{-1}\mathbf{BC}) e^{-\mathbf{BC}r} = -2e^{-\mathbf{CB}r} \mathbf{C} e^{-\mathbf{BC}r}.$$

Hence, with $\tau = t - s$,

$$\begin{aligned} \text{Cov}(\xi_t | \xi_s) &= 2\beta^{-1} \int_0^\tau e^{-\mathbf{CB}r} \mathbf{C} e^{-\mathbf{BC}r} dr = \beta^{-1} (F(0) - F(\tau)) \\ &= \beta^{-1} (\mathbf{B}^{-1} - e^{-\mathbf{CB}\tau} \mathbf{B}^{-1} e^{-\mathbf{BC}\tau}). \end{aligned}$$

Finally, $(\mathbf{CB})^j \mathbf{B}^{-1} = \mathbf{B}^{-1} (\mathbf{BC})^j$ for all $j \geq 0$, so $e^{-\mathbf{CB}\tau} \mathbf{B}^{-1} = \mathbf{B}^{-1} e^{-\mathbf{BC}\tau}$, and therefore

$$\text{Cov}(\xi_t | \xi_s) = \beta^{-1} \mathbf{B}^{-1} (\mathbf{I} - e^{-2\mathbf{BC}\tau}),$$

which reduces to $\beta^{-1} \mathbf{B}^{-1} (1 - e^{-2\mathbf{CB}\tau})$ when $n = 1$.

3.2.1 Error Metrics

In this section we detail the error metrics used to compare transition PDFs of the full dynamics and coarse-grained models. Three metrics are chosen from vari-

ous contexts to provide robust error analysis. In the overdamped Langevin setting, the 2-Wasserstein distance and Kullback-Leibler Divergence have closed forms for Gaussians.

Throughout, the transition starts at time $s = 0$ from $\mathbf{q}_0 = \Phi^* \xi_0$ (so that $\zeta_0 = 0$, as set above), and $\tau = t - s = t > 0$ denotes the time lag. Denote the transition PDF of the full overdamped Langevin dynamics, projected onto the resolved variables, by $P_L(\tau) = \mathcal{N}(\mu_L(\tau), \Sigma_L(\tau))$, where

$$\mu_L(\tau) = \Phi e^{-\mathbf{A}\tau} \Phi^* \xi_0, \quad \Sigma_L(\tau) = \beta^{-1} \Phi \mathbf{A}^{-1} (\mathbf{I} - e^{-2\mathbf{A}\tau}) \Phi^*, \quad (3.25)$$

and denote the transition PDF of Approach $k \in \{0, 1, 2\}$ by $P_k(\tau) = \mathcal{N}(\mu_k(\tau), \Sigma_k(\tau))$, where, from the previous section,

$$\mu_0(\tau) = e^{-\mathbf{A}_0\tau} \xi_0, \quad \Sigma_0(\tau) = \beta^{-1} \mathbf{A}_0^{-1} (\mathbf{I} - e^{-2\mathbf{A}_0\tau}), \quad (3.26)$$

$$\mu_1(\tau) = e^{-\mathbf{B}\tau} \xi_0, \quad \Sigma_1(\tau) = \beta^{-1} \mathbf{B}^{-1} (\mathbf{I} - e^{-2\mathbf{B}\tau}), \quad (3.27)$$

$$\mu_2(\tau) = e^{-\mathbf{CB}\tau} \xi_0, \quad \Sigma_2(\tau) = \beta^{-1} \mathbf{B}^{-1} (\mathbf{I} - e^{-2\mathbf{BC}\tau}). \quad (3.28)$$

All metrics below are evaluated at $\tau > 0$; at $\tau = 0$ both laws are the point mass at ξ_0 .

3.2.1.1 2-Wasserstein Distance

The 2-Wasserstein distance has been applied to measure the distance between Gaussian distributions. In particular, the 2-Wasserstein distance for PDFs F, G with means μ_x, μ_y and covariance matrices Σ_x, Σ_y , is given by

$$d^2(F, G) = \|\mu_x - \mu_y\|^2 + \text{Tr}(\Sigma_x + \Sigma_y - 2(\Sigma_x^{1/2} \Sigma_y \Sigma_x^{1/2})^{1/2}) \quad (3.29)$$

where $\|\cdot\|$ is the Euclidean norm. The 2-Wasserstein distance between the transition

PDFs of the full dynamics and Approach k at lag τ is therefore

$$d^2(P_L(\tau), P_k(\tau)) = \|\mu_L(\tau) - \mu_k(\tau)\|^2 + \text{Tr} \left(\Sigma_L(\tau) + \Sigma_k(\tau) - 2 \left(\Sigma_L(\tau)^{1/2} \Sigma_k(\tau) \Sigma_L(\tau)^{1/2} \right)^{1/2} \right) \quad (3.30)$$

As $\tau \rightarrow \infty$, both means decay to zero, $\Sigma_L(\tau) \rightarrow \beta^{-1} \mathbf{B}^{-1}$ (since $\Phi \mathbf{A}^{-1} \Phi^* = \mathbf{B}^{-1}$), and $\Sigma_0(\tau) \rightarrow \beta^{-1} \mathbf{A}_0^{-1}$, so for Approach 0 the distance converges to the equilibrium value

$$d^2(P_L, P_0) = \text{Tr} \left(\beta^{-1} \mathbf{B}^{-1} + \beta^{-1} \mathbf{A}_0^{-1} - 2 \left(\beta^{-1} \mathbf{B}^{-1/2} \beta^{-1} \mathbf{A}_0^{-1} \mathbf{B}^{-1/2} \right)^{1/2} \right), \quad (3.31)$$

while for Approaches 1 and 2 it converges to zero, since $\Sigma_1(\tau), \Sigma_2(\tau) \rightarrow \beta^{-1} \mathbf{B}^{-1}$ as well.

3.2.1.2 Kullback-Leibler Divergence

The KL divergence utilizes information entropy to measure how far an approximating probability distribution is from a true probability distribution. We take the full dynamics as the reference. In our case, it is given by

$$D_{KL}(P_L(\tau) \| P_k(\tau)) = \int_{\mathbb{R}^n} P_L(x) \log \left(\frac{P_L(x)}{P_k(x)} \right) dx \quad (3.32)$$

The Kullback-Leibler divergence for two n -dimensional Gaussian distributions F, G with means μ_x, μ_y and covariance matrices Σ_x, Σ_y , is given by

$$D_{KL}(F \| G) = \frac{1}{2} \left(\ln \frac{\det \Sigma_y}{\det \Sigma_x} + \text{Tr}(\Sigma_y^{-1} \Sigma_x) + (\mu_y - \mu_x)^T \Sigma_y^{-1} (\mu_y - \mu_x) - n \right),$$

so that, suppressing the dependence on τ ,

$$D_{KL}(P_L \| P_k) = \frac{1}{2} \left(\ln \frac{\det \Sigma_k}{\det \Sigma_L} + \text{Tr}(\Sigma_k^{-1} \Sigma_L) + (\mu_k - \mu_L)^T \Sigma_k^{-1} (\mu_k - \mu_L) - n \right). \quad (3.33)$$

3.2.1.3 Total Variation

The total variation is also used to compare probability distributions. The formula to calculate total variation between distributions $P_L(\tau)$ and $P_k(\tau)$ is given by

$$\text{TV}(P_L(\tau), P_k(\tau)) = \frac{1}{2} \int_{\mathbb{R}^n} |P_L(x) - P_k(x)| dx \quad (3.34)$$

Unlike the previous two metrics, the total variation distance between Gaussians does not have a closed form in general, so it is computed numerically by the Trapezoid rule.

3.2.1.4 Non-equilibrium window

The resolved variables of the full dynamics have equilibrium law $\pi_\xi = \mathcal{N}(0, \beta^{-1} \mathbf{B}^{-1})$. To restrict attention to the regime before the full system has equilibrated, for a tolerance $\varepsilon > 0$ we define

$$\tau_{eq}(\varepsilon) = \inf\{\tau > 0 : d(P_L(\tau), \pi_\xi) \leq \varepsilon\}, \quad (3.35)$$

and report errors for $\tau \in (0, \tau_{eq}(\varepsilon)]$.

3.3 Cases of Analysis

Depending on the formulation of the transition PDF, we consider 2 applications of the transition PDFs of the full and reduced models. Numerical results based on these cases are in the following section. Given an initial time t_0 and timesteps $s < t$, we

could have the following scenarios:

- Transition PDFs from t_0 to t with initial condition at equilibrium (the stationary distribution of the full dynamics). In this context, the transition PDF captures long-time behavior of the system because marginal distributions stay at equilibrium. Also allows study of short-time dynamics, because the conditional evolution $p(q_t | X_{t_0})$ for small $t - t_0$ reflects how trajectories move locally around equilibrium. In this work, we use this case to show that transition PDF analysis can recover the ACF analysis presented in [53].
- Transition PDFs from t_0 to t with initial condition out of equilibrium (shifted mean, altered covariance, or arbitrary non-equilibrium state). In this context, the transition PDF captures short-time modeling error due to both drift and memory approximations in CG models. This allows investigation of how CG models reproduce relaxation toward equilibrium, which is our main contribution.

3.4 Numerical Experiments

In this section, we detail the numerical experiments performed to highlight the importance of capturing modeling error using transition probability laws.

3.4.1 2D example

For adequate comparison, we follow the formulation of the 2-dimensional example presented in [53]. The authors note that all 2-dimensional SDEs of the form 3.1 can be described using two parameters λ and θ . We choose a basis such that $\mathbf{A}$ is diagonal and rescale so that the lowest eigenvalue is 1. Then, we can assume that

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & \lambda \end{bmatrix} \quad (3.36)$$

with $\lambda \geq 1$. In general, any choice of coarse-graining selector map can be described as a rotation of this matrix by some angle $\theta \in (-\frac{\pi}{2}, \frac{\pi}{2})$ with

$$\Phi = (\cos \theta \sin \theta) \quad \Psi = (-\sin \theta \cos \theta).$$

With this formulation, λ describes the scale separation present in a system, and θ tunes how aligned the coarse-grained map is to the eigenspace corresponding to the slowest timescale. As such, we consider values of $\theta \in \{0.05, 0.2, 0.4, \frac{\pi}{4}\}$ and vary $\lambda \in [1.1, 1000]$.

First, we calculate the 2-Wasserstein distance, Kullback-Leibler divergence, and total variation distance between the full and reduced transition PDFs for a system in which the initial mean is in equilibrium $\mu_0 = (0, 0)$. The errors are plotted in Fig. 3.1 against the range of λ values and match the results found with ACFs in [53], meaning that our analysis can recover the findings in that work. Namely, Approach 1 errors are accurate for small timescale separation, but increase and level out for large timescale separation. Approach 2 is less accurate at small timescale separation but errors decrease as timescale separation increases.

Second, we calculate the errors of transition PDFs when the initial mean is out of equilibrium in some arbitrary state, namely $\mu_0 = (1, 0)$, for which errors are similar to that of the equilibrium case, and $\mu_0 = (5, -4)$, for which errors present a more significant change in trend. Errors plotted against λ are shown in Figures 3.2 and ??.

For the non-equilibrium case, we also plot errors for varying time lags. In Figure ??, we plot errors for $\lambda = 1.1$ and $\theta \in \{0.05, 0.2, 0.4, \frac{\pi}{2}\}$ across time values in $t \in [0, 10]$. We plot the same for $\lambda = 1000$ in Figure 3.4.

3.4.2 40-dimensional spring-mass system

To test this analysis in the multidimensional setting, we also consider a 40-dimensional spring-mass system. The spring mass system describes masses along a spring and their

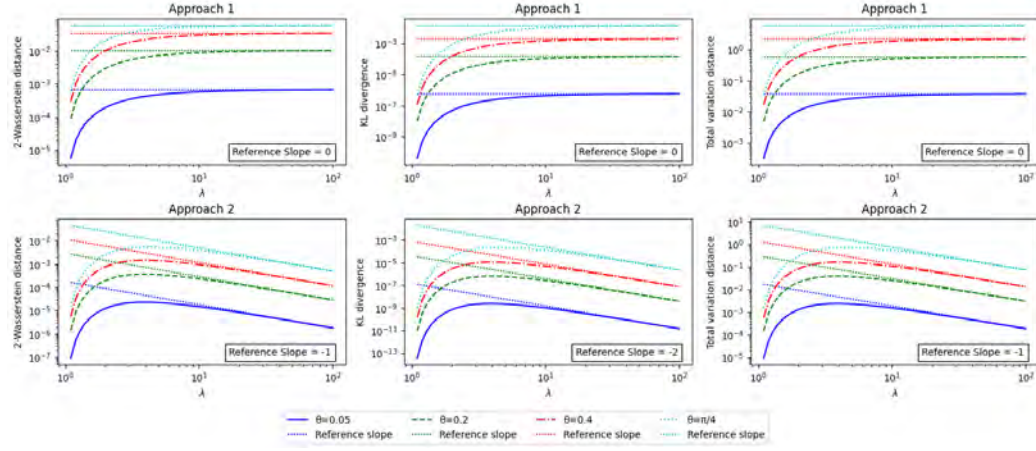


Figure 3.1: Error functions for a 2D system with initial mean in equilibrium ($\mu_0 = (0, 0)$) plotted against $\lambda \in [1.1, 1000]$. Top Row: Approach 1 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance. Bottom row: Approach 2 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance.

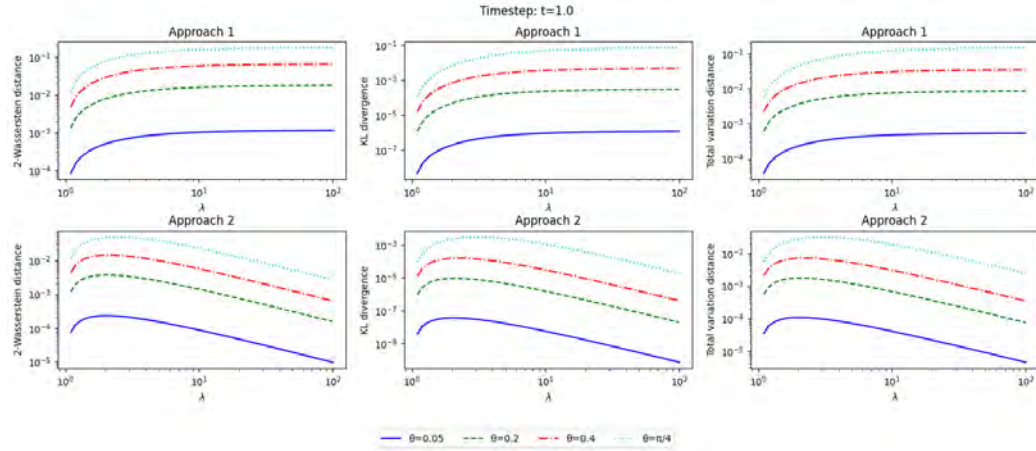


Figure 3.2: Error functions for a 2D system with initial mean outside of equilibrium ($\mu_0 = (1, 0)$) plotted against $\lambda \in [1.1, 1000]$. Top Row: Approach 1 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance. Bottom row: Approach 2 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance.

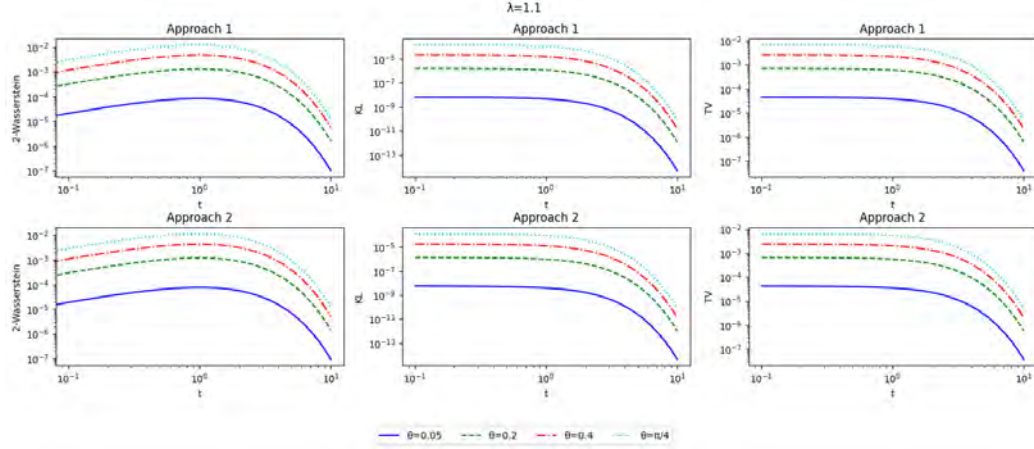


Figure 3.3: Error functions plotted against time lags in the range of $t \in [0, 10]$ for a case where the initial mean is outside of equilibrium $\mu_0 = (1, 0)$. Here, $\lambda = 1.1$. Top Row: Approach 1 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance. Bottom row: Approach 2 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance.

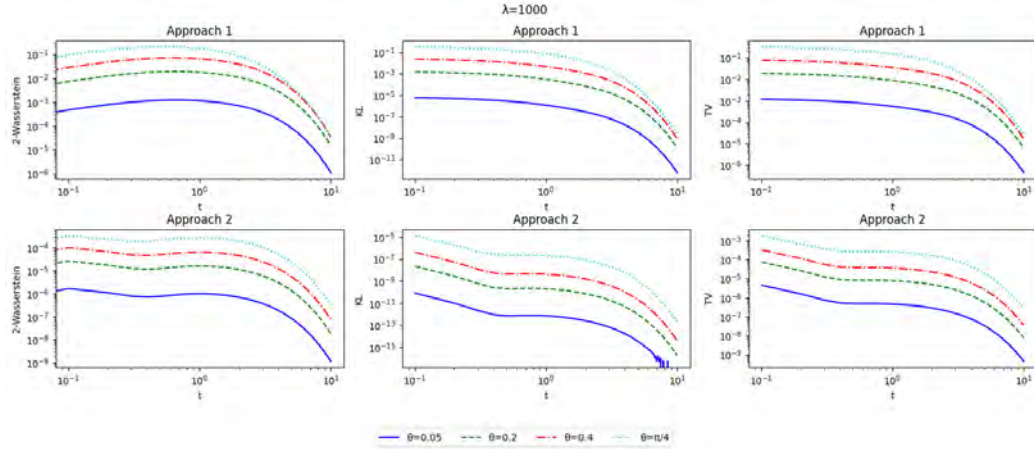


Figure 3.4: Error functions plotted against time lags in the range of $t \in [0, 10]$ for a case where the initial mean is outside of equilibrium $\mu_0 = (1, 0)$. Here, $\lambda = 1000$. Top Row: Approach 1 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance. Bottom row: Approach 2 2-Wasserstein Distance, Kullback-Leibler Divergence, and Total Variation Distance.

displacement from equilibrium in one dimension. For this example, the number of masses is set to be $N = 40$, and it is straightford to be extended in a more general setting.

The 1D chain spring-mass system modeled by an overdamped Langevin under inverse temperature β takes the form

$$d\mathbf{q}_t = -\mathbf{K}\mathbf{q}_t dt + \sqrt{2\beta^{-1}} d\mathbf{W}_t$$

where $\mathbf{K}$ is the stiffness matrix for the spring interations. This form is reached through taking the limit as the friction constant γ goes to infinity in the underdamped example. In this setting, if k is uniform across springs,

$$\mathbf{K} = k \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix} \quad (3.37)$$

If each mass $q_1, \dots, q_N$ has a different spring coefficient $k_1, \dots, k_n$, this becomes

$$\mathbf{K} = \begin{bmatrix} k_1 & -k_1 & 0 & \cdots & 0 \\ -k_1 & k_1 + k_2 & -k_2 & \cdots & 0 \\ 0 & -k_2 & k_2 + k_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -k_{N-1} \\ 0 & \cdots & 0 & -k_{N-1} & 2k_{N-1} \end{bmatrix} \quad (3.38)$$

For the experiments, we set $k_1 = 1, k_3 = 100$, and let k_2 vary from 0.1 to 20.01 with increments of 0.1. We examine a system with $N = 40$ and select $n = 2$ resolved variables using the map $\Phi = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0} \end{bmatrix}$, with the first and second masses being the resolved

variables. We will then systematically vary the ratio between stiff and soft variables to represent various timescale separation. Greater timescale separation corresponds to when there is a larger gap in these values. Larger values of k correspond to faster variables, variables that relax faster to equilibrium.

We compute the relative eigenvalue gap between slow and fast modes in the system as

$$\text{Relative Eigenvalue gap} = \frac{|\lambda_n(\mathbf{A}) - \lambda_{n-1}(\mathbf{A}_0)|}{\lambda_{n-1}(\mathbf{A}_0)} \quad (3.39)$$

and then plot this value against k_2 in Figure 3.5 to highlight that as k_2 increases with k_1 and k_3 fixed, the relative eigenvalue gap also increases. And so, k_2 controls the spectral gap present in a given regime. In this case, we define the angle between the resolved and unresolved variables to be

$$\theta = \min(\pi - x, x), \quad x = \frac{\|\boldsymbol{\alpha}\|_F}{\|\mathbf{A}_0\|_F}. \quad (3.40)$$

We also plot the resulting θ values against a range of k_2 values in Figure 3.5 to show that θ is also correlated positively with k_2 . The resulting transition PDF errors for this example are also plotted in Figure 3.5, and again we see that when coupling is weak, Approach 1 outperforms Approach 2, while Approach 2 prefers a larger gap. We also see that all three error metrics roughly agree with each other.

3.5 Conclusions

In this chapter, we have presented a novel lens through which to view the modeling error that emerges from reducing a system via coarse-graining. We utilize the error in transition probability law between full and reduced dynamics to capture the entire probability law of the dynamics, as opposed to just the first and second moments that are captured by autocovariance-based analysis. Our analysis is qualitatively consistent with the ACF analysis presented in [53], that Approach 1 outperforms

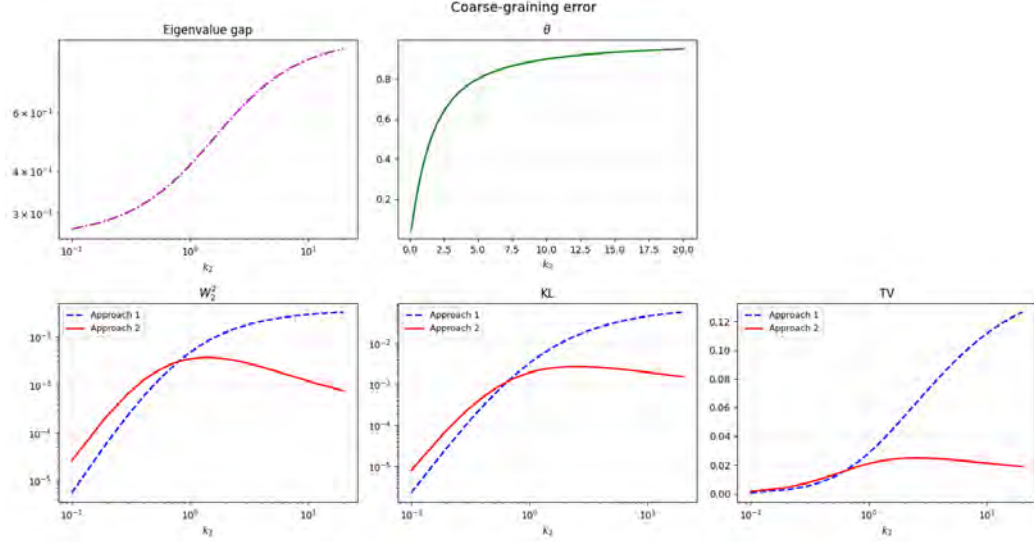


Figure 3.5: Top row: Eigenvalue gap (left) and theta values (right) plotted against values of k_2 . Bottom row: Errors plotted against k_2 , for initial mean out of equilibrium $\mu_0 = \mathbf{0}$

Approach 2 only when the timescale separation is relatively small, and otherwise Approach 2 outperforms the others. We also extend the analysis to quantify error when a model has an initial state that is out of equilibrium. The autocovariance analysis cannot capture this at all. Our analysis shows that the further a system starts from equilibrium, the less reliable the spectral gap can become as a metric.

Overall, these findings suggest that modeling error in coarse-graining is not a fixed property of the approximation, but rather that it depends on how far out of equilibrium the system begins, and transitional PDF error uniquely captures this. With the new finding that spectral gap is not the definitive metric for coarse-graining error analysis, in the next chapter we explore the use of error in relaxation time for analysis and propose a coarse-graining variable selection algorithm that uses this error as a guiding principle.

CHAPTER 4: LEADING ORDER ESTIMATES FOR RELAXATION TIME ERROR

With a more robust analysis result regarding modeling error given in the previous chapter, we turn our attention to the practical concern of how to select coarse-graining variables in real applications. This chapter provides a continuation of the study of modeling error and explores how error analysis can be incorporated into a computational coarse-grained variable selection framework. Namely, we first perform further error analysis, this time regarding the relaxation time of the full and reduced dynamics.

4.1 Relaxation time of a linear Langevin SDE and coarse graining approaches

Recall a linear Langevin SDE with symmetric positive definite (SPD) drift matrix $\mathbf{A}$ given by

$$dq_t = -\mathbf{A}q_t dt + \sqrt{2\beta^{-1}}d\mathbf{w}_t. \quad (4.1)$$

The asymptotic decay rate of (4.1) is

$$\kappa(\mathbf{A}) := \min_i \operatorname{Re} \lambda_i(\mathbf{A}) > 0, \quad (4.2)$$

and the relaxation time is its reciprocal,

$$\tau_{\text{rel}} := \frac{1}{\kappa(\mathbf{A})} = \frac{1}{\min_i \operatorname{Re} \lambda_i(\mathbf{A})}. \quad (4.3)$$

Because $\mathbf{A}$ is symmetric positive definite the spectrum is real and positive, so we have $\tau_{\text{rel}} = 1/\lambda_{\min}(\mathbf{A})$.

For coarse-graining approaches 0 and 1, the SDEs take the same form with a SPD drift matrix and thus relaxation times are computed in the same manner.

For Approach 0, the SDE and corresponding relaxation time are given by

$$d\xi_t = -\mathbf{A}_0\xi_t dt + \sqrt{2\beta^{-1}}d\mathbf{w}_{0,t}, \quad \tau_{\text{rel}}^{(0)} = \frac{1}{\lambda_{\min}(\mathbf{A}_0)} \quad (4.4)$$

and for Approach 1,

$$d\xi_t = -\mathbf{B}\xi_t dt + \sqrt{2\beta^{-1}}d\mathbf{w}_{0,t}, \quad \tau_{\text{rel}}^{(1)} = \frac{1}{\lambda_{\min}(\mathbf{B})} \quad (4.5)$$

where again $\mathbf{B} = \mathbf{A}_0 - \alpha\mathbf{A}_1^{-1}\alpha^*$.

Now, Approach 2 has a drift matrix that is not necessarily symmetric, given by

$$d\xi_t = -\mathbf{CB}\xi_t dt + \sqrt{2\beta^{-1}}\mathbf{C}d\mathbf{w}_{0,t}, \quad \mathbf{C} = (\mathbf{I} + \alpha\mathbf{A}_1^{-2}\alpha^*)^{-1}, \quad \mathbf{B} = \mathbf{A}_0 - \alpha\mathbf{A}_1^{-1}\alpha^*. \quad (4.6)$$

Unlike Approaches 0 and 1, the relaxation time formula is not automatic, since $\mathbf{CB}$ is not necessarily symmetric. Its eigenvalues need not be real, let alone positive. We verify both properties below. Although $\mathbf{CB}$ is not symmetric in general, it is similar to a normalized symmetric case

$$\tilde{\mathbf{B}} := \mathbf{C}^{1/2}\mathbf{B}\mathbf{C}^{1/2} = \mathbf{C}^{-1/2}(\mathbf{CB})\mathbf{C}^{1/2}, \quad (4.7)$$

via the invertible matrix $\mathbf{C}^{1/2}$ as $\mathbf{C} \succ 0$. Similarity transformations preserve eigenvalues, so

$$\lambda_i(\mathbf{CB}) = \lambda_i(\tilde{\mathbf{B}}) \quad \text{for all } i. \quad (4.8)$$

Since $\mathbf{B} \succ 0$ and $\mathbf{C}^{1/2}$ is invertible, $\tilde{\mathbf{B}} = \mathbf{C}^{1/2}\mathbf{B}\mathbf{C}^{1/2}$ is also symmetric positive definite.

Hence, the relaxation time of Approach 2 becomes

$$\tau_{\text{rel}}^{(2)} = \frac{1}{\lambda_{\min}(\mathbf{CB})} = \frac{1}{\lambda_{\min}(\mathbf{C}^{1/2}\mathbf{B}\mathbf{C}^{1/2})}. \quad (4.9)$$

4.1.1 Non-perturbative bounds on the ordering of relaxation time

Let $\mathbf{A} \succ 0$ with blocks as above, $\mathbf{B} = \mathbf{A}_0 - \alpha\mathbf{A}_1^{-1}\alpha^*$ and $\mathbf{C} = (\mathbf{I} + \alpha\mathbf{A}_1^{-2}\alpha^*)^{-1}$.

Then

$$\lambda_{\min}(\mathbf{A}) \leq \lambda_{\min}(\mathbf{B}) \leq \lambda_{\min}(\mathbf{A}_0), \quad \lambda_{\min}(\mathbf{CB}) \leq \lambda_{\min}(\mathbf{B}), \quad (4.10)$$

equivalently $\tau_{\text{rel}}^{(0)} \leq \tau_{\text{rel}}^{(1)} \leq \tau_{\text{rel}}^{(2)}$ and $\tau_{\text{rel}}^{(1)} \leq \tau_{\text{rel}}$.

(i) We have that $\mathbf{B}^{-1} = \Phi\mathbf{A}^{-1}\Phi^*$ with Φ having orthonormal rows, so by the Rayleigh quotient characterization $\lambda_{\max}(\mathbf{B}^{-1}) \leq \lambda_{\max}(\mathbf{A}^{-1})$, and inverting reverses the inequality. (ii) $\alpha\mathbf{A}_1^{-1}\alpha^* \succeq 0$ gives $\mathbf{B} \preceq \mathbf{A}_0$, hence $\lambda_{\min}(\mathbf{B}) \leq \lambda_{\min}(\mathbf{A}_0)$ by the Weyl inequality. (iii) Since $\alpha\mathbf{A}_1^{-2}\alpha^* \succeq 0$ we have $\mathbf{C} \preceq \mathbf{I}$, so $\lambda_{\max}(\mathbf{C}) \leq 1$. Ostrowski's theorem applied to the congruence $\tilde{\mathbf{B}} = \mathbf{C}^{1/2}\mathbf{B}\mathbf{C}^{1/2}$ yields $\lambda_{\min}(\tilde{\mathbf{B}}) \leq \lambda_{\max}(\mathbf{C}) \lambda_{\min}(\mathbf{B}) \leq \lambda_{\min}(\mathbf{B})$, and $\lambda_{\min}(\mathbf{CB}) = \lambda_{\min}(\tilde{\mathbf{B}})$ by similarity.

Lemma 4.1.1 confirms that Approaches 0, 1 and 2 order themselves correctly regardless of coupling strength. Consequently, the clock time scale of Approach 1 is slower than that of Approach 0, and the clock time scale of Approach 2 is slower than that of Approach 1. Moreover compared to the reference system, Approach 1 never overestimates the true relaxation time.

4.2 Relaxation time error bound: Approach 1

We are interested in the error between relaxation times of full dynamics and resolved dynamics, because it will give us insight as to whether the CG approaches preserve the slow physics, and this error quantifies the cost of ignoring memory as our approaches are Markovian approximations to non-Markovian CG derivations. All CG approaches

reproduce the equilibrium statistics of the resolved variables exactly, and their only error is dynamical. This is precisely what makes the relaxation-time gap the natural error metric. Given the relaxation time formulas, we will now pull from perturbation theory to derive leading order estimates on the error between relaxation time of true and resolved dynamics.

Recall that we decompose the full dynamics $\mathbf{A}$ into block matrix form by splitting the contributions of resolved variables ξ_t and unresolved variables ζ_t :

$$\mathbf{q}_t = \begin{bmatrix} \xi_t \\ \zeta_t \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}_0 & \alpha \\ \alpha^* & \mathbf{A}_1 \end{bmatrix} \quad (4.11)$$

where

$$\mathbf{A}_0 = \Phi \mathbf{A} \Phi^* \in \mathbb{R}^{n \times n} \quad \mathbf{A}_1 = \Psi \mathbf{A} \Psi^* \in \mathbb{R}^{m \times m} \quad \alpha = \Phi \mathbf{A} \Psi^* \in \mathbb{R}^{n \times m}$$

$$\mathbf{w}_{0,t} = \Phi \mathbf{w}_t \in \mathbb{R}^n \quad \mathbf{w}_{1,t} = \Psi \mathbf{w}_t \in \mathbb{R}^m$$

The coupling between the resolved and unresolved dynamics is governed by the matrix α . Now consider the case where the coupling is “small,” i.e, let $\alpha = \varepsilon \mathbf{\Gamma}$, where ε is some parameter that determines the magnitude of the coupling, and $\mathbf{\Gamma}$ is some matrix that determines the shape of the coupling. This decomposes the coupling into the structure, living in $\mathbf{\Gamma}$, and the strength, measured by ε , which is a common approach in perturbation theory.

So, reformulating our problem in perturbation theory language gives

$$\begin{aligned}\mathbf{A} &= \begin{bmatrix} \mathbf{A}_0 & 0 \\ 0 & \mathbf{A}_1 \end{bmatrix} + \varepsilon \begin{bmatrix} 0 & \mathbf{\Gamma} \\ \mathbf{\Gamma}^* & 0 \end{bmatrix} \\ &= \mathbf{A}^{(0)} + \varepsilon \mathbf{H}\end{aligned}$$

where $\mathbf{H}$ represents the perturbation block matrix and again, ε determines the strength of the coupling. Denote the eigenvalues of $\mathbf{A}$ to be (λ_k, v_k) . Denote the eigenvalue/vector pairs of $\mathbf{A}_0$ to be (γ_k, ν_k) , and the eigenpairs of $\mathbf{A}_1$ to be (w_k, μ_k) .

Via perturbation theory, we assume the eigenvalues and eigenvectors of $\mathbf{A}$ can be Taylor expanded in powers of ε , where the coefficients $a_0, a_1, \dots$ are scalars and $x_0, x_1, \dots$ are vectors:

$$\begin{aligned}\lambda(\varepsilon) &= \lambda(0) + \varepsilon \lambda'(0) + \varepsilon^2 \frac{1}{2} \lambda''(0) + \dots \\ &:= a_0 + \varepsilon a_1 + \varepsilon^2 a_2 \\ v(\varepsilon) &= x_0 + \varepsilon x_1 + \varepsilon^2 x_2 + \dots\end{aligned}$$

Here, $a_0, a_1, \dots$ and $x_0, x_1, \dots$ are correction terms. We additionally impose the normalization condition $x_0^T x_0 = 1$, which fixes the scaling ambiguity present in the definition of eigenvectors, together with $x_0^T x_1 = 0$.

To be precise about the methodology: these formulas track how a single eigenvalue of the matrix $\mathbf{A}$ changes as ε varies from 0. We are treating the value of the eigenvalue of interest as a function of ε , and then Taylor expanding around $\varepsilon = 0$.

We are interested in $\lambda_{min}(\mathbf{A})$. Note that the smallest eigenvalue of the unperturbed block matrix $\mathbf{A}^{(0)}$ is the smallest eigenvalue of either $\mathbf{A}_0$ or $\mathbf{A}_1$. For our purposes, we assume that the resolved variables are slow and the unresolved are fast, and so we

assume the strict inequality $\lambda_{min}(\mathbf{A}_0) < \lambda_{min}(\mathbf{A}_1)$. And so, $\lambda_{min}(\mathbf{A}^{(0)}) = \lambda_{min}(\mathbf{A}_0)$ exactly, while for the coupled matrix $\lambda_{min}(\mathbf{A}(\varepsilon)) = \lambda_{min}(\mathbf{A}_0) + O(\varepsilon^2)$, as we now show. The strict inequality guarantees that $\lambda_{min}(\mathbf{A}_0)$ is a simple eigenvalue of the full block matrix $\mathbf{A}^{(0)}$ (provided it is also simple within $\mathbf{A}_0$, which we assume), which is required for the perturbation theory to be valid; it also guarantees that the denominators $\lambda_{min}(\mathbf{A}_0) - w_k$ appearing below are bounded away from zero.

The eigenvectors of $\mathbf{A}^{(0)}$ associated with the $\mathbf{A}_0$ block take the form $\begin{bmatrix} \nu_k \\ 0 \end{bmatrix}$ with eigenvalue γ_k , and those associated with the $\mathbf{A}_1$ block take the form $\begin{bmatrix} 0 \\ \mu_k \end{bmatrix}$ with eigenvalue w_k . And so, $\lambda_{min}(\mathbf{A}^{(0)}) = \lambda_{min}(\mathbf{A}_0)$ and has eigenvector $v_0 = \begin{bmatrix} \nu_0 \\ 0 \end{bmatrix}$.

And so, our Taylor expansion will focus on $\lambda_{min}(\mathbf{A})$ and we start with

$$\lambda_{min}(\mathbf{A}(\varepsilon)) = \lambda_{min}(\mathbf{A}_0) + \varepsilon a_1 + \varepsilon^2 a_2 + O(\varepsilon^3)$$

Rather than computing derivatives explicitly, we can exploit the eigenvalue problem $\mathbf{A}(\varepsilon)x(\varepsilon) = \lambda(\varepsilon)x(\varepsilon)$. We will substitute in $\mathbf{A} = \mathbf{A}^{(0)} + \varepsilon\mathbf{H}$ and match powers of ε as follows. In what follows, we go through the details for a general eigenvalue, and apply results to the minimum eigenvalue at the end.

$$\begin{aligned} (\mathbf{A}^{(0)} + \varepsilon\mathbf{H})(x_0 + \varepsilon x_1 + \varepsilon^2 x_2 + \cdots) &= \\ (a_0 + \varepsilon a_1 + \varepsilon^2 a_2 + \cdots)(x_0 + \varepsilon x_1 + \varepsilon^2 x_2 + \cdots) &= \\ \mathbf{A}^{(0)}x_0 + \varepsilon(\mathbf{A}^{(0)}x_1 + \mathbf{H}x_0) + \varepsilon^2(\mathbf{A}^{(0)}x_2 + \mathbf{H}x_1) + \cdots &= \\ a_0x_0 + \varepsilon(a_0x_1 + a_1x_0) + \varepsilon^2(a_0x_2 + a_1x_1 + a_2x_0) + \cdots & \end{aligned}$$

Matching powers of ε gives us the following:

$$\begin{aligned} O(1) : \quad & \mathbf{A}^{(0)}x_0 = a_0x_0 \\ O(\varepsilon) : \quad & \mathbf{A}^{(0)}x_1 + \mathbf{H}x_0 = a_0x_1 + a_1x_0 \\ O(\varepsilon^2) : \quad & \mathbf{A}^{(0)}x_2 + \mathbf{H}x_1 = a_0x_2 + a_1x_1 + a_2x_0 \end{aligned}$$

Hence, we now solve the $O(\varepsilon)$ equation for a_1 :

$$\begin{aligned} \mathbf{A}^{(0)}x_1 + \mathbf{H}x_0 &= a_0x_1 + a_1x_0 \\ (\mathbf{A}^{(0)} - a_0\mathbf{I})x_1 &= (a_1\mathbf{I} - \mathbf{H})x_0 \\ x_0^T(\mathbf{A}^{(0)} - a_0\mathbf{I})x_1 &= x_0^T(a_1\mathbf{I} - \mathbf{H})x_0 \end{aligned}$$

Note that

$$x_0^T(\mathbf{A}^{(0)} - a_0\mathbf{I}) = x_0^T\mathbf{A}^{(0)} - x_0^Ta_0 = (\mathbf{A}^{(0)}x_0 - a_0x_0)^T$$

because $\mathbf{A}^{(0)}$ is symmetric, and

$$(\mathbf{A}^{(0)}x_0 - a_0x_0)^T = 0$$

from the fact that $\mathbf{A}^{(0)}x_0 = a_0x_0$, given by the $O(1)$ equation.

Continuing, we have

$$\begin{aligned} (\mathbf{A}^{(0)}x_0 - a_0x_0)^Tx_1 &= x_0^T(a_1\mathbf{I} - \mathbf{H})x_0 \\ 0 &= x_0^T(a_1\mathbf{I} - \mathbf{H})x_0 \\ 0 &= a_1x_0^Tx_0 - x_0^T\mathbf{H}x_0 \\ a_1 &= x_0^T\mathbf{H}x_0. \end{aligned}$$

This is the first-order correction. To remind us of our main goal, this means that we can now say:

$$\begin{aligned}\lambda_{\min}(\mathbf{A}(\varepsilon)) &= \lambda_{\min}(\mathbf{A}_0) + \varepsilon a_1 + \varepsilon^2 a_2 + \cdots \\ &= \lambda_{\min}(\mathbf{A}_0) + \varepsilon x_0^T \mathbf{P} x_0 + \cdots\end{aligned}$$

We do the same with the $O(\varepsilon^2)$ equation for a_2 . Taking the inner product with x_0 , we have

$$x_0^T \mathbf{A}^{(0)} x_2 + x_0^T \mathbf{H} x_1 = a_0 x_0^T x_2 + a_1 x_0^T x_1 + a_2$$

By symmetry of $\mathbf{A}^{(0)}$, we have $x_0^T \mathbf{A}^{(0)} x_2 = (\mathbf{A}^{(0)} x_0)^T x_2 = a_0 x_0^T x_2$, and so the term $x_0^T \mathbf{A}^{(0)} x_2$ on the left cancels exactly with the term $a_0 x_0^T x_2$ on the right (note that no condition on $x_0^T x_2$ is needed for this), leaving

$$x_0^T \mathbf{P} x_1 = a_1 x_0^T x_1 + a_2 \tag{4.12}$$

It remains to justify the condition $x_0^T x_1 = 0$. The eigenvector $x(\varepsilon)$ is defined only up to scaling at each order, and we may fix this freedom by requiring $x_0^T x(\varepsilon) = 1$ for all ε (the so-called intermediate normalization, standard in Rayleigh–Schrödinger perturbation theory). Matching powers of ε in $x_0^T (x_0 + \varepsilon x_1 + \varepsilon^2 x_2 + \cdots) = 1$ gives

$$O(1) : \quad x_0^T x_0 = 1,$$

$$O(\varepsilon) : \quad x_0^T x_1 = 0.$$

This cancels out the term $a_1 x_0^T x_1$ from above. With this, we can say that

$$x_0^T \mathbf{P} x_1 = a_2 \quad (4.13)$$

Now, we will expand x_1 in the eigenbasis of $\mathbf{A}^{(0)}$. Let $\{(\lambda_k, v_k)\}$ be all eigenpairs of $\mathbf{A}^{(0)}$, with (λ_0, v_0) the minimal pair as above. Write $x_1 = \sum_k c_k v_k$. Substituting into the $O(\varepsilon)$ equation and we get

$$\sum_k c_k (\lambda_k - \lambda_0) v_k = (a_1 \mathbf{I} - \mathbf{H}) v_0 \quad (4.14)$$

Taking inner product with v_j with $j \neq 0$:

$$\begin{aligned} c_j (\lambda_j - \lambda_0) &= a_1 v_j^T v_0 - v_j^T \mathbf{H} v_0 \\ c_j &= \frac{-v_j^T \mathbf{H} v_0}{\lambda_j - \lambda_0} = \frac{v_j^T \mathbf{H} v_0}{\lambda_0 - \lambda_j} \end{aligned}$$

So $x_1 = \sum_{k \neq 0} \frac{v_k^T \mathbf{H} v_0}{\lambda_0 - \lambda_k} v_k$. Plugging into the equation for a_2 :

$$a_2 = \sum_{k \neq 0} \frac{|v_k^T \mathbf{H} v_0|^2}{\lambda_0 - \lambda_k}$$

Putting everything together, we find that

$$\lambda_{\min}(\mathbf{A}) = \lambda_{\min}(\mathbf{A}_0) + \varepsilon v_0^T \mathbf{H} v_0 + \varepsilon^2 \sum_{k \neq 0} \frac{|v_k^T \mathbf{H} v_0|^2}{\lambda_0 - \lambda_k} + O(\varepsilon^3). \quad (4.15)$$

However, note that $a_1 = x_0^T \mathbf{P} x_0 = \begin{bmatrix} \nu_0 \\ 0 \end{bmatrix}^T \begin{bmatrix} 0 & \mathbf{\Gamma} \\ \mathbf{\Gamma}^* & 0 \end{bmatrix} \begin{bmatrix} \nu_0 \\ 0 \end{bmatrix} = 0$. And so we have

$$\lambda_{\min}(\mathbf{A}) = \lambda_{\min}(\mathbf{A}_0) + \varepsilon^2 \sum_{k \neq 0} \frac{|v_k^T \mathbf{H} v_0|^2}{\lambda_0 - \lambda_k} + O(\varepsilon^4). \quad (4.16)$$

In fact, the remainder above is $O(\varepsilon^4)$ rather than $O(\varepsilon^3)$: letting $\mathbf{S} = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & -\mathbf{I} \end{bmatrix}$, we have $\mathbf{S}\mathbf{A}(\varepsilon)\mathbf{S} = \mathbf{A}(-\varepsilon)$, so $\mathbf{A}(\varepsilon)$ and $\mathbf{A}(-\varepsilon)$ are similar and share the same spectrum. Each eigenvalue is therefore an even function of ε , and all odd-order corrections vanish, not only a_1 .

Now consider the sum $\sum_{k \neq 0} \frac{|v_k^T \mathbf{H} v_0|^2}{\lambda_0 - \lambda_k}$ that ranges over all eigenvectors of $\mathbf{A}^{(0)}$, i.e., those coming from both the $\mathbf{A}_0$ and $\mathbf{A}_1$ blocks. For an eigenvector $v_k = \begin{bmatrix} \nu_k \\ 0 \end{bmatrix}$ coming from the $\mathbf{A}_0$ block, we have $v_k^T \mathbf{H} v_0 = \begin{bmatrix} \nu_k \\ 0 \end{bmatrix}^T \begin{bmatrix} 0 & \mathbf{\Gamma} \\ \mathbf{\Gamma}^* & 0 \end{bmatrix} \begin{bmatrix} \nu_0 \\ 0 \end{bmatrix} = 0$. Thus, only the eigenvectors from the $\mathbf{A}_1$ block contribute; for these, $v_k = \begin{bmatrix} 0 \\ \mu_k \end{bmatrix}$ and we can write $v_k^T \mathbf{P} v_0 = \mu_k^T \mathbf{\Gamma}^* \nu_0$. The perturbation equation now reads

$$\lambda_{\min}(\mathbf{A}) = \lambda_{\min}(\mathbf{A}_0) + \varepsilon^2 \sum_k \frac{|\mu_k^T \mathbf{\Gamma}^* \nu_0|^2}{\lambda_{\min}(\mathbf{A}_0) - w_k} + O(\varepsilon^4), \quad (4.17)$$

where the sum now ranges over the eigenpairs (w_k, μ_k) of $\mathbf{A}_1$.

We can apply the same approach for the matrix $\mathbf{B} = \mathbf{A}_0 - \varepsilon^2 \mathbf{\Gamma} \mathbf{A}_1^{-1} \mathbf{\Gamma}^*$ but let $\delta = \varepsilon^2$ and $\mathbf{H} = -\mathbf{\Gamma} \mathbf{A}_1^{-1} \mathbf{\Gamma}^*$. Taking the perturbation approach on $\mathbf{A}_0 + \delta(-\mathbf{\Gamma} \mathbf{A}_1^{-1} \mathbf{\Gamma}^*)$, we get

$$\lambda_{\min}(\mathbf{B}) = \lambda_{\min}(\mathbf{A}_0) + \varepsilon^2 \nu_0^T (-\mathbf{\Gamma} \mathbf{A}_1^{-1} \mathbf{\Gamma}^*) \nu_0 + O(\varepsilon^4)$$

Expanding $\nu_0^T \mathbf{\Gamma} \mathbf{A}_1^{-1} \mathbf{\Gamma}^* \nu_0$ in the eigenbasis of $\mathbf{A}_1$, we get $\nu_0^T \mathbf{\Gamma} \mathbf{A}_1^{-1} \mathbf{\Gamma}^* \nu_0 = \sum_k \frac{|\mu_k^T \mathbf{\Gamma}^* \nu_0|^2}{w_k}$.

Now, taking the difference, we have

$$\begin{aligned}
\lambda_{\min}(\mathbf{A}) - \lambda_{\min}(\mathbf{B}) &= \varepsilon^2 \sum_k |\mu_k^T \mathbf{\Gamma}^* \nu_0|^2 \left(\frac{1}{\lambda_{\min}(\mathbf{A}_0) - w_k} + \frac{1}{w_k} \right) + O(\varepsilon^4) \\
&= \varepsilon^2 \lambda_{\min}(\mathbf{A}_0) \sum_k \frac{|\mu_k^T \mathbf{\Gamma}^* \nu_0|^2}{(\lambda_{\min}(\mathbf{A}_0) - w_k)w_k} + O(\varepsilon^4)
\end{aligned}$$

Since $0 < \lambda_{\min}(\mathbf{A}_0) < w_k$ for all k , each term of the sum is negative, and so $\lambda_{\min}(\mathbf{A}) \leq \lambda_{\min}(\mathbf{B})$ to leading order, consistent with the Schur complement inequality derived earlier. Define

$$g = \lambda_{\min}(\mathbf{A}_0) \sum_k \frac{|\mu_k^T \mathbf{\Gamma}^* \nu_0|^2}{(w_k - \lambda_{\min}(\mathbf{A}_0))w_k} > 0, \quad (4.18)$$

so that $\lambda_{\min}(\mathbf{A}) - \lambda_{\min}(\mathbf{B}) = -\varepsilon^2 g + O(\varepsilon^4)$, i.e.,

$$\lambda_{\min}(\mathbf{B}) = \lambda_{\min}(\mathbf{A}) + \varepsilon^2 g + O(\varepsilon^4).$$

Taylor expanding,

$$\begin{aligned}
\frac{1}{\lambda_{\min}(\mathbf{B})} &= \frac{1}{\lambda_{\min}(\mathbf{A}) + \varepsilon^2 g} + O(\varepsilon^4) \\
&= \frac{1}{\lambda_{\min}(\mathbf{A}) \left(1 + \frac{\varepsilon^2 g}{\lambda_{\min}(\mathbf{A})}\right)} + O(\varepsilon^4) \\
&= \frac{1}{\lambda_{\min}(\mathbf{A})} - \frac{\varepsilon^2 g}{\lambda_{\min}(\mathbf{A})^2} + O(\varepsilon^4)
\end{aligned}$$

where the geometric series expansion in the last step is valid provided $\varepsilon^2 g \ll \lambda_{\min}(\mathbf{A})$; the remainder is $O(\varepsilon^4)$ only in this regime. Also taking note of the fact that $\lambda_{\min}(\mathbf{A})^2 = \lambda_{\min}(\mathbf{A}_0)^2 + O(\varepsilon^2)$, we have

$$\left| \frac{1}{\lambda_{\min}(\mathbf{B})} - \frac{1}{\lambda_{\min}(\mathbf{A})} \right| = \frac{\varepsilon^2 g}{\lambda_{\min}(\mathbf{A}_0)^2} + O(\varepsilon^4)$$

In this form, in which g is written in terms of the eigenbasis of $\mathbf{A}_1$, we can see exactly which unresolved modes contribute and how strongly.

4.3 Relaxation time error bound: Approach 2

We now extend the perturbative analysis to Approach 2. Since $\mathbf{CB}$ is not symmetric, we perturb the symmetric matrix $\tilde{\mathbf{B}} = \mathbf{C}^{1/2}\mathbf{B}\mathbf{C}^{1/2}$ instead, which shares $\mathbf{CB}$'s eigenvalues by similarity.

With $\alpha = \varepsilon\mathbf{\Gamma}$ and $X := \mathbf{\Gamma}\mathbf{A}_1^{-2}\mathbf{\Gamma}^*$,

$$\mathbf{C} = (\mathbf{I} + \varepsilon^2 X)^{-1} = \mathbf{I} - \varepsilon^2 X + O(\varepsilon^4), \quad \mathbf{C}^{1/2} = \mathbf{I} - \frac{\varepsilon^2}{2} X + O(\varepsilon^4). \quad (4.19)$$

Recalling $\mathbf{B} = \mathbf{A}_0 - \varepsilon^2 Y + O(\varepsilon^4)$ with $Y := \mathbf{\Gamma}\mathbf{A}_1^{-1}\mathbf{\Gamma}^*$, we multiply out

$$\tilde{\mathbf{B}} = \mathbf{C}^{1/2}\mathbf{B}\mathbf{C}^{1/2} = \mathbf{A}_0 - \varepsilon^2 \left[Y + \frac{1}{2}(X\mathbf{A}_0 + \mathbf{A}_0 X) \right] + O(\varepsilon^4) =: \mathbf{A}_0 + \varepsilon^2 \mathbf{Q} + O(\varepsilon^4). \quad (4.20)$$

This has exactly the same shape as $\mathbf{B}$'s expansion, so the first-order perturbation result derived above for $\mathbf{B}$ applies verbatim to $\tilde{\mathbf{B}}$:

$$\lambda_{\min}(\mathbf{CB}) = \lambda_{\min}(\tilde{\mathbf{B}}) = \lambda_{\min}(\mathbf{A}_0) + \varepsilon^2 \nu_0^\top \mathbf{Q} \nu_0 + O(\varepsilon^4). \quad (4.21)$$

Using $\mathbf{A}_0 \nu_0 = \lambda_{\min}(\mathbf{A}_0) \nu_0$ and symmetry of X , we have simplifies:

$$\nu_0^\top X \mathbf{A}_0 \nu_0 = \nu_0^\top \mathbf{A}_0 X \nu_0 = \lambda_{\min}(\mathbf{A}_0) \nu_0^\top X \nu_0, \quad (4.22)$$

so

$$\nu_0^\top \mathbf{Q} \nu_0 = -\nu_0^\top Y \nu_0 - \lambda_{\min}(\mathbf{A}_0) \nu_0^\top X \nu_0. \quad (4.23)$$

Expanding in the eigenbasis of $\mathbf{A}_1$, with $c_k := \mu_k^\top \mathbf{\Gamma}^* \nu_0$,

$$\nu_0^\top Y \nu_0 = \sum_k \frac{|c_k|^2}{w_k}, \quad \nu_0^\top X \nu_0 = \sum_k \frac{|c_k|^2}{w_k^2}, \quad (4.24)$$

which gives

$$\lambda_{\min}(\mathbf{CB}) = \lambda_{\min}(\mathbf{A}_0) - \varepsilon^2 \sum_k |c_k|^2 \frac{w_k + \lambda_{\min}(\mathbf{A}_0)}{w_k^2} + O(\varepsilon^4). \quad (4.25)$$

Write $\lambda_0 := \lambda_{\min}(\mathbf{A}_0)$. All three eigenvalues $\lambda_{\min}(\mathbf{A})$, $\lambda_{\min}(\mathbf{B})$, $\lambda_{\min}(\mathbf{CB})$ take the common form $\lambda_0 - \varepsilon^2 \sum_k |c_k|^2 f_k + O(\varepsilon^4)$ with mode weights

$$f_k^{(A)} = \frac{1}{w_k - \lambda_0}, \quad f_k^{(B)} = \frac{1}{w_k}, \quad f_k^{(CB)} = \frac{w_k + \lambda_0}{w_k^2}. \quad (4.26)$$

Since $0 < \lambda_0 < w_k$ for all k , one has $f_k^{(B)} < f_k^{(CB)} f_k^{(A)}$ pointwise, and therefore

$$\lambda_{\min}(\mathbf{A}) < \lambda_{\min}(\mathbf{CB}) < \lambda_{\min}(\mathbf{B}) \implies \tau_{\text{rel}}^{(0)} \leq \tau_{\text{rel}}^{(1)} \leq \tau_{\text{rel}}^{(2)} \leq \tau_{\text{rel}}, \quad (4.27)$$

confirming quantitatively that Approach 2 sits strictly between Approach 1 and the true relaxation time.

Define

$$h := \lambda_0^2 \sum_k \frac{|c_k|^2}{w_k^2(w_k - \lambda_0)} > 0. \quad (4.28)$$

A direct computation gives $f_k^{(A)} - f_k^{(CB)} = \lambda_0^2/[w_k^2(w_k - \lambda_0)]$, so

$$\lambda_{\min}(\mathbf{CB}) = \lambda_{\min}(\mathbf{A}) + \varepsilon^2 h + O(\varepsilon^4). \quad (4.29)$$

Taylor expanding as before,

$$\left| \tau_{\text{rel}}^{(2)} - \tau_{\text{rel}} \right| = \frac{\varepsilon^2 h}{\lambda_0^2} + O(\varepsilon^4), \quad \text{vs.} \quad \left| \tau_{\text{rel}}^{(1)} - \tau_{\text{rel}} \right| = \frac{\varepsilon^2 g}{\lambda_0^2} + O(\varepsilon^4). \quad (4.30)$$

Comparing h and g term by term, each mode's contribution to h is exactly that mode's contribution to g rescaled by $\lambda_0/w_k \in (0, 1)$:

$$h = \sum_k \frac{\lambda_0}{w_k} g_k < \sum_k g_k = g, \quad g_k := \frac{\lambda_0 |c_k|^2}{(w_k - \lambda_0) w_k}. \quad (4.31)$$

Thus the diffusion correction $\mathbf{C}$ in Approach 2 suppresses each unresolved mode's contribution to the relaxation-time error by precisely the ratio λ_0/w_k of the resolved to unresolved eigenvalues: modes with $w_k \gg \lambda_0$ are corrected almost completely, while modes with w_k close to λ_0 (poor separation) retain most of their Approach-1 error. This is consistent with the fact that we require sufficient timescale separation for this analysis.

4.4 Conclusions

This section presented new leading order estimates on the error in relaxation time between the full and reduced dynamics, which rely on the coupling strength between the resolved and unresolved variables. This work could be utilized to develop a coarse-graining variable selection algorithm for cases in which the resolved variables are required to have physical meaning.

So far, we have analyzed numerical stability in time integration and modeling error in the classical setting. We have also showed the potential benefit of incorporating machine learning approaches into classical frameworks. In light of the emergence of machine learning based modeling approaches, the question arises: what changes and when the approximation is represented by a Scientific Machine Learning model? In the remaining sections, we carefully examine some popular machine learning-based

modeling approaches, and explore ways in which classical techniques can improve their accuracy. Meanwhile, we continue to highlight the importance of numerical stability and modeling error, while introducing computational cost as a third consideration.

CHAPTER 5: PRELIMINARIES ON SCIENTIFIC MACHINE LEARNING METHODS

In the following three chapters, we pivot to the machine learning setting and explore various error modes of common machine learning models and develop model improvements to address them. In each of these projects, the focus is on some combination of numerical accuracy, modeling error or efficiency, further elucidating the need to balance these facets in mathematical modeling frameworks. Through these efforts, we show that insights gained from the classical setting can be transferred to modern methods.

We begin with an overview of the machine learning-based methods that have shown promise in recent years for fluid and molecular dynamics modeling. Machine learning methods can be data-driven or physics-informed, or a combination of both. This work focuses primarily on the physics-informed versions of these models, however, any of these approaches can be made data-driven by simply adding terms into their loss functions that enforce the data. We focus on physics-informed approaches due to the appeal of not requiring expensive data to train the machine learning methods, which is not possible in classical approaches.

5.1 Physics-informed neural networks

In this section, we present physics-informed neural networks (PINNs). In recent years, a huge research focus has been on scientific machine learning methods for physical systems [58, 59, 60], for example, fluid mechanics and rheology [61, 62, 63, 64, 65, 66, 67, 68], metamaterial development [69, 70, 71], high speed flows [72], and power systems [73, 74, 75, 76, 77], among many other applications. In particular, physics-

informed neural networks, or PINNs [8], allow for accurately representing differential operators through automatic differentiation, leading to finding the solution to partial differential equations (PDEs) without explicit mesh generation. As physical systems often lack robust data, PINNs incorporate the differential equations of a system in the loss function of a neural network. Using automatic differentiation to minimize the loss function, the physics acts as a constraint for potential output solutions. The resulting models are able to train effectively using minimal training data. PINNs have been used successfully in a host of problems (see [60] for a recent review and future perspectives).

PINNs [8] allow for finding solutions to physical systems and discovery of ordinary or partial differential equations with limited training data. PINNs, as illustrated in Fig. 5.1, incorporate the dynamics of the system in the loss function of the network to enforce physical laws or domain expertise. Specifically, we define a system over an open, bounded domain $\Omega \in \mathbb{R}^n$ with boundary $\partial\Omega$ as follows:

$$s_t + \mathcal{N}_x[s] = \mathbf{0}, \quad x \in \Omega, t \in [0, T], \quad (5.1)$$

$$s(x, t) = g(x, t), \quad x \in \partial\Omega, t \in [0, T], \quad (5.2)$$

$$s(x, 0) = u(x), \quad x \in \Omega, \quad (5.3)$$

where g and u are given functions for the boundary and initial conditions, respectively, while x denotes the spatial coordinates, t is the temporal coordinate, and $\mathcal{N}_x$ is a differential operator with respect to x . Our goal is to utilize deep neural networks (DNNs) to approximate $s(x, t)$. The parameters (weights and biases) for the network are denoted by θ , which are optimized to minimize the following loss function:

$$\mathcal{L}(\theta) = \lambda_{ic}\mathcal{L}_{ic}(\theta) + \lambda_{bc}\mathcal{L}_{bc}(\theta) + \lambda_r\mathcal{L}_r(\theta), \quad (5.4)$$

with subscripts ic, bc, r corresponding to loss terms with respect to initial conditions,

boundary conditions, and PDE residual. We sample N_{ic} , N_{bc} , and N_r points in the space-time domain for the initial conditions, boundary conditions, and residual collocation points, respectively. The training data sets are denoted by

$$\{(x_{ic}^i), u(x_{ic}^i)\}_{i=1}^{N_{ic}}, \{(x_{bc}^i, t_{bc}^i), g(x_{bc}^i, t_{bc}^i)\}_{i=1}^{N_{bc}}, \text{ and } \{(x_r^i, t_r^i)\}_{i=1}^{N_r}.$$

A common choice for the terms in the loss function is through the mean squared errors (MSEs) according to

$$\mathcal{L}_{ic}(\theta) = \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} [\mathcal{F}(x_{ic}^i, 0; \theta) - u(x_{ic}^i)]^2, \quad (5.5)$$

$$\mathcal{L}_{bc}(\theta) = \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} [\mathcal{F}(x_{bc}^i, t_{bc}^i; \theta) - g(x_{bc}^i, t_{bc}^i)]^2, \quad (5.6)$$

$$\mathcal{L}_r(\theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} [\mathcal{F}_t(x_r^i, t_r^i; \theta) + \mathcal{N}_x[\mathcal{F}(x_r^i, t_r^i; \theta)]]^2, \quad (5.7)$$

where $\mathcal{F}(x, t; \theta)$ denotes the output of the network with parameters θ at (x, t) .

The selection of the weights λ_{ic} , λ_{bc} , and λ_r can have an impact on physics-informed training and several methods have been developed for adaptively choosing these weights, including soft-attention mechanism weights [78], the neural tangent kernel weights [79], among other variations [80, 81, 82]. However, we have opted to hand-pick these terms based on knowledge of the system being learned instead of focusing on the details of adaptive weight selection, which is not the focus of this work. The exact values used in the examples shown here given in Section ???. Nonetheless, we note that any of the aforementioned adaptive weighting methods could be applied in training stacking PINNs seamlessly.

5.2 Physics-informed deep operator networks

In this section, we present physics-informed deep operator networks (PI-DeepONets) which provide the basis for the work in Chapter 8. Physics-informed operator training

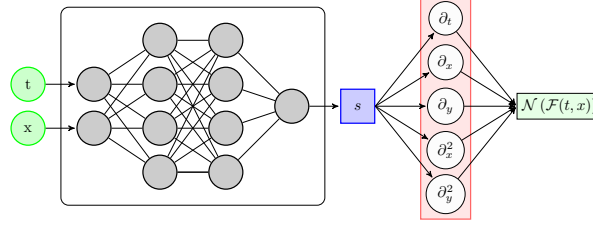


Figure 5.1: Physics-informed neural network

allows for learning the solution as the map between two Banach spaces. While PINNs learn the solution for a single initial condition and boundary condition, PI-DeepONets learn the solution for a family of initial conditions forming a Banach space. Several methods of operator learning are currently extremely popular [83, 84, 85, 86]. In this work, we will focus on DeepONets [84].

Consider a general parametric PDE of the form

$$\mathcal{N}(u, s) = 0 \quad (5.8)$$

with boundary conditions

$$\mathcal{B}(u, s) = 0. \quad (5.9)$$

The PDE solution is denoted by $\mathcal{G}(u) = s(u)$ for $\mathcal{G} : \mathcal{U} \rightarrow \mathcal{S}$, where $\mathcal{U}$ is the space of input parameters, generally the initial conditions, with $u \in \mathcal{U}$, and $\mathcal{S}$ is the space of PDE solutions on a domain Ω . $s \in \mathcal{S}$ is an unknown function governed by the PDE system.

A standard DeepONet consists of two DNNs, the branch and the trunk. The branch and trunk networks are trained simultaneously and combined in a dot product to express the solution. The input to the branch network is the function $u \in \mathcal{U}$ discretized at a discrete set of M points, which are at fixed locations for each sample in the training and test set. The input to the trunk network is the independent variables, typically time and spatial coordinates. The output of the DeepONet is

denoted by

$$\mathcal{G}_\theta(\mathbf{u})(\mathbf{x}) = \sum_{k=1}^p b_k(u_1, \dots, u_M) t_k(\mathbf{x}) \quad (5.10)$$

where θ denotes the trainable parameters of the unstacked DeepONet [87, 84].

As in our previous work [88], we choose to use “modified” DeepONets [89], which introduce encoder layers for the branch and trunk nets. At each hidden layer, the branch and trunk are combined in a convex combination with the encoder layers. This modification has been shown to increase the accuracy of DeepONet training [89].

Physics-informed DeepONets (PI-DeepONets) are trained in much the same manner as PINNs, with similar loss functions. The loss function is given by

$$\mathcal{L}_{DoN}(\theta) = \lambda_{bc} \mathcal{L}_{bc}(\theta) + \lambda_r \mathcal{L}_r(\theta) \quad (5.11)$$

where λ_{bc} and λ_r are tunable weighting terms and $\mathcal{L}_{bc}(\theta)$ and $\mathcal{L}_{res}(\theta)$ are the MSEs in satisfying the PDE system, given by

$$\mathcal{L}_r(\theta) = \frac{1}{N} \frac{1}{R} \sum_{i=1}^N \sum_{j=1}^R \left[\mathcal{N}(u^i, \mathcal{G}_\theta(\mathbf{u}^i)(x_j^i)) \right]^2 \quad (5.12)$$

$$\mathcal{L}_{bc}(\theta) = \frac{1}{N} \frac{1}{R_{bc}} \sum_{i=1}^N \sum_{j=1}^{R_{bc}} \left[\mathcal{B}(u^i, \mathcal{G}_\theta(\mathbf{u}^i)(x_j^i)) \right]^2 \quad (5.13)$$

The number of collocation points are R for the residual term and R_{bc} for the boundary condition terms, which are randomly sampled in the domain Ω and the domain of $\mathcal{B}$, and denoted by $\{x_j^i\}_{i=1}^r$ and $\{x_j^i\}_{i=1}^{R_{bc}}$.

5.3 Kolmogorov-Arnold Networks

In this section, we present the formulation of Kolmogorov-Arnold networks (KANs), which are the foundation of the work provided in Chapter ?? . In [90], the authors

proposed approximating a multivariate function $f(\mathbf{x})$ by a model of the form

$$f(\mathbf{x}) \approx \sum_{i_{n_l-1}=1}^{m_{n_l-1}} \varphi_{n_l-1, i_{n_l}, i_{n_l-1}} \left(\sum_{i_{n_l-2}=1}^{m_{n_l-2}} \cdots \left(\sum_{i_2=1}^{m_2} \varphi_{2, i_3, i_2} \left(\sum_{i_1=1}^{m_1} \varphi_{1, i_2, i_1} \left(\sum_{i_0=1}^{m_0} \varphi_{0, i_1, i_0}(x_{i_0}) \right) \right) \right) \cdots \right), \quad (5.14)$$

which they denote as a Kolmogorov-Arnold network. Here, n_l is the number of layers in the KAN, $\{m_j\}_{j=0}^{n_l}$ is the number of nodes per layer, $\varphi_{i,j,k}$ are the univariate activation functions, and $\mathbf{x} = (x_1, \dots, x_{m_0})$. We denote the right-hand side of eq:KAN as $\mathcal{K}(\mathbf{x})$. The activation functions $\varphi_{k-1, i_k, i_{k-1}}$ in the k th layer are polynomials of degree k on a grid with g grid points. They are represented by a weighted combination of a basis function $b(x)$ and a B-spline,

$$\varphi(x) = w_b b(x) + w_s \text{spline}(x), \quad (5.15)$$

where

$$b(x) = \frac{x}{1 + e^{-x}}$$

and

$$\text{spline}(x) = \sum_i c_i B_i(x).$$

Here, $B_i(x)$ is a polynomial of degree k , and c_i , w_b , and w_s are trainable parameters.

KANs evaluate the B-splines on a precomputed grid. In one dimension, in a domain $[a, b]$, a grid with g_1 intervals has grid points $\{t_0 = a, t_1, t_2, \dots, t_{g_1} = b\}$; cf. [90]. Grid extension [90] allows for fitting a new, fine-grained spline to a coarse-grained spline, increasing the expressivity of the KAN. The coarse splines are transferred to the fine splines following the procedure described in [90] and [91].

In this work, we do not consider the method for enforcing sparsity of the trainable activation functions as outlined in [90] and instead only consider the mean squared error (MSE) in the loss function. Many variations of KANs have been proposed recently. However, in this work, we consider only the formulation as outlined in [90],

although we note that the domain decomposition method presented here could be combined with many variants of KANs.

5.4 Conclusions

This chapter contained a presentation on three major methods of focus in the field of SciML: PINNs, PI-DeepONets, and KANs. These methods show promise on their own, as shown in referenced works, but it is important to provide an answer on what practitioners can do when one of these models is desirable but not performing well. In the next chapter, we discuss how the mathematical modeling framework changes when improvements are made (pulling from classical approaches) to the models architecture and training hierarchy.

CHAPTER 6: STACKED MULTIFIDELITY PHYSICS-INFORMED NEURAL NETWORKS

We first explore how improving the models architecture and training hierarchy can improve accuracy via the problem of training PINNs for chaotic dynamical systems. Chaotic systems provide an ideal setting for this work because the presence of fixed points presents challenges for PINNs to approximate without access to data. The work in this chapter is based on the work in “Stacked networks improve physics-informed training: applications to neural networks and deep operator networks,” published in AIMS Foundation of Data Sciences.

Despite its promise, physics-informed training with little-to-no data can be challenging. For instance, PINNs tend to fail for dynamical systems for many possible reasons. The fixed points of a system, whether stable or unstable, create attractive optimal solutions that may not match the desired solution for a given initial condition [92]. Solutions to dynamical systems must fit the physical laws of the system, initial and boundary conditions, as well as data. Optimizing multiple objectives in this way can create issues, as minimizing the full loss function does not guarantee convergence with respect to each particular loss term. This can lead to a failure of the proposed solution to represent the solution specified by the initial or boundary conditions, which typically looks like a model training to the trivial solution. In a similar manner to PINNs, physics-informed DeepONets (PI-DeepONets) can be extremely difficult to train in the absence of data [93].

Since the introduction of PINNs, several methods have been introduced to improve their training [94]. One portion of these efforts have been devoted to building unique architectures, including the works on multifidelity networks [95] and finite basis do-

main decomposition schemes [96, 97]. Multifidelity PINNs [95] traditionally use some data [98, 99], in addition to a physics-informed term, to train more accurately than allowed by physics alone. In finite basis PINNs, neural networks approximate the solution to the differential equation as a finite set of basis functions with compact support [96, 97]. Similarly for DeepONets, when some low-resolution numerical data is available multifidelity DeepONets [100, 88, 101] have been shown to improve the training for physics-informed problems.

6.1 When PINNs fail to train

Despite the promising results of PINNs, there exist some important examples for which PINNs fail to train with the standard framework, such as a pendulum with damping and the wave equation [92, 102, 103, 104]. For dynamical systems in particular, solutions must fit the physical laws on the system, initial and boundary conditions, as well as any given data. With complex loss functions, minimizing the full function does not ensure convergence for each particular loss term.

A simple example, yet one where a PINN struggles significantly to train, is a damped pendulum, shown in Fig. 6.1. The system is governed by a system of two first-order ordinary differential equations (ODEs) for $t \in [0, T]$

$$\frac{ds_1}{dt} = s_2, \tag{6.1}$$

$$\frac{ds_2}{dt} = -\frac{b}{m}s_2 - \frac{g}{L}\sin(s_1), \tag{6.2}$$

where s_1 and s_2 are the position and velocity of the pendulum, respectively. The initial conditions are chosen as $s_1(0) = s_2(0) = 1$. We take $m = L = 1$, $b = 0.05$, $g = 9.81$, and $T = 20$ as in [93].

While the trained outputs from the ten PINNs shown in Fig. 6.1 satisfy the initial condition, the solutions decay to zero past $t \approx 14$. We note that $s_1 = s_2 = 0$ would be a true solution for the pendulum if the initial conditions were $s_1(0) = s_2(0) = 0$.

Thus, once the solution decays to zero, the loss term associated with the residual is minimized, however, the solution is not correct.

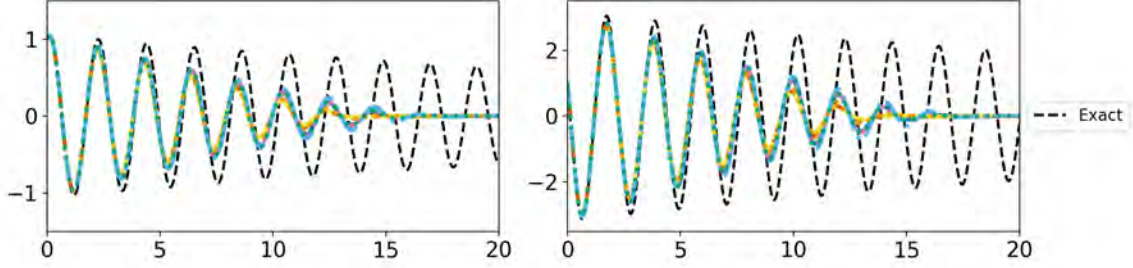


Figure 6.1: PINN results s_1 (left) and s_2 (right) as a function of time for the pendulum problem for ten random initial seedings. In each case, the PINN solution decays and does not agree well with the exact solution.

6.2 Methods

In this section, we describe multifidelity PINNs and stacking PINNs, and their extensions to DeepONets. The preliminaries on PINNs and DeepONets were covered in Sections 7.2.1.1 and 5.2, respectively.

6.2.1 Multifidelity PINNs

One method used to improve the training of PINNs is multifidelity learning, where physics-informed cases can be augmented by a small amount of data or additional physics knowledge to increase the accuracy of the final solution. Many approaches and architectures for multifidelity PINNs have been proposed for a wide range of applications, e.g., [95, 105, 106, 107, 108, 109]. In this section we summarize the approach taken in this paper, based on the approach in [95].

A multifidelity PINN consists of two neural networks, which are trained simultaneously to learn the nonlinear, $\mathcal{F}_{nl}$, and linear, $\mathcal{F}_l$, correlations between the low-fidelity approximation and the high-fidelity approximation of the system, as shown in Fig. 6.2. The low-fidelity approximation can be the prediction of a neural network trained with low-fidelity data as in [95]. The output of the multifidelity PINN is the convex

combination of the learned correlation networks

$$\mathcal{F}_{MF}(x, t; \theta) = |\alpha| \mathcal{F}_{nl}(x, t; \theta) + (1 - |\alpha|) \mathcal{F}_l(x, t, \theta), \quad (6.3)$$

where the set of trainable parameters of each network is denoted by θ and α is a trainable coefficient that controls the interplay between the linear and nonlinear components. To enforce learning the linear correlation, $\mathcal{F}_l$ does not use an activation function. The multifidelity loss function is given by:

$$\mathcal{L}_{MF}(\theta) = \lambda_{ic} \mathcal{L}_{ic}(\theta) + \lambda_{bc} \mathcal{L}_{bc}(\theta) + \lambda_r \mathcal{L}_r(\theta) + \alpha^4. \quad (6.4)$$

The penalty term α^4 is chosen so that α is small, and the network predominately learns a linear correlation when possible. By maximizing the linear correlatin, the nonlinear network can be smaller, reducing the number of trainable parameters.

Previous work with multifidelity DNNs and multifidelity PINNs [howard2023continual](#), [meng2019multifidelity](#) have chosen to add a regularization term that penalizes the sum of the squares of the weights and biases of the nonlinear correlation network, instead of the penalty term in Eq. 6.4. The goal of such a term was to prevent overfitting of the nonlinear correlation, as well as enforce learning a linear correlation when possible. In our tests, this regularization did not perform as well, perhaps because we train PINNs without data, so overfitting to data is not a concern. In addition, whether we use the exact same equation and loss function at every stacking level or we allow small modifications between levels, we expect that the linear correlation should be quite strong, in contrast with data-driven multifidelity training which uses low- and high-fidelity datasets for the low- and high-fidelity models.

As a very general concept, multifidelity PINNs are flexible and can use data or physics-informed training for both the low- and high-fidelity approximations. For example, data generated by a low order numerical model could be used as low-fidelity

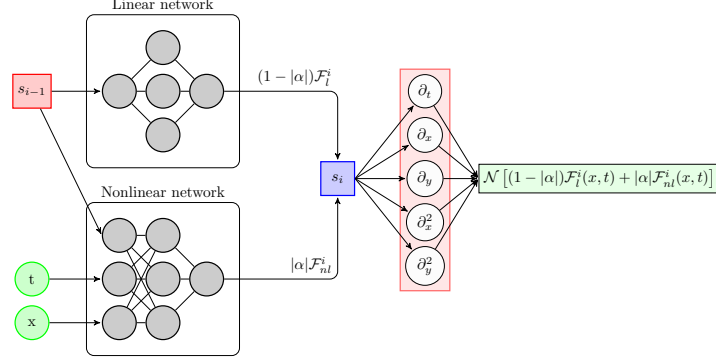


Figure 6.2: Multifidelity physics-informed neural network

data because it will have a high amount of error. Then, physics can be enforced for the high-fidelity approximation using physics-informed training. In another case, a low order approximation of the physics could be enforced for the low-fidelity training, and a high order approximation of the physics could be used for the high-fidelity training. The advantage of multifidelity PINNs is that they allow for combinations of the available knowledge about the problem, producing more accurate and robust approximations than training with the low-fidelity or high-fidelity data or physics alone. We leverage this flexibility and consider the possibility of the low-fidelity component being a multifidelity PINN on its own, and gradually build up a stack of multiple multifidelity PINNs as described in 6.2.2.

6.2.2 Stacking PINNs

We propose the use of *stacking PINNs*, in which multiple multifidelity PINNs are “stacked,” so that the output of each multifidelity PINN is taken as the low-fidelity approximation for a new multifidelity PINN, as illustrated in Fig. 6.3. As we show in this work, stacking PINNs allow for more expressive solutions and can train for cases where single fidelity PINNs and traditional multifidelity PINNs cannot reach a satisfactory solution. In addition, stacking PINNs can use smaller network sizes, reducing the total number of trainable parameters needed to reach a given error.

The process is implemented through the following steps:

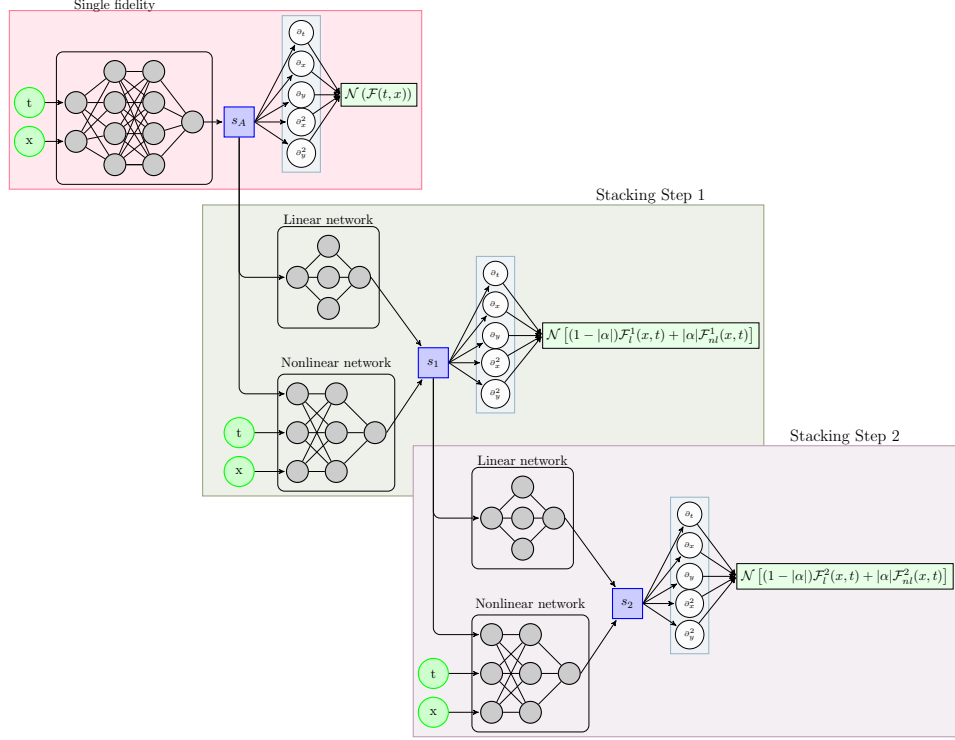


Figure 6.3: Stacking multifidelity physics-informed neural network

1. Step 0: Train a single fidelity PINN to approximate the solution. This network is denoted $\mathcal{F}^0(x, t; \theta^0)$.
2. Stacking steps: for step $i > 0$, train a multifidelity PINN $\mathcal{F}^i(x, t; \theta^i)$ that takes the output from the previous step $\mathcal{F}^{i-1}(x, t; \theta^{i-1})$ as a low fidelity approximation.

One advantage of the stacking method is that we can use information from previously trained levels as a prior to inform the training of the current level. For example, we transfer the weights from previously trained networks to the current network instead of initializing the current network parameters randomly, akin to transfer learning. In notation $\theta_0^i = \theta^{i-1}$ and $\alpha_0^i = \alpha^{i-1}$ for each stacking step $i > 1$, where θ_0^i denotes the initial values of the trainable parameters. We also note that, unlike regular multifidelity PINNs where the low- and high-fidelity networks are often trained simultaneously, we perform the stacking process sequentially. In other words, once

a network is trained at level i , its parameters (θ^i and α^i) are frozen throughout the next stacking steps. This alleviates the computational burden of optimizing a large number of parameters at once. Furthermore, it allows us to progressively increase the number of stacking levels, e.g., using sanity checks to determine whether additional stacking levels are needed, without the need to retrain the whole stack.

6.3 Results with PINNs

We first demonstrate the performance of stacking networks by exploring some cases where PINNs are known to fail to train. In particular, we consider the damped pendulum problem in Sec. 6.3.1, followed by a toy multiscale problem in Sec. 6.3.2. After that, we thoroughly investigate solving the 1D wave equation by the means of PINNs in Sec. 6.3.3. In particular, we consider the cases where the same equation is enforced at all stacking levels as well as where different equations are imposed at different levels to further illustrate the flexibility of the stacking idea.

6.3.1 Pendulum

To show how the method proposed in Sec. 6.2.2 works, we consider an undamped pendulum. The system is governed by a system of two ODEs for $t \in [0, T]$

$$\frac{ds_1}{dt} = s_2, \tag{6.5}$$

$$\frac{ds_2}{dt} = -\frac{b}{m}s_2 - \frac{g}{L}\sin(s_1), \tag{6.6}$$

$$\tag{6.7}$$

where s_1 represents the position of the pendulum and s_2 its velocity. We choose the initial conditions $s_1(0) = s_2(0) = 1$ and take $m = L = 1, b = 0.05, g = 9.81$ and $T = 20$. When a single fidelity PINN is used to solve this problem, the solution matches the initial conditions but decays to 0 past $t \approx 14$. The zero solution is a

solution to this equation when the initial conditions are also set to 0, so the residual loss term in the PINN formulation is minimized, but the solution is not correct.

To set up the problem, we first train a single fidelity PINN for the pendulum problem, as in Fig. 6.1. Clearly, this single fidelity PINN does not accurately capture the dynamics of the pendulum for long times. For each of the ten random initializations in Fig. 6.1, we iteratively train a stacking PINN for up to ten stacking steps. After each stacking step, the solution gets closer to the exact solution for the pendulum, shown in Fig. 6.6a.

We define the relative ℓ_2 error by

$$error = \frac{\|\mathcal{F}^i(x, t; \theta^i) - s(x, t)\|_2}{\|s(x, t)\|_2}. \quad (6.8)$$

After four to eight stacking steps, the pendulum reaches a fixed value of the relative ℓ_2 error, which does not decrease further with additional stacking PINNs. The relative ℓ_2 errors after each stacking step are shown in Fig. 6.6b. The final relative errors are close to constant, and the initial relative errors are also quite close. Interestingly, each case seems to hit a critical point after which the relative ℓ_2 error begins to decrease rapidly, before reaching and plateauing at the final value. The average training time is 1.613 hours on one NVIDIA P100 GPU.

For the damped pendulum, stacking PINNs allows for accurate solutions up to $T = 20$.

6.3.2 Multiscale problems

We now consider a toy model, inspired by [96]:

$$\frac{ds}{dx} = \omega_1 \cos(\omega_1 x) + \omega_2 \cos(\omega_2 x), \quad (6.9)$$

$$s(0) = 0, \quad (6.10)$$

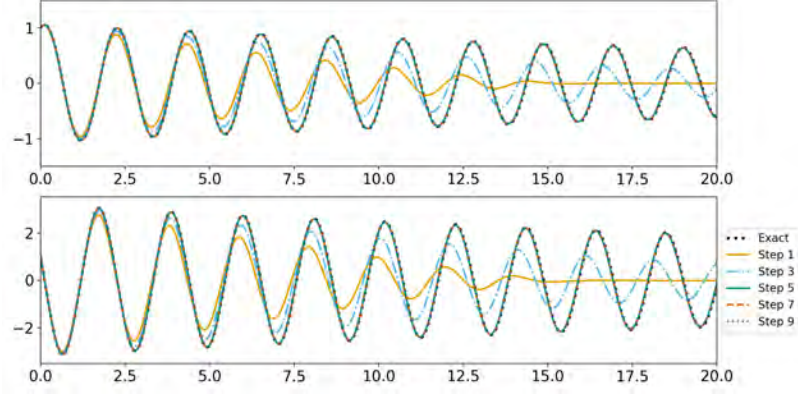


Figure 6.4: Stacking PINN results for an illustrative example of s_1 (top) and s_2 (bottom) as a function of time for the pendulum problem up to nine stacking steps.

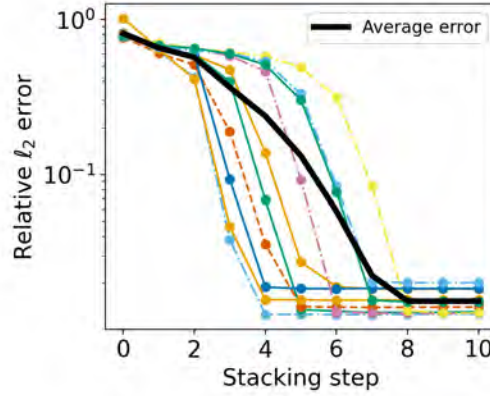


Figure 6.5: Pendulum relative ℓ_2 training errors after each stacking step for ten initial seedings.

Figure 6.6: Stacking PINN training for the pendulum problem.

on domain $\Omega = [0, 20]$ with $\omega_1 = 1$ and $\omega_2 = 15$. The exact solution for this problem is $s(x) = \sin(\omega_1 x) + \sin(\omega_2 x)$.

We consider several cases. This problem is possible to solve with a standard single fidelity (SF) PINN, however, it requires a very large network. In Table 6.1, we show that the stacking PINN can reach a relative ℓ_2 error lower than the best single fidelity PINN with just three stacking levels, and with less than one third the number of trainable parameters needed in the single fidelity case. Importantly, the stacking PINN continues to improve with more multifidelity steps, shown in Fig. 6.10, and after ten stacking levels it reaches a relative error an order of magnitude lower than

the best SF PINN, even though it has 61% of the trainable parameters.

The advantage of stacking PINNs for this problem is thus twofold. First, the stacking networks can reach a similar relative ℓ_2 error to the relative ℓ_2 error from a single fidelity network, but with a significantly smaller number of trainable parameters. For large applications pushing the memory limits of contemporary GPUs, this can offer an advantage where the network can be trained with a series of consecutive smaller networks, each of which is easier to train and does not have memory limitations. The cost of such training is, of course, the necessary wall clock time for sequential training of the networks. The second advantage of the stacking PINNs is that by adding additional levels, they can reach a significantly smaller relative ℓ_2 error than the single fidelity PINN, while still possibly having fewer trainable parameters.

Table 6.1: Relative ℓ_2 errors and network sizes for the multiscale problem in Sec. 6.3.2. All stacking PINNs begin with a single fidelity network with size 3×32 . For the stacking cases, we report the first stacking level that has a relative ℓ_2 error less than the minimum single fidelity error, and the relative ℓ_2 error after the tenth stacking level.

Method	Network size	Trainable parameters	Final relative error	Run time (h)
Single fidelity	3×32	2209	1.3419	0.169
Single fidelity	4×64	12673	0.6482	0.183
Single fidelity	4×128	49921	0.1123	0.185
Single fidelity	5×64	16833	0.0265	0.196
Stacking	$4 \times 16, 1 \times 5, 3$ levels	4900	0.0249	0.450
Stacking	$4 \times 16, 1 \times 5, 10$ levels	11179	0.0061	1.579

6.3.3 Wave equation

The wave equation is given by

$$s_{tt}(x, t) - c^2 s_{xx}(x, t) = 0, \quad x \in (0, 1) \times (0, 1), \quad (6.11)$$

where c is a scalar parameter for wave speed. We consider the following boundary and initial conditions:

$$s(0, t) = s(1, t) = 0, \quad t \in [0, 1], \quad (6.12)$$

$$s(x, 0) = \sin(\pi x) + 0.5 \sin(4\pi x), \quad x \in [0, 1], \quad (6.13)$$

$$s_t(x, 0) = 0, \quad x \in [0, 1]. \quad (6.14)$$

The exact solution for the wave equation is given by

$$s(x, t) = \sin(\pi x) \cos(c\pi t) + 0.5 \sin(4\pi x) \cos(4c\pi t). \quad (6.15)$$

We use the wave equation to illustrate another key advantage of the stacking PINN framework. While up to this point we have trained with the same equations for each stacking level, it is possible, and in some cases advantageous, to change the equations for each stacking level to make it easier to train the early levels. We have found that a well-trained low-fidelity model, possibly for a slightly different and simpler equation, can produce more robust results than training with the same equation for all stacking levels. One advantage of training with an equation that is easier to learn with the single fidelity network is that one can generally use a smaller network, limiting the number of training parameters.

For the wave equation, we consider four cases, varying the parameter c in Eq. 6.11 during each stacking level according to Table 6.2. When c is smaller, the PINN is much easier to train. Thus, by starting c small and increasing to our target value of $c = 2$ or $c = 4$, we hope to begin with a very accurate prediction, for a slightly modified equation, and then correct with the prediction for the correct equation.

In Fig. 6.11, we show the training for Case 1, where c is held fixed at $c = 2$. The final relative ℓ_2 error is 2.913×10^{-3} after 14 stacking levels. When compared with Case 3 where c is varied, shown in Fig. 6.12, Case 1 reaches a lower relative ℓ_2 error

Table 6.2: Schedule for increasing c with stacking steps for each of the cases considered for the wave equation case. The right column has the relative ℓ_2 error for the final stacking step. Note that the relative error is calculated with respect to the exact solution with $c = 2$ for cases 1, 2, and 3, and $c = 4$ for Case 4.

Case	Schedule for c	Error
Case 1	$c = 2$ for all stacking steps and the single fidelity network	2.913×10^{-3}
Case 2	$c = [1, 1.25, 1.5, 1.75, 2]$, and then $c = 2$ for all remaining stacking steps	4.274×10^{-3}
Case 3	$c = 1$ for the single fidelity level and $c = 2$ for all stacking steps	3.805×10^{-3}
Case 4	$c = [1.0, 2.0, 2.0, 3.0, 3.0, 3.0, 3.0]$, and then $c = 4$ for remaining stacking steps	8.396×10^{-3}
Case 5	$c = 4$ for all stacking steps and the single fidelity network	2.630×10^{-2}
Single fidelity	$c = 2$	4.129×10^{-2}
Single fidelity	$c = 4$	1.379

after many stacking layers, see Fig. 6.13. However, Case 3 converges to a smaller relative ℓ_2 error in early stacking steps. Depending on the performance goal, Case 3 may represent a need for less training through using fewer stacking levels, while still achieving accurate results. The final relative ℓ_2 error for Case 3 is 3.805×10^{-3} . Case 2 reaches a final relative ℓ_2 error of 4.274×10^{-3} . The average training time is 2.094 hours for 12 stacking steps and 5.010 hours for 20 stacking steps on one NVIDIA P100 GPU.

Cases 1, 2, and 3 have relatively similar training performance as shown by similar values of the final relative ℓ_2 errors, and all result in accurate predictions for the wave equation with a one order of magnitude reduction in the error compared to the single fidelity PINN. Cases 4 and 5 represent pushing to accurately predict a more difficult problem, with $c = 4$. In Case 4, c is gradually increased, while in Case 5 $c = 4$ is fixed. In both cases, the stacking PINN is able to capture the fine-scale oscillations of this problem. However, Case 4 is more accurate, with a relative ℓ_2 error of 8.396×10^{-3} for the final stacking layer. This relative ℓ_2 error is three orders of magnitude smaller

than the single fidelity training for the wave equation with $c = 4$, and smaller than the Case 5 relative ℓ_2 error.

6.3.4 2D wave equation

To show a more complicated example, we also consider the two-dimensional time-dependent wave equation, given by

$$s_{tt}(x, y, t) - c^2 s_{xx}(x, y, t) - c^2 s_{yy}(x, y, t) = 0, \quad (x, y, t) \in [0, 1] \times [0, 1] \times [0, 1], \quad (6.16)$$

where $c = 2$ is a scalar parameter for the wave speed. We consider the following boundary and initial conditions:

$$s(0, y, t) = s(1, y, t) = 0, \quad y \in [0, 1], t \in [0, 1], \quad (6.17)$$

$$s(x, 0, t) = s(x, 1, t) = 0, \quad x \in [0, 1], t \in [0, 1], \quad (6.18)$$

$$s(x, y, 0) = \sin(2\pi x) \sin(2\pi y), \quad (x, y) \in [0, 1] \times [0, 1], \quad (6.19)$$

$$s_t(x, y, 0) = 0, \quad (x, y) \in [0, 1] \times [0, 1], \quad (6.20)$$

The exact solution for the 2D wave equation is given by

$$s(x, y, t) = \sin(2\pi x) \sin(2\pi y) \cos(2\sqrt{2}c\pi t). \quad (6.21)$$

The stacking PINN is able to accurately capture the solution over the full time domain after several stacking levels, shown in Fig. 6.16. The relative ℓ_2 error converges to a final value of 0.0017, as shown in Fig. 6.17. The training time is 4.074 hours on one NVIDIA P100 GPU.

6.4 Conclusions

While PINNs showed promise for solving systems without the need for data, their inability to train for chaotic systems demonstrated the need for the iterative multifidelity approach used in this chapter. This highlights a case in which a classical technique (namely iterative solving) can be applied to a modern method to improve solution accuracy. The following chapter has a similar format but explores a different failure mode for PINNs. Specifically, we explore how domain decomposition (a classical numerical technique) can be used to improve the training of machine learning methods for problems whose solutions exhibit complex geometries and high frequencies.

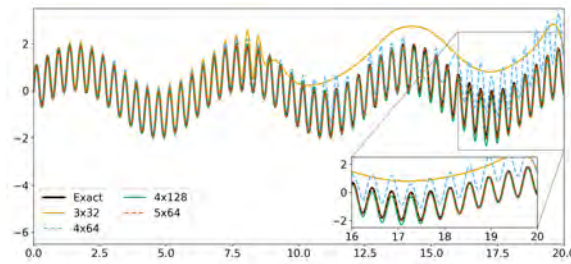


Figure 6.7: Single fidelity

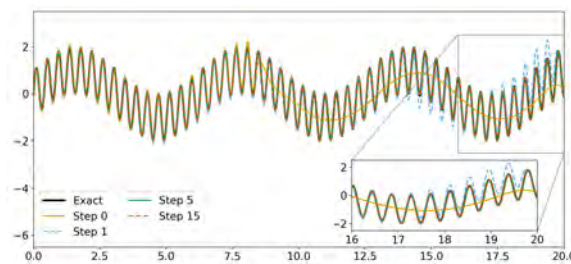


Figure 6.8: Stacking PINN

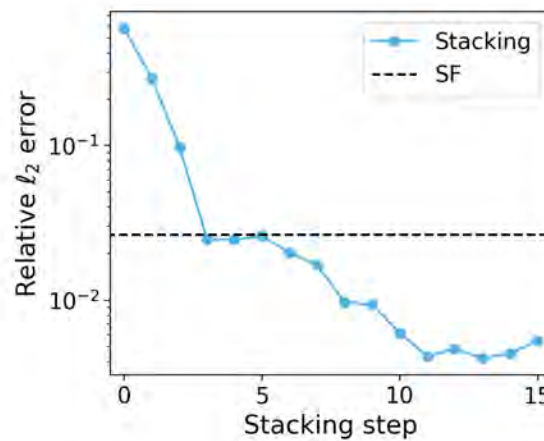
Figure 6.9: Relative ℓ_2 errors

Figure 6.10: Results for the multiscale problem. (a) Single fidelity results for a variety of network sizes. (b) Multifidelity stacking results. (c) Relative ℓ_2 errors for the stacking PINN. The black dashed line in (c) is the lowest error from the single fidelity training for the 5×64 network. For the stacking PINN, step 0 is the single fidelity step and step 1 is the first multifidelity step.

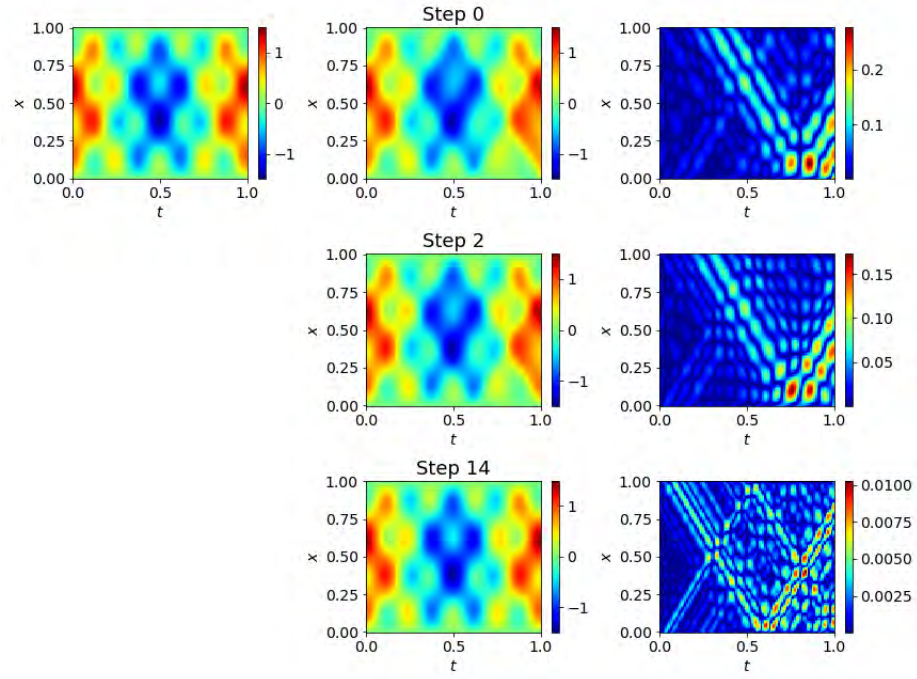


Figure 6.11: Results for the stacking PINN for the wave equation for Case 1. The top left figure is the exact solution. The results from training steps 0, 2, and 14 are in the middle column. Their respective absolute errors are in the right column.

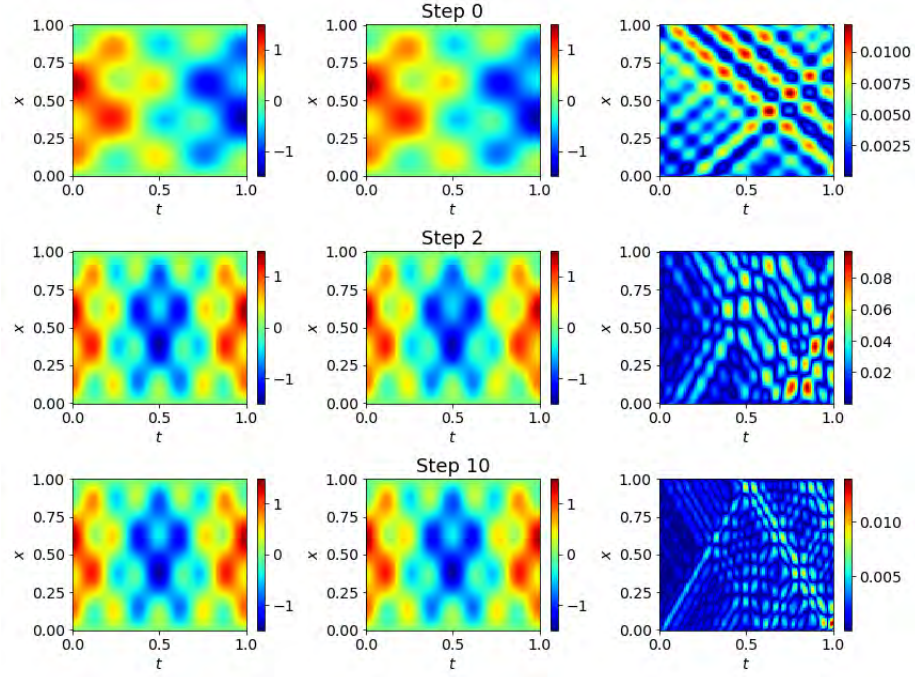


Figure 6.12: Results for the stacking PINN for the wave equation for Case 3. The left column has the exact solution for the value of c used for a given stacking step. The results from training steps 0, 2, and 10 are in the middle column. Their respective absolute errors are in the right column.

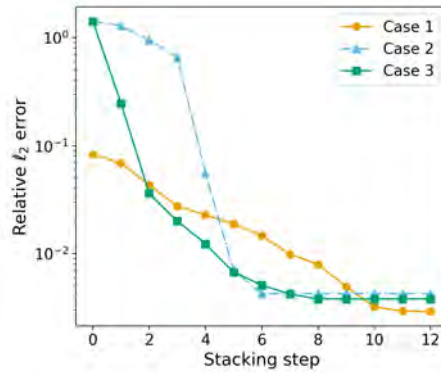
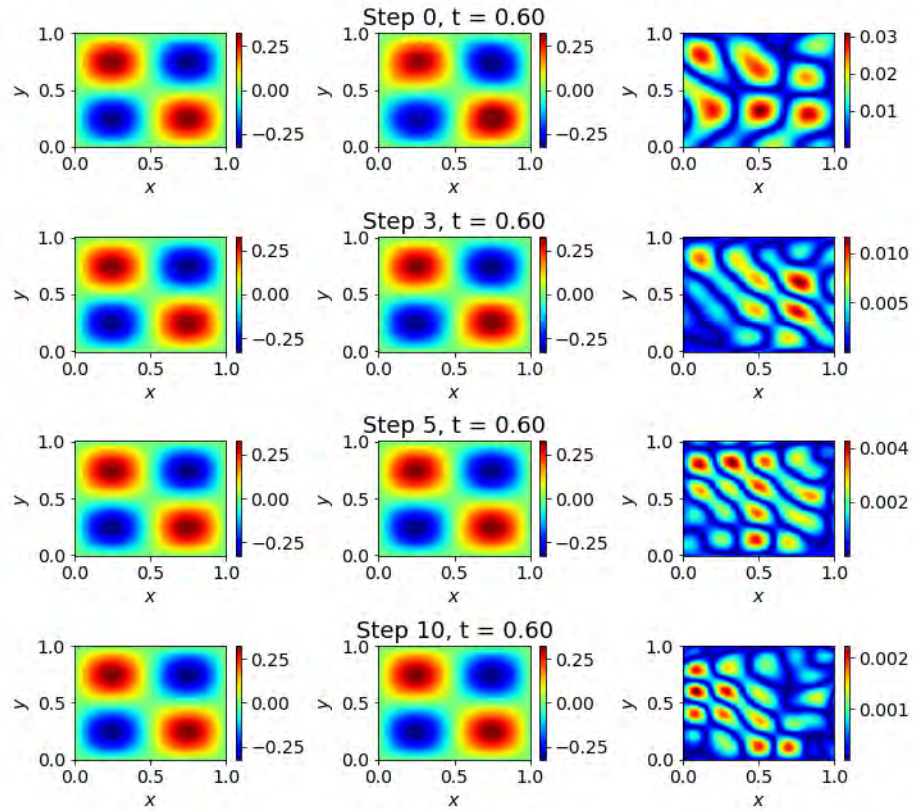
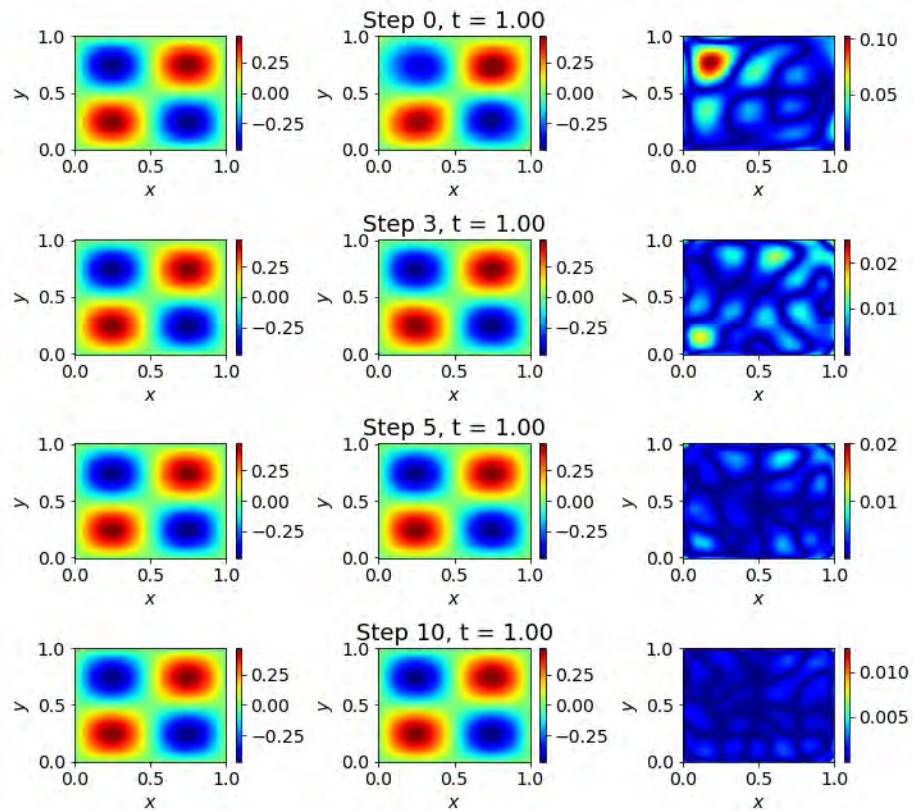


Figure 6.13: Relative ℓ_2 errors for the stacking PINNs for the wave equations for Case 1 (circles), Case 2 (triangles), and Case 3 (squares). The relative ℓ_2 is calculated using the exact solution for the wave equation with $c = 2$.



0.48

Figure 6.14: $t = 0.6$ 

0.48

Figure 6.15: $t = 1$ Figure 6.16: Results for the 2D wave equation at two different times, $t = 0.6$ (a) and $t = 1$ (b).

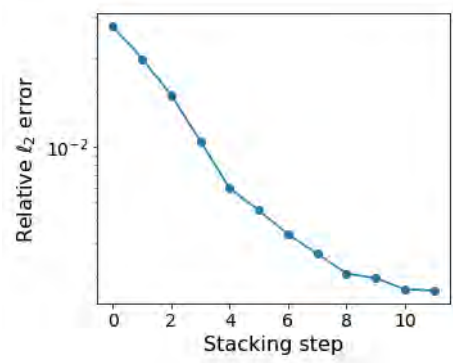


Figure 6.17: Relative ℓ_2 error for the 2D wave equation calculated at 125,000 evenly spaced grid points in the training domain.

CHAPTER 7: DOMAIN DECOMPOSITION FOR SCIENTIFIC MACHINE LEARNING METHODS

This chapter is based on the work in *Finite basis Kolmogorov-Arnold Networks: domain decomposition for data-driven and physics-informed problems* by Amanda Howard, Bruno Jacob, Sarah Helfert, Alexander Heinlein and Panos Stinis and an upcoming manuscript titled *Adaptive Domain Decomposition for finite basis Physics-informed Neural Networks* by Sarah Helfert, Amanda Howard, Wanxin Chen, Panos Stinis, and Alexander Heinlein. These works highlight the benefits of blending classical numerical approaches with scientific machine learning methods. In particular, we highlight the application of domain decomposition, a classical approach, techniques to improve the performance of PINNs and KANs. Domain decomposition is a technique used for problems with complex problem domains, in which separate models are trained on user-defined subdivisions of the entire domain. This work shows that the numerical accuracy of PINNs and KANs can be improved for problems exhibiting high frequency features and complex geometries using domain decomposition.

This chapter begins with a presentation of FBPINNs in Section 7.0.0.1, the domain decomposition approach which is designed to apply a classical domain decomposition approach to SciML methods. Then, this chapter is split into two sections: Section 7.1 focuses on a novel approach for domain decomposition KANs, and Section 7.2 presents a novel approach for adaptively selecting the domain decomposition subdivisions for PINNs. Both works are built upon the framework of Finite basis PINNs (FBPINNs), presented in [96].

Domain decomposition techniques can improve training in such cases [110, 111, 112, 113, 96]; for a general overview of domain decomposition approaches in scientific

machine learning, see also [114, 115]. Rather than training the network over the entire domain, there are many approaches for breaking up the domain into subdomains and training networks on each subdomain. A popular choice for domain decomposition applied to PINNs is extended PINNs (XPINNs) introduced in [110], with a parallel version presented in [116], and extended in [111, 117]. XPINNs employ a separate subnetwork for each subdomain, which are stitched together across boundaries and trained in parallel. Another class of domain decomposition techniques for PINNs consists of time marching methods, where, in general, a network is initially trained for a small subdomain, and then the subdomain is slowly expanded until it contains the entire domain. Methods in this class include backward-compatible PINNs, which consist of a single network shared by all subdomains [112]. Also similar is the time marching approach in [113]. Finally, a particularly successful technique is finite basis PINNs (FBPINNs) [96, 118, 119, 120], which use partition of unity functions to weight the output of neural networks in each domain; these have also been extended to randomized neural networks [121, 122]. In comparison to other techniques, FBPINNs offer simplicity because the boundary conditions between adjacent domains do not have to be enforced via additional penalty terms in the loss function. Section 7.1 describes how this approach can be extended to KANs to produce Finite basis KANs (FBKANs), and Section 7.2 describes an approach to adaptively select the partitions of the subdomains on which to train an FBPINN.

7.0.0.1 Finite basis PINNs

FBPINNs [123], use many small neural networks on overlapping subdomains whose outputs are summed to represent the solution on the whole problem domain. They have shown superior performance in cases where a solution exhibits high frequencies or multi-scale features.

Given a problem domain $\Omega \in \mathbb{R}^d$, we divide Ω into L overlapping subdomains Ω_i . Each subdomain is assigned a separate neural network $\mathcal{NN}_i(\mathbf{x}; \theta_i)$, and smooth,

differentiable window functions $w_i(\mathbf{x})$ are used to combine their outputs into a global solution. The approximate solution $\hat{u}(\mathbf{x}; \theta)$ is given by

$$\hat{u}(\mathbf{x}; \theta) = \mathcal{C}[\overline{\mathcal{NN}}(\mathbf{x}; \theta)], \quad (7.1)$$

where

$$\overline{\mathcal{NN}}(\mathbf{x}; \theta) = \sum_{i=1}^L w_i(\mathbf{x}) \cdot \mathcal{NN}_i(\text{norm}_i(\mathbf{x}); \theta_i). \quad (7.2)$$

Here, $\mathcal{NN}_i(\text{norm}_i(\mathbf{x}); \theta_i)$ is the neural network applied to domain Ω_i , $\text{norm}_i(\mathbf{x})$ denotes the normalization applied to inputs of the subdomain, $\mathcal{C}$ is a constraining operator that enforces boundary or initial conditions on the solution ansatz (as in Section 7.2.1.1), and $w_i(\mathbf{x})$ is a smooth window function that confines each network to its subdomain. In particular, these window functions ensure each neural network contributes only within its respective subdomain, with its influence rapidly decaying to zero outside that region. This means that the neural network only has to be evaluated on collocation points within its respective subdomain $x \in \Omega_i$. The window functions satisfy a partition of unity (POU) constructed with the expression

$$w_i = \frac{\hat{w}_i}{\sum_{i=1}^L \hat{w}_i} \quad \text{where} \quad \text{supp}(\hat{w}_i) \subset \overline{\Omega_i}, \quad (7.3)$$

where $\hat{w}_i$ denotes the unnormalized window function. Here, L represents the total number of subdomains. The unnormalized window functions can be, for instance, chosen as:

$$\hat{w}_i(\mathbf{x}) = \prod_{j=1}^d \psi\left(\frac{x^j - c^j}{h_i^j}\right), \quad \text{where} \quad \psi(t) = \begin{cases} (1 + \cos(\pi t))^2, & |t| \leq 1, \\ 0, & \text{otherwise} \end{cases} \quad (7.4)$$

where $\mathbf{c} \in \mathbb{R}^d$ are the centers of each subdomains and $\mathbf{h} \in \mathbb{R}^d$ are the half-width of its support in each dimension.

For uniformly spaced subdomains with center spacing $d^j = c_{i+1}^j - c_i^j$, we parameterize the half-width as

$$h_i^j = \frac{d^j + \delta}{2}, \quad (7.5)$$

where δ denotes the total overlap width between adjacent subdomains.

The same approach used for PINNs is then used to train an FBPINN. In particular, given a set of collocation points $\{\mathbf{x}_i\}_{i=1}^N$, the loss function for the FBPINN is

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N (\mathcal{D}\hat{u}(\mathbf{x}_i; \theta) - \mathbf{f}(\mathbf{x}_i))^2 \quad (7.6)$$

An example of an overlapping domain decomposition and of a single partition of unity function with $d = 2$ and $L = 4$ are shown in fig:POU_{2d}.

Then, the FBPINN architecture reads

$$f_\theta(\mathbf{x}) = \sum_{j=1}^L \omega_j(x) f_j(\mathbf{x}; \theta^j), \quad (7.7)$$

where $f_j(\cdot; \theta^j)$ is the neural network with parameters θ^j that corresponds to the subdomain Ω_j ; it is localized to Ω_j by multiplication with the partition of unity function ω_j , which is zero outside of Ω_j . The FBPINN model is trained in the same way as the PINN model, i.e., using initial, boundary, and residual loss terms; cf. sec:pinns. Note that, in the original FBPINNs work [96], hard enforcement of initial and boundary conditions is employed, such that only the residual loss term remains. Moreover, in [119] the approach has been extended to multilevel domain decompositions, in [120] this domain decomposition approach has been applied to long-range time-dependent problems, and in [121, 122] it has been extended to randomized neural networks.

7.1 Finite basis Kolmogorov-Arnold Networks: domain decomposition for data-driven and physics-informed problems

The methods discussed in the previous sections, namely PINNs and DeepONets, are developed using the theory of multilayer perceptrons. In this section, we focus on an alternative model architecture, based on the Kolmogorov-Arnold representation theorem [124], which has recently been introduced in [90]. KANs offer advantages over MLPs in some contexts, such as for continual learning and noisy data. While MLPs use trainable weights and biases on edges with fixed activation functions, KANs employ trainable activation functions represented by splines. This switch may be beneficial because splines are easy to adjust locally and adapt to different resolutions. KANs have also been shown to provide increased interpretability over an MLP. Consequently, it is essential to provide a thorough analysis of the performance and applicability of KANs. An active area of research involves exploring how to utilize a KAN effectively and discerning in which cases KANs might be preferred over classical MLPs. In this work, we develop an architecture for KANs based on overlapping domain decompositions, which allows for more accurate solutions for complex, and particularly multiscale, problems.

Since the publication of [90], numerous variations of KANs have been released, including fractional KANs [125], deep operator KANs [126], physics-informed KANs (PI-KANs)[127], KAN-informed neural networks (KINNs)[128], temporal KANs [129], graph KANs [130, 131, 132], Chebyshev KANs (cKANs) [133], convolutional KANs [134], ReLU-KANs [135], among others. KANs have been applied to satellite image classification [136], abnormality detection [137], and computer vision [138]. Although these results are promising, many open questions remain about how KANs can be best utilized and optimized. Many of the recent variations employ alternative functions to the B-spline parametrization, while others extend KANs through well-developed methods from the field of machine learning, such as physics-informed neural networks

(PINNs) [8], deep operator networks (DeepONets) [87], and recurrent neural networks. Because KANs still follow the general network architecture of an MLP, aside from the switch between the edges and neurons, it is often straightforward to apply such techniques to KANs.

7.1.1 Methods

The formulation of KANs are detailed in Section 5.3 and the formulation of FBPINNs are detailed in Section 7.0.0.1. In this section, we describe the method for applying the FBPINN framework to improve the training of KANs as well, in the form of Finite Basis KANs (FBKANs).

7.1.1.1 Finite basis Kolmogorov–Arnold networks

In finite basis Kolmogorov–Arnold networks (FBKANs), we replace the NN used in FBPINNs with a KAN. The function approximation in eq:KAN then becomes

$$\begin{aligned} f(\mathbf{x}) &\approx \sum_{j=1}^L \omega_j(x) \left[\sum_{i_{n_l-1}=1}^{m_{n_l-1}} \varphi_{n_l-1, i_{n_l}, i_{n_l-1}}^j \left(\sum_{i_{n_l-2}=1}^{m_{n_l-2}} \cdots \left(\sum_{i_1=1}^{m_1} \varphi_{1, i_2, i_1}^j \left(\sum_{i_0=1}^{m_0} \varphi_{0, i_1, i_0}^j(x_{i_0}) \right) \right) \cdots \right) \right] \\ &= \sum_{j=1}^L \omega_j(x) \mathcal{K}_j(\mathbf{x}; \theta^j), \end{aligned} \quad (7.8)$$

where $\mathcal{K}_j(\mathbf{x}; \theta^j)$ denotes the j^{th} KAN with trainable parameters θ^j and ω_j denotes the corresponding finite basis partition of unity function introduced in sec:fbpinns. FBKANs are trained to minimize the PINN loss function

$$\mathcal{L}(\theta) = \lambda_{ic} \mathcal{L}_{ic}(\theta) + \lambda_{bc} \mathcal{L}_{bc}(\theta) + \lambda_r \mathcal{L}_r(\theta) + \lambda_{data} \mathcal{L}_{data}(\theta), \quad (7.9)$$

composed of

$$\begin{aligned}
\mathcal{L}_{ic}(\theta) &= \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} \left(\sum_{j=1}^L \omega_j(\mathbf{x}) \mathcal{K}_j(\mathbf{x}_{ic}^i; \theta^j) - u(\mathbf{x}_{ic}^i) \right)^2, \\
\mathcal{L}_{bc}(\theta) &= \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} \left(\sum_{j=1}^L \omega_j(\mathbf{x}) \mathcal{K}_j(\mathbf{x}_{bc}^i; \theta^j) - g(\mathbf{x}_{bc}^i, t_{bc}^i) \right)^2, \\
\mathcal{L}_r(\theta) &= \frac{1}{N_r} \sum_{i=1}^{N_r} \left(\frac{\partial}{\partial t} \left[\sum_{j=1}^L \omega_j(\mathbf{x}) \mathcal{K}_j(\mathbf{x}_r^i; \theta^j) \right] + \mathcal{N}_{\mathbf{x}} \left[\sum_{j=1}^L \omega_j(\mathbf{x}) \mathcal{K}_j(\mathbf{x}_r^i; \theta^j) \right] \right)^2, \\
\mathcal{L}_{data}(\theta) &= \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} \left(\sum_{j=1}^L \omega_j(\mathbf{x}) \mathcal{K}_j(\mathbf{x}_{data}^i; \theta^j) - f(\mathbf{x}_{data}^i) \right)^2,
\end{aligned}$$

using the FBKAN model eq:FBKAN. Here, $\theta = \{\theta^j\}_{j=1}^L$ is the set of trainable parameters, and the loss function $\mathcal{L}$ contains both a term for data-driven training $\mathcal{L}_{data}$ and terms for physics-informed training $\mathcal{L}_{ic}$, $\mathcal{L}_{bc}$, and $\mathcal{L}_r$. In this way, FBKANs are adaptable to given problem characteristics.

For FBKANs, the grid of each local KAN $\mathcal{K}_j$ is defined separately on each subdomain Ω^j . To do so, we densely sample points $x \in \Omega$ and compute the partition of unity function $\omega_j(x)$. We then take the boundaries of each subdomain based on the partition of unity functions as

$$\begin{aligned}
a^j &= \min_{\substack{x \text{ such that} \\ \omega^j(x) > 10^{-4}}} x, \\
b^j &= \max_{\substack{x \text{ such that} \\ \omega^j(x) > 10^{-4}}} x.
\end{aligned}$$

Hence, we obtain a custom grid for each subdomain.

We evaluate the model performance using the relative ℓ_2 error, calculated by

$$\frac{\|f(x) - \sum_{j=1}^L \omega_j(x) \mathcal{K}_j(x; \theta^j)\|_2}{\|f(x)\|_2},$$

where f is the ground truth solution.

Our implementation is based on JAX [139], using the `jaxKAN` package [140] for the KAN implementation. For simplicity, we employ the same hyperparameters for local KAN models defined on the subdomains, including the width and depth; this reduces the total number of hyperparameters. However, in general, the parameters for each individual KAN $\mathcal{K}_j$ could be chosen differently.

7.1.2 Data-driven results

In this section, we consider data-driven problems, which are trained without physics meaning that $\lambda_{ic} = \lambda_{bc} = \lambda_r = 0$. These results specifically illustrate how FBKANs can improve predictions for noisy data and multiscale oscillations.

7.1.2.1 Data-driven test 1

In this section, we consider a toy problem of fitting the function

$$f(x) = \exp[\sin(0.3\pi x^2)], \quad (7.10)$$

on $x \in [0, 8]$. This problem is designed to test both the scaling of FBKANs as we increase the number of subdomains, as well as the impact of noisy data on training FBKANs.

7.1.2.2 Scaling of FBKANs

First, we consider clean data, $N_{data} = 1\,200$, which is sampled uniformly from the interval $[0, 8]$. We present results for training on $L = 1, 2, 4, 8, 16$, and 32 subdomains. Note that the function described in 7.10 is highly oscillatory, which makes it difficult to capture its behavior with a small KAN. As shown in fig:Test1a, increasing the number of subdomains from $L = 2$ to $L = 8$ and $L = 32$ significantly decreases the pointwise error in the FBKAN predictions. Doubling the number of subdomains doubles the number of trainable parameters of the FBKAN model, increasing its expressivity. We observe approximately first order convergence of the relative ℓ_2 error for a large

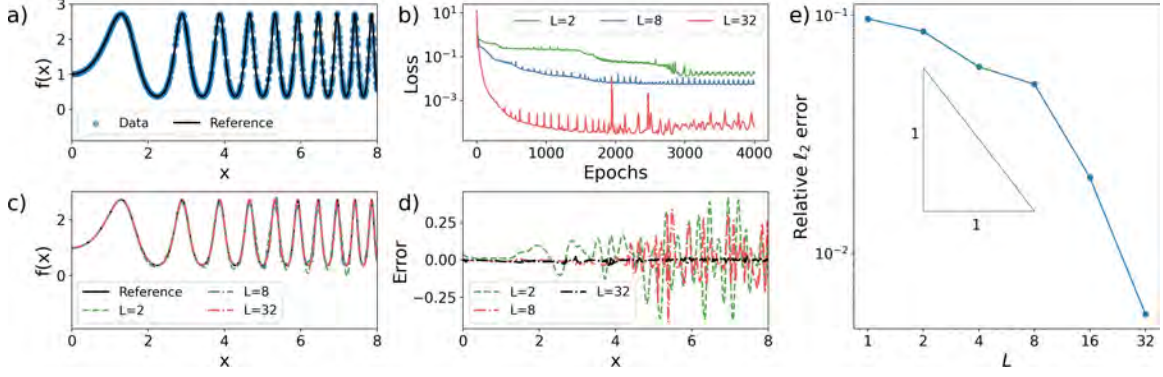


Figure 7.2: Results with $L = 2, 8$, and 32 subdomains for 7.10. (a) Training data and plot of exact $f(x)$. (b) Loss curves (7.9). (c) Plot of the outputs $f(x)$ and $\sum_{j=1}^L \omega_j(x) \mathcal{K}_j(x)$. (d) Pointwise errors $f(x) - \sum_{j=1}^L \omega_j(x) \mathcal{K}_j(x)$. (e) Scaling results for Test 1 with L subdomains.

number of subdomains in 7.2.

7.1.2.3 FBKANs with noisy data

One strength of KANs over MLPs is their increased accuracy for noisy data [90]. We now test FBKANs on noisy training data using four subdomains ($L = 4$). Therefore, we sample 600 training points from a uniform distribution in $[0, 8]$ and evaluate $f(x)$. Then, we add Gaussian white noise with zero mean and varying magnitude to $f(x)$, up to an average relative magnitude of 20 %, as the training data set. The method is then tested on 1000 evenly spaced points in $[0, 8]$ without noise.

The training set and results are shown in fig:Test1b. In the case with the largest noise, with relative noise of 18.1% added to the training set, the KAN yields a relative ℓ_2 error of 0.1404, whereas the FBKAN yields a relative error of 0.0646. For all noise levels tested, the FBKAN consistently has a lower relative error than the plain KAN in fig:Test1b, and the predictions are robust to noisy data.

7.1.2.4 Data-driven test 2

Next, we consider fitting the function

$$f(x, y) = \sin(6\pi x^2) \sin(8\pi y^2), \quad (7.11)$$

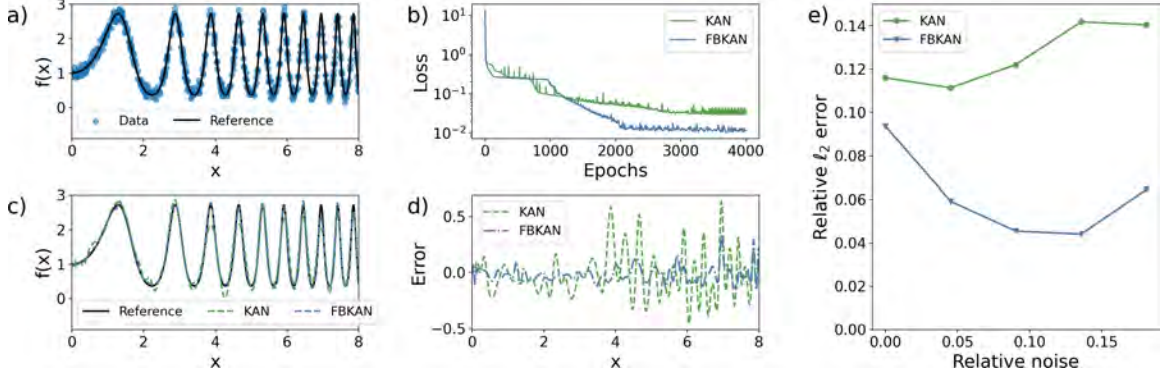


Figure 7.3: Results for Test 1 with noisy training data; (a) Example training data and plot of exact $f(x)$ with 9.6% mean relative noise. (b) Loss curves for an example training with 9.6% mean relative noise. (c) Plot of the outputs $f(x)$ and $\sum_{j=1}^L \omega_j(x) \mathcal{K}_j(x)$ with 9.6% mean relative noise. (d) Pointwise errors $f(x) - \sum_{j=1}^L \omega_j(x) \mathcal{K}_j(x)$ with 9.6% mean relative noise. (e) Relative ℓ_2 error of the KANs and FBKANs with respect to the magnitude of the noise added to the training data.

for $(x, y) \in [0, 1] \times [0, 1]$. This data-driven example exhibits fine-scale oscillations. We test two cases: (1) a KAN and FBKAN with a fixed grid, denoted by KAN-1 / FBKAN-1 and (2) a KAN and FBKAN with grid extension, denoted by KAN-2 / FBKAN-2. The FBKAN has $L = 4$ subdomains. The training set is composed of 10000 points randomly sampled from a uniform distribution.

In both cases, we begin training with $g = 5$ grid points. In the grid extension case, the grid increases every 600 iterations as listed in ??, and the learning rate drops by 20 % each time the grid is updated. As can be observed in fig:Test2a, the final training loss for FBKAN-1 with a fixed grid is approximately the same as the training loss for KAN-2 with the grid extension approach, even though each FBKAN network has six times fewer grid points. This is also reflected in the relative errors reported in tab:test2. Comparing the KAN-1 and FBKAN-1 models in fig:Test2b, KAN-1 struggles to capture the fine-scale features of the data accurately and has a larger relative error. As can be seen in fig:Test2c, KAN-2 and FBKAN-2 both outperform their counterparts with a static grid, but FBKAN-2 is better able to capture the data than KAN-2.

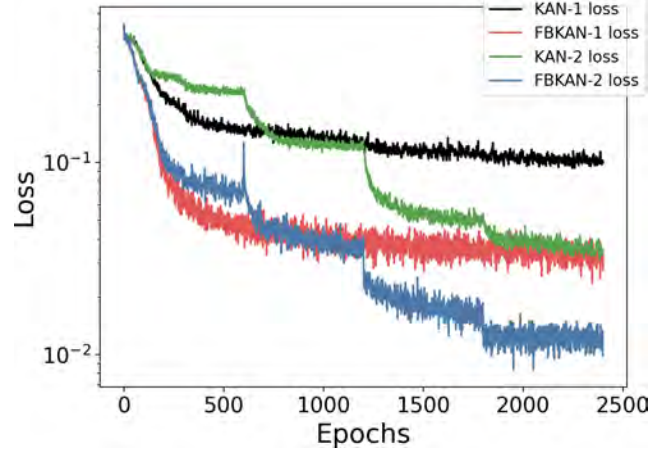


Figure 7.4: Training loss curves for the data-driven test 2.

Table 7.1: Relative ℓ_2 errors for data-driven test 2.

name	grid type	relative error
KAN-1	fixed grid	2.36×10^{-1}
FBKAN-1	fixed grid	7.43×10^{-2}
KAN-2	grid extension	8.10×10^{-2}
FBKAN-2	grid extension	2.27×10^{-2}

7.1.3 Physics-informed results

7.1.3.1 Physics-informed test 1

As the first physics-informed example, we consider the following one-dimensional problem with multiscale features:

$$\begin{aligned} \frac{df}{dx} &= 4 \cos(4x) + 40 \cos(40x), \\ f(0) &= 0, \end{aligned}$$

for $x \in [-4, 4]$. The analytical solution reads

$$f(x) = \sin(4x) + \sin(40x). \quad (7.12)$$

We sample 400 collocation points from a uniform distribution ($N_r = 400$) at every training iteration. We start with a grid with $g = 5$ and extend it by 5 every 1000

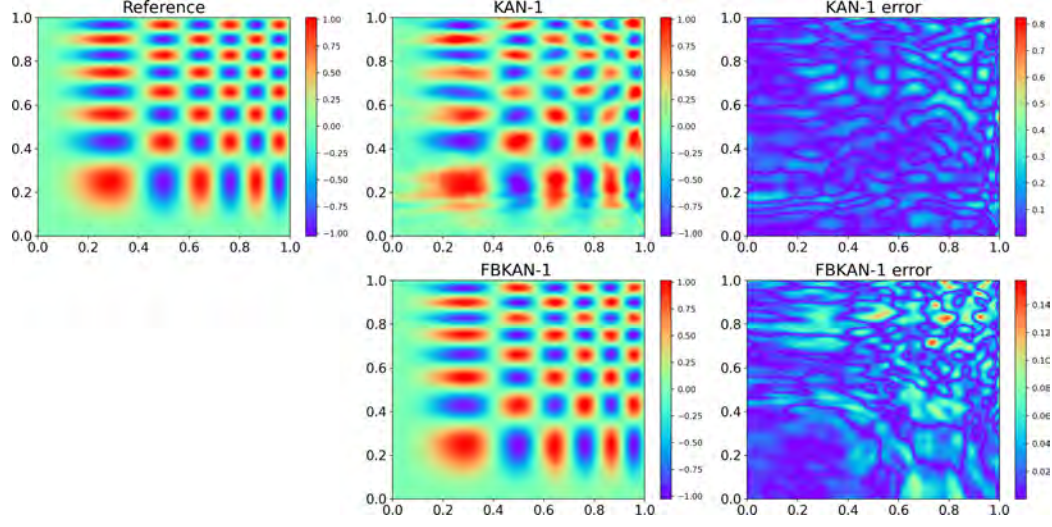


Figure 7.5: Reference solution (left), predicted solutions (middle) and relative errors (right) for KAN-1 and FBKAN-1 for data-driven test 2. Note that the error plots have different color scales.

iterations. The KAN reaches a relative ℓ_2 error of 0.2407 with $g_{final} = 20$, whereas the FBKAN for four subdomains ($L = 4$) reaches a relative ℓ_2 error of 0.0898 for $g_{final} = 20$. For eight subdomains ($L = 8$), the FBKAN reaches a relative error of 0.0369 with $g_{final} = 15$ and 0.0662 with $g_{final} = 10$.

7.1.3.2 Physics-informed test 2

Next, we test the Helmholtz equation

$$\frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial x^2} + k_h^2 f - q(x, y) = 0, \quad (x, y) \in [-1, 1] \times [-1, 1],$$

$$f(-1, y) = f(1, y) = 0, \quad y \in [-1, 1],$$

$$f(x, -1) = f(x, 1) = 0, \quad x \in [-1, 1],$$

with

$$q(x, y) = -(a_1\pi)^2 \sin(a_1\pi x) \sin(a_2\pi y) - (a_2\pi)^2 \sin(a_1\pi x) \sin(a_2\pi y) + k_h^2 \sin(a_1\pi x) \sin(a_2\pi y)$$

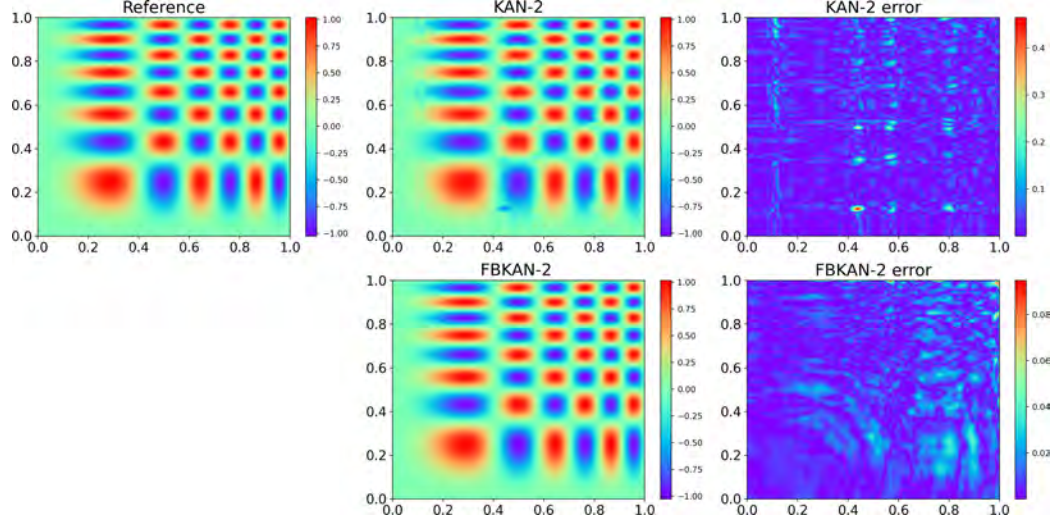


Figure 7.6: Reference solution (left), predicted solutions (middle) and relative errors (right) for KAN-2 and FBKAN-2 for data-driven test 2. Note that the error plots have different color scales.

which is obtained by using the solution

$$f(x, y) = \sin(a_1 \pi x) \sin(a_2 \pi y).$$

In our tests, we vary a_1 and a_2 .

For each choice of (a_1, a_2) we consider three training cases. In the first case, we use higher-order splines with $k = 5$ and a fixed grid with $g = 5$ (denoted by KAN-1 / FBKAN-1). In the second case, we take $k = 3$ and use grid extension to increase g by 5 every 600 iterations, starting with $g = 5$ (denoted by KAN-2 / FBKAN-2). We use one hidden layer with width 10 for both cases. Additionally, we test a third case, denoted by KAN-3 / FBKAN-3, where we consider the same hyperparameters as KAN-1 / FBKAN-1 but with width 5 in the hidden layer. The relative ℓ_2 errors are reported in `tab:error-pi2`.

In all cases, the FBKAN outperforms the corresponding KAN model. The grid extension KAN cases, where we increase the grid size during training, comes at the expense of increasing the computational time by a factor of 1.4 – 1.5 in our tests. For

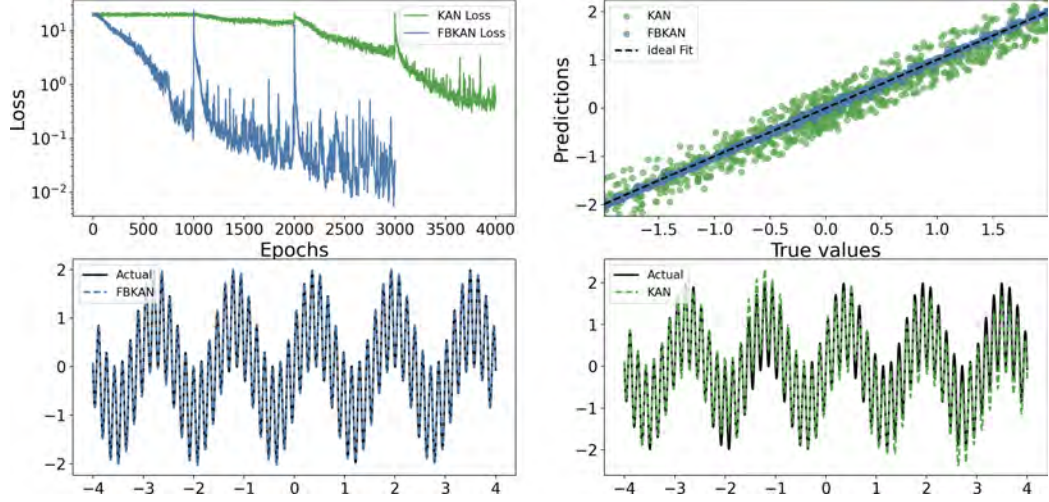


Figure 7.7: Results for physics-informed test 1 with $L = 8$. The FBKAN clearly outperforms the KAN, reaching a lower loss with fewer grid points.

$a_1 = 1$ and $a_2 = 4$, the smaller FBKAN-3 network is sufficient to represent the solution accurately. However, for larger values of a_1 and a_2 , the larger network in FBKAN-1 is necessary. The results for KAN-1 / FBKAN-1 are shown in fig:Test4a,fig:Test4b for $(a_1, a_2) = (1, 4)$ and $(a_1, a_2) = (4, 4)$, respectively. In fig:Test4c, we consider $a_1 = a_2 = 6$ with $L = 4, 9$ and 16 , demonstrating the further refinement possible with additional subdomains.

Table 7.2: Relative ℓ_2 errors for physics-informed test 2.

	$a_1 = 1, a_2 = 4$	$a_1 = 4, a_2 = 4$	$a_1 = 6, a_2 = 6$
KAN-1	0.0259	0.5465	1.1254
FBKAN-1, L=4	0.0102	0.0267	0.1151
FBKAN-1, L=9	0.0213	0.0239	0.0399
FBKAN-1, L=16	0.0037	0.0128	0.0321
KAN-2	0.0180	0.2045	0.5854
FBKAN-2	0.0112	0.0427	0.2272
KAN-3	0.3771	0.5488	1.2825
FBKAN-3	0.0214	0.2760	0.9797

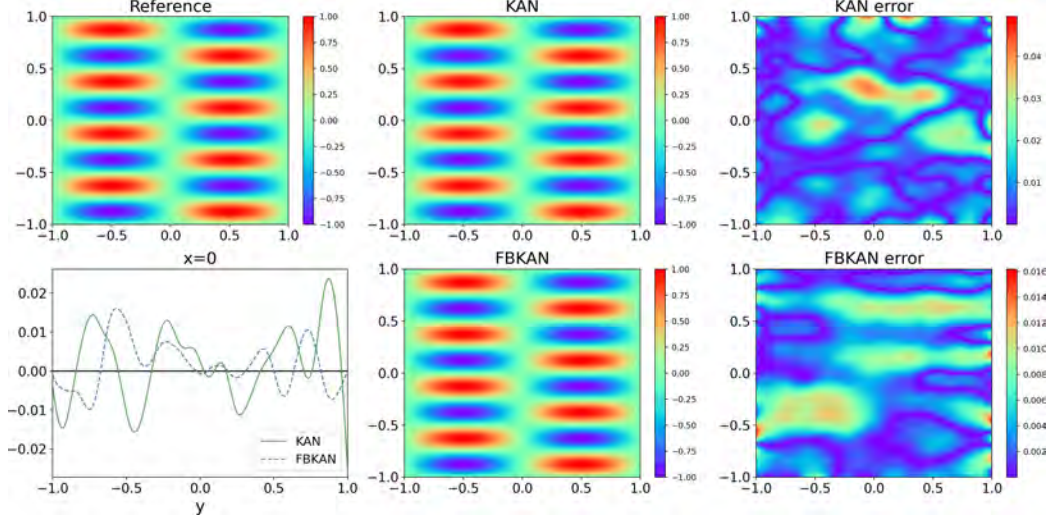


Figure 7.8: Results for physics-informed test 2 with KAN-1 and FBKAN-1, and $a_1 = 1, a_2 = 4, L = 4$. The solution along the line $x = 0$ is given in the bottom left subfigure. Note that the error plots have different color scales.

7.1.3.3 Physics-informed test 3

Finally, we consider the wave equation

$$\frac{\partial^2 f}{\partial t^2} - c^2 \frac{\partial^2 f}{\partial x^2} = 0, \quad (x, t) \in [0, 1] \times [0, 1],$$

$$f(0, t) = 0, \quad t \in [0, 1],$$

$$f(1, t) = 0, \quad t \in [0, 1],$$

$$f(x, 0) = \sin(\pi x) + 0.5 \sin(4\pi x), \quad x \in [0, 1],$$

$$f_t(x, 0) = 0, \quad x \in [0, 1],$$

which has the exact solution

$$f(x, t) = \sin(\pi x) \cos(c\pi t) + 0.5 \sin(4\pi x) \cos(4c\pi t).$$

For $c = \sqrt{2}$, the KAN model has a relative ℓ_2 error of 0.1402, and the FBKAN with $L = 4$ has a relative ℓ_2 error of 0.0153, as illustrated in fig:Test5a. We then consider the more challenging case with $c = 2$, shown in fig:Test5b. The KAN has a relative

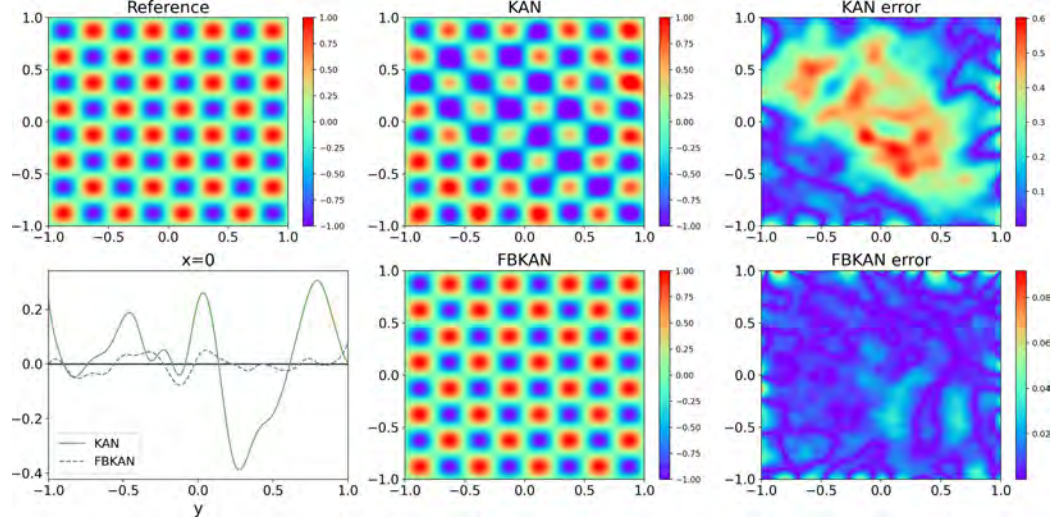


Figure 7.9: Results for physics-informed test 2 with KAN-1 and FBKAN-1, and $a_1 = 4, a_2 = 4, L = 4$. The solution along the line $x = 0$ is given in the bottom left subfigure. Note that the error plots have different color scales.

ℓ_2 error of 0.1778 and the FBKAN with $L = 4$ has a relative ℓ_2 error of 0.0587.

7.2 Adaptive Domain Decomposition for finite basis Physics-informed Neural Networks

While FBPINNs and FBKANs show promise for improving the accuracy of PINNs and KANs for problems with complex geometries, the number and placement of subdomains for domain decomposition is not always obvious. In this section, we develop an adaptive approach for discovering partitions that improve the accuracy of FBPINN training, removing the need for practitioners to define the partitions by hand.

FBPINNs localize the approximation of the solution via overlapping subdomains and window functions that form a partition-of-unity [123]. Here, independent networks are trained on overlapping subdomains, and their outputs are summed via POU window functions to represent the global solution. FBPINNs ensure interface continuity is strictly enforced across subdomain boundaries via the overlapping subdomains through the PINN solution ansatz, thus removing the need for additional terms in the loss function. This approach was extended to use multiple levels of do-

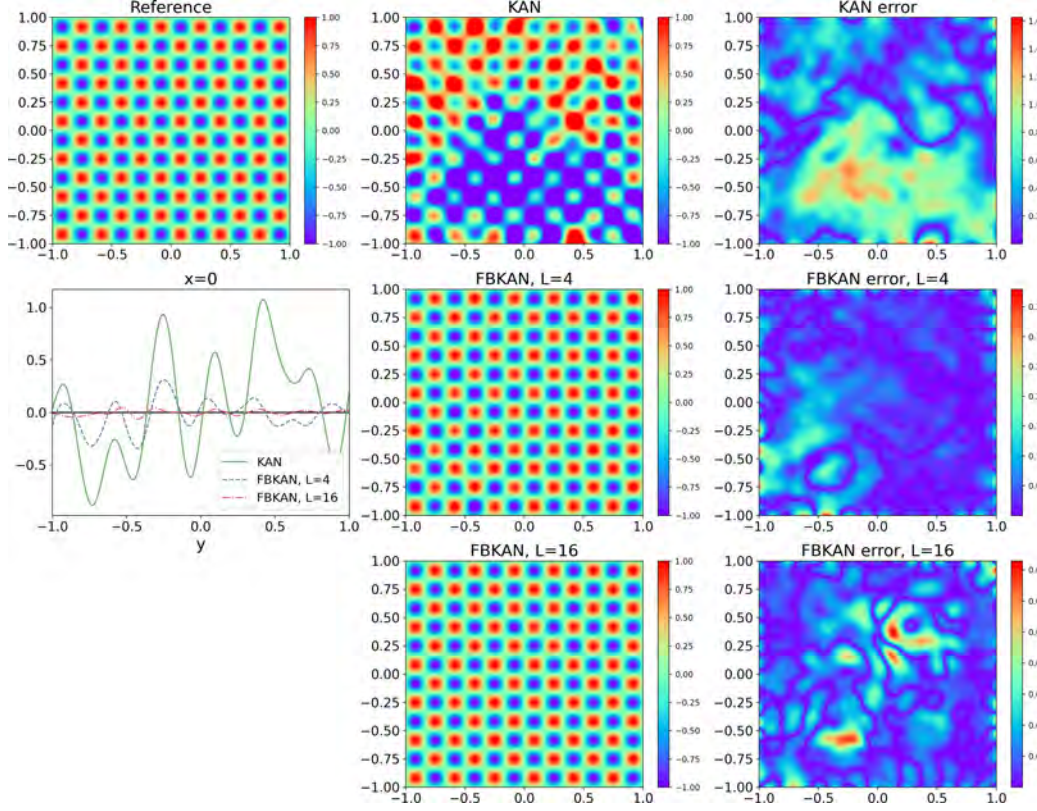


Figure 7.10: Results for physics-informed test 2 with KAN-1 and FBKAN-1 and $a_1 = a_2 = 6$, with $L = 4$ and $L = 16$. The solution along the line $x = 0$ is given in the middle left subfigure. Note that the error plots have different color scales.

main decomposition in the solution ansatz in [141], inspired by multilevel Schwarz methods. Multilevel FBPINNs improve the performance of PINNs for high frequency and multiscale problems.

In this work, we propose a method for adaptively defining the domain decomposition for FBPINNs. This adaptive domain decomposition combines partition-of-unity networks (POUnets, [142]) with FBPINNs to learn the optimal number of and location of partitions of the problem domain. While adaptive approaches to domain decomposition are currently limited in the literature, an adaptive domain decomposition technique was introduced in [143], where a coarse global PINN’s spatial residual profile is used to automatically place subdomain boundaries for localized refinement in regions of high residual. Also, [144] presented the dual adaptive domain

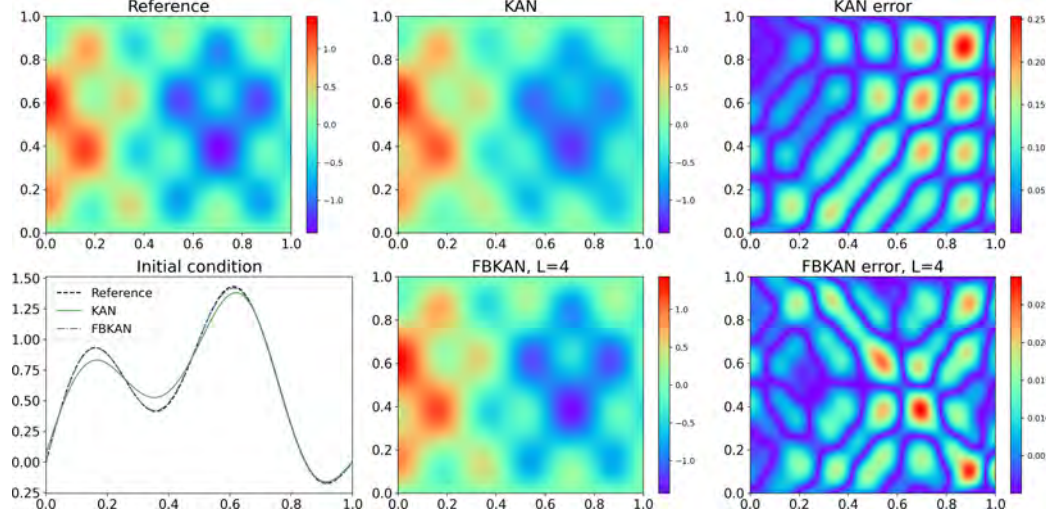


Figure 7.11: Results for physics-informed test 3 with $c = \sqrt{2}$ and $L = 4$. Note that the error plots have different color scales.

decomposition PINN (DADD-PINN), which uses a dual adaptive approach to locate challenging features within subdomains while coordinating features between multiple subdomains. Other works to note are AB-PINNs, which parameterize subdomains as radial basis functions and add subdomains on-the-fly in regions with high residual loss [145] and PINN Balls [146], which use adversarial adaptive sampling [147] to fully learn the domain decomposition.

POUnets [142] are a data-driven approach inspired by partition of unity approach to function approximation, where the solution is represented as a combination of local polynomial basis functions. POUnets simultaneously learn the subdivisions of the subdomain and the coefficients of the basis functions on each subdomain. We use POUnets in this work to learn window functions that define subdomains for FBPINN training. POUnets learn irregular shaped partitions of the problem domain. However, we observe that, in the cases we study, regular shaped partitions result in more accurate FBPINN predictions. And so, we propose separable POUnets, which use a per-axis softmax to generate more regular shaped domains. POUnets have successfully been extended to multilevel domain decompositions to accelerate adaptive partitioning [148], and in [149], a POUnet is used to learn a domain decomposition

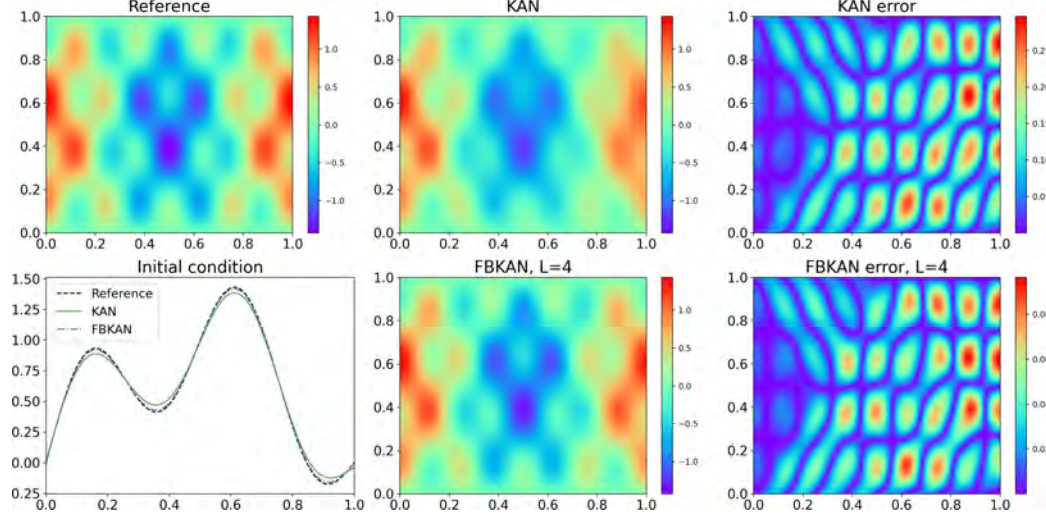


Figure 7.12: Results for physics-informed test 3 with $c = 2$ and $L = 4$. Note that the error plots have different color scales.

for the forcing data that is then used to train a PINN.

7.2.1 Methodology

7.2.1.1 Physics-informed neural networks

PINNs approximate the solution to a PDE by minimizing a loss function that enforces a differential operator given by the governing equations. We define a system over an open, bounded domain $\Omega \in \mathbb{R}^n$ with boundary $\partial\Omega$ as follows:

$$\begin{aligned}
 \mathcal{D}[\mathbf{u}] &= \mathbf{f}, & x \in \Omega, \quad t \in [0, T], \\
 \mathcal{B}_k[\mathbf{u}] &= \mathbf{g}_k, \quad x \in \Gamma_k, \quad k = 1, 2, \dots, K, \quad t \in [0, T], \\
 \mathcal{G}[\mathbf{u}] &= \mathbf{h} & x \in \Omega, \quad t = 0.
 \end{aligned} \tag{7.13}$$

where $\mathbf{x}$ is an n -dimensional input vector in $\Omega \subset \mathbb{R}^d$, $\partial\Omega$ denotes the boundary of the domain, $\mathcal{D}$ represents the differential operator, $\mathcal{B}$ denotes the operator enforcing the boundary conditions and $\mathcal{G}$ denotes the operator that enforces initial conditions. The solution to the PDE is denoted by $\mathbf{u} \in \mathbb{R}^m$, with forcing term $\mathbf{f}$, initial condition $\mathbf{h}$, and boundary conditions $\mathbf{g}_k$. The total number of boundary conditions is K and Γ_k represents the k -th boundary, so that $\cup_{k=1}^K \Gamma_k = \partial\Omega$. The trainable parameters

(weights and biases) for the network are denoted by θ , which are updated to minimize the following loss function:

$$\mathcal{L}(\theta) = \lambda_{ic}\mathcal{L}_{ic}(\theta) + \lambda_{bc}\mathcal{L}_{bc}(\theta) + \lambda_r\mathcal{L}_r(\theta), \quad (7.14)$$

with subscripts ic, bc, r corresponding to loss terms with respect to initial conditions, boundary conditions, and PDE residual. We sample N_{ic} , N_{bc} , and N_r points in the space-time domain for the initial conditions, boundary conditions, and residual collocation points, respectively. The training collocation points are denoted by $\{(\mathbf{x}_{ic}^i, 0)\}_{i=1}^{N_{ic}}$, $\{(x_{k_i}, t_{k_i})\}_{i=1}^{N_{bc}}$, and $\{(\mathbf{x}_r^i, t_r^i)\}_{i=1}^{N_r}$. A common choice for the terms in the loss function is through the residuals according to

$$\mathcal{L}_{ic}(\theta) = \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} [\mathcal{G}[\mathcal{NN}(\mathbf{x}_{ic}^i, 0, \theta)] - \mathbf{h}(\mathbf{x}_i, 0)]^2, \quad (7.15)$$

$$\mathcal{L}_{bc}(\theta) = \sum_{k=1}^K \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} [\mathcal{B}_k[\mathcal{NN}(\mathbf{x}_{k_i}, t_{k_i}, \theta)] - \mathbf{g}_k(\mathbf{x}_{k_i}, t_{k_i})]^2, \quad (7.16)$$

$$\mathcal{L}_r(\theta) = \frac{1}{N_r} \sum_{i=1}^{N_r} [\mathcal{D}[\mathcal{NN}(\mathbf{x}_r^i, t_r^i, \theta)] - \mathbf{f}(\mathbf{x}_i, t_r^i)]^2, \quad (7.17)$$

where $\mathcal{NN}(x, t; \theta)$ denotes the output of the network with parameters θ at (x, t) .

The multi-objective loss function (Equation 7.14) must balance the minimization of the residual, boundary, and initial condition terms, which can contribute to the failure of PINNs [150]. As such, the selection of loss weights λ_{ic} , λ_{bc} , λ_r significantly impacts how well a PINN approximates the solution to the PDE, as the weights help to ensure the loss terms are well balanced. Many methods for adaptively selecting and updating these weights have been proposed and used in practice, including, soft-attention mechanism weights [151] and neural tangent kernel weights [150]. Further, the self-adaptive hybrid weighting and sampling scheme presented in [152] suggests that adaptive re-weighting is most successful when coupled with adaptive resampling. In this work, when a multi-objective loss function is required, we fix the loss weights

(denoted in Table 7.10) to highlight the efficacy of adaptive domain decomposition. However, in the following Section 7.2.1.1, we present an alternative method for hard constraining the boundary and initial conditions, and use this approach when possible.

Hard constrained PINNs

The multi-objective loss function present in PINNs can cause complications during gradient descent. Another approach, presented in [153], is to develop a solution ansatz that strictly enforces the boundary or initial conditions, or both. More precisely, the solution to the PDE is approximated by $[\mathcal{C}u](\mathbf{x}; \theta) \approx u(\mathbf{x})$ where $\mathcal{C}$ is a constraining operator that exactly enforces the boundary or initial condition. This approach allows training to focus on minimizing the interior residual of the governing PDE. We use this approach when such an ansatz can be easily defined.

For instance, in this work, we focus on example problems with Dirichlet boundary conditions, given by

$$\mathbf{u}(\mathbf{x}) = \mathbf{g}_D(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \quad (7.18)$$

To exactly impose these conditions, we can choose a smooth scalar function $\phi_D : \partial\Omega \rightarrow \mathbb{R}$ with $\phi_D > 0$ in Ω and $\phi_D|_{\partial\Omega} = 0$, and set

$$\hat{\mathbf{u}}_D(\mathbf{x}; \theta) = \mathbf{g}_D(x) + \phi_D(\mathbf{x})\mathcal{N}\mathcal{N}(\mathbf{x}; \theta). \quad (7.19)$$

Then $\hat{\mathbf{u}}_D|_{\partial\Omega} = \mathbf{g}_D$ holds by construction, independently of θ . The choice of ϕ is problem dependent, and we document the choice of ϕ for each problem in the results section.

Note that this specific approach is restricted to problems with Dirichlet boundary conditions. However, we also test our method on problems with periodic boundary conditions, where hard constraining boundary terms is more challenging for FBPINNs. For instance, to enforce periodic boundary conditions, one can apply

a Fourier embedding [154, 155] inside each local network $\mathcal{NN}_i$, but the window function applied to each network may break periodicity. So, in this work, we strongly enforce boundary/initial conditions for problems with Dirichlet boundary conditions, and otherwise use the multi-objective loss function with fixed weights defined in Section 7.2.1.1.

7.2.1.2 Partition-of-Unity Networks

Partition-of-Unity Networks (POUnets), presented in [156], are used to approximate functions by learning POU window functions that partition the problem domain with a neural network while polynomial spaces with learnable coefficients make higher-order approximations on each partition. For our approach, we use the POUNets in an iterative manner to learn the window functions that are later used as subdomains for training an FBPINN, as detailed in Section 7.2.1.3. In this section, we present POUNets as they pertain to our method.

Consider a partition of unity $\Phi = \{\omega_i(\mathbf{x})\}_{i=1}^{N_{\text{part}}}$ satisfying $\sum_i \omega_i(\mathbf{x}) = 1$ and $\omega_i(\mathbf{x}) \geq 0$ for all $\mathbf{x}$. POUNets approximate the solution y to some system with the approximant

$$y_{\text{POU}}(\mathbf{x}) = \sum_{\alpha=1}^{N_{\text{part}}} \omega_i(\mathbf{x}) \sum_{j=1}^{\dim(V)} c_{i,j} P_j(\mathbf{x}) \quad (7.20)$$

where $V = \text{span}\{P_j\}$ and $c_{i,j} \in \mathbb{R}$ are coefficients in the polynomial approximations. A common basis choice is $V = \pi_m(\mathbb{R}^d)$, which is the space of polynomials of order m .

One approach is to construct the POU Φ prior to fitting the coefficients $c_{i,j}$ through geometric meshing. Instead, POUNets represent the POU as a neural network, in which the weights and biases ξ are trained to fit the data in the L_2 sense. A variety of architectures can be used to represent these window functions. The original work [142] used a shallow RBF-network and a standard MLP to represent the window functions. We relegate the formulation of the MLP architecture to Appendix ??, but note that the learnable parameters θ_{pou} for the MLP are the weights and biases of

the neural network. The RBF-Net [157, 158] has the form

$$\omega_i(\mathbf{x}; \theta_{\text{pou}}) = \frac{\exp(-|\mathbf{x} - \xi_{1,i}|^2 / \xi_{2,i}^2)}{\sum_j \exp(-|\mathbf{x} - \xi_{1,j}|^2 / \xi_{2,j}^2)}, \quad 1 \leq i \leq N_{\text{part}}. \quad (7.21)$$

This shallow RBF-network uses a single hidden layer without additional nonlinear transformations. Here, the learnable parameters for the POUnet are $\theta_{\text{pou}} = (\xi_1, \xi_2)$, where ξ_1 are the RBF centers and ξ_2 are the RBF shape parameters. The output of the RBF-network is determined by the centers and width parameters of the RBF functions, without additional nonlinear layers.

In practice, the partitions learned by a POUnet with either the MLP or RBF architecture are often irregular in shape. An example of this phenomenon can be seen in Fig. 7.13, where the subdomains group according to the regions of high frequency. Depending on the problem of interest, however, the FBPINN may perform better with a domain decomposition that consists of regular-shaped partitions. And so, we also present Separable POUnets as an alternative POUnet architecture in Section 7.2.1.2 that generates more regular partitions.

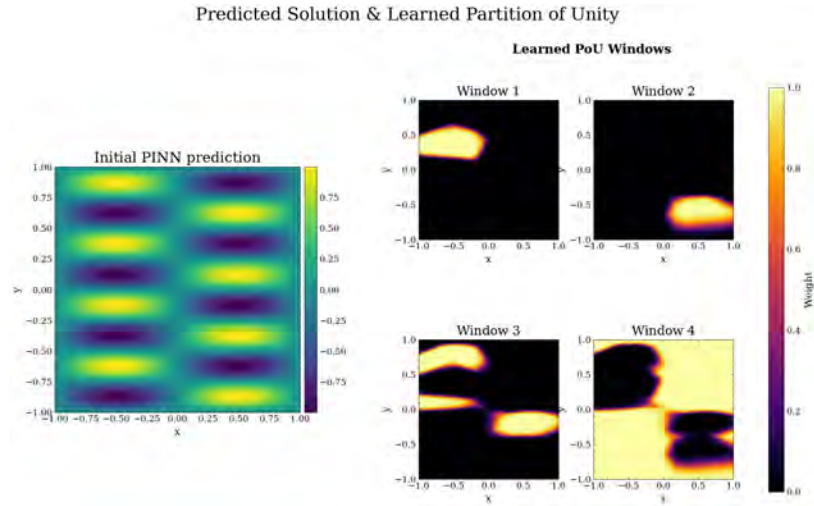


Figure 7.13: Predicted solution by the initial PINN for the Helmholtz equation (Eq. 7.28) with $a_1 = 1, a_2 = 4$ (left) and learned subdomains by the POUnet (right). In this figure, the prediction on the left is used as data for the POUnet, which outputs the subdomains on the right.

The localized basis coefficients and the localization (POU) are fit simultaneously by minimizing a dual objective loss function:

$$\min_{\theta_{pou}, \mathbf{c}} \sum_{i=1}^{N_{data}} \left| \sum_{\alpha=1}^{N_{part}} \phi_{\alpha}(\mathbf{x}_i; \theta_{pou}) \sum_{\beta=1}^{\dim(V)} c_{\alpha, \beta} P_{\beta}(\mathbf{x}_i) - y_i \right|^2. \quad (7.22)$$

We use least-squares gradient descent (LSGD) [159] to alternate between updating the polynomial approximation parameters $\mathbf{c}$ and the network parameters for the network that represents the window function θ . We describe this process in Algorithm 1. In essence, we first fix θ , and update $\mathbf{c}$ which has an exact solution because it is a typical least squares problem. Then, fix $\mathbf{c}^*$, and take one gradient descent step to update θ with the loss function

$$\mathcal{L}_{pou}(\theta_{pou}, \mathbf{c}^*) = \sum_{i=1}^{N_{data}} |y_{pou}(\mathbf{x}_i; \theta_i, \mathbf{c}^*) - y_i|^2. \quad (7.23)$$

In our work, we introduce the addition of an optional penalty term to the loss function of the POUnet to enforce wider overlaps. POUnets are trained to minimize an objective that rewards localized window functions, which in practice results in sharp transitions between subdomains. However, FBPINNs require sufficient overlap width between subdomains to ensure continuity of the learned solution. Our solution to this problem is detailed in Section 7.2.1.2 and is further motivated by our fixed domain decomposition ablation studies in Section 7.2.2.2.

Algorithm 1 Least-squares Gradient Descent (LSGD) [159]

Require: $\theta_{pou, \text{old}}, \mathbf{c}_{\text{old}}, n_{\text{epochs}}, \rho, \eta$

Ensure: $\theta_{pou, \text{new}}, \mathbf{c}_{\text{new}}$

- 1: **function** LSGD($\theta_{pou, \text{old}}, \mathbf{c}_{\text{old}}, n_{\text{epochs}}, \rho, \eta$)
 - 2: $\theta_{pou} \leftarrow \theta_{pou, \text{old}}, \quad \mathbf{c} \leftarrow \mathbf{c}_{\text{old}}$
 - 3: $\mathbf{c} \leftarrow \text{LS}(\theta_{pou})$
 - 4: $\theta_{pou} \leftarrow \text{GD}(\theta_{pou}, \mathbf{c}, \eta)$
 - 5: **end function**
 - 6: $\mathbf{c}_{\text{new}} \leftarrow \text{LS}(\theta_{pou})$
 - 7: $\theta_{\text{new}} \leftarrow \theta_{pou}$
-

Separable POUnets

For the problems we consider, regular or nearly-regular subdomains consistently produce the most accurate FBPINN approximation due to symmetry. As such, we propose separable POUnets, which aim to generate more regular partitions by performing a per-axis softmax. While this approach generalizes to n -dimensions, we present the approach in 2D.

For $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ and total number of partitions $N = n_{x_1} \times n_{x_2}$, where n_{x_1} and n_{x_2} are the number of partitions along the x_1 and x_2 dimensions, respectively, each partition, indexed by (i, j) , has an associated partition function:

$$\{w_{ij}(\mathbf{x})\}_{i=1, j=1}^{n_{x_1}, n_{x_2}}.$$

Two independent one-dimensional functions f_{x_1}, f_{x_2} , in our case MLPs or RBF-nets, produce axis-wise predictions:

$$w_{ij}(\mathbf{x}) = \frac{\exp(f_{x_1}(x_1; \theta_{x_1})_i + f_{x_2}(x_2; \theta_{x_2})_j)}{\sum_{p=1}^{n_{x_1}} \sum_{q=1}^{n_{x_2}} \exp(f_{x_1}(x_1; \theta_{x_1})_p + f_{x_2}(x_2; \theta_{x_2})_q)}, \quad i = 1, \dots, n_{x_1}, j = 1, \dots, n_{x_2}.$$

The denominator factorizes,

$$\sum_{p=1}^{n_{x_1}} \sum_{q=1}^{n_{x_2}} \exp(f_{x_1}(x_1; \theta_{x_1})_p + f_{x_2}(x_2; \theta_{x_2})_q) = \left(\sum_{p=1}^{n_{x_1}} \exp(f_{x_1}(x_1; \theta_{x_1})_p) \right) \left(\sum_{q=1}^{n_{x_2}} \exp(f_{x_2}(x_2; \theta_{x_2})_q) \right),$$

and we can rewrite w_{ij} in the fully factorized form:

$$w_{ij}(\mathbf{x}) = \frac{\exp(f_{x_1}(x_1; \theta_{x_1})_i)}{\sum_{p=1}^{n_{x_1}} \exp(f_{x_1}(x_1; \theta_{x_1})_p)} \frac{\exp(f_{x_2}(x_2; \theta_{x_2})_j)}{\sum_{q=1}^{n_{x_2}} \exp(f_{x_2}(x_2; \theta_{x_2})_q)}. \quad (7.24)$$

These partitions satisfy the POU condition, ensuring that their sum is always one at any point in the domain: $\sum_{i=1}^{n_{x_1}} \sum_{j=1}^{n_{x_2}} w_{ij}(\mathbf{x}) = 1 \quad \forall \mathbf{x} \in \Omega$.

And so, the single softmax is algebraically identical to first normalizing each axis ($\sum_i w_i = 1, \sum_j w_j = 1$) and then taking their outer product. This compact formulation preserves the POU property. We test both Separable-MLP-POUnets and Separable-RBF-POUnets alongside the standard MLP and RBF-net formulations in this work, but our default setting for each problem is the Separable-RBF-POUnet because the subdomains derived from the Separable-RBF-POUnets often results in better final error approximations after FBPINN training.

Overlap regularization term

We also introduce an overlap regularization term to the loss function of the POUnet to ensure that the learned domain decomposition includes wide enough overlapping regions between subdomains. The necessity of this is motivated in our ablation studies in Section 7.2.2.2, in which we show that simply adding collocation points to the overlapping regions cannot reliably improve accuracy without increasing overlap width.

We introduce a penalty term to Equation 7.22 that penalizes the Laplacian of the window functions. The Laplacian of a window function w_{ij} controls the curvature of the subdomain, so this penalty term penalizes rapid blending between subdomains. The penalty is given by

$$\mathcal{L}_{lap}(\theta_{pou}) = \frac{1}{N} \sum_{n=1}^N \sum_{i,j}^{N_{part}} \|\Delta w_{ij}(x_n, \theta_{pou})\|_2^2 \quad (7.25)$$

With the Laplacian penalty term, our composite loss function for a POUnet becomes

$$\mathcal{L}(\theta_{pou}, \mathbf{c}^*) = \mathcal{L}_{pou}(\theta_{pou}, \mathbf{c}^*) + \lambda_{lap} \mathcal{L}_{lap}(\theta_{pou}) \quad (7.26)$$

To demonstrate the efficacy of this added penalty term, we use a POUnet to learn a subdivision of the problem domain for the 2D Helmholtz problem presented in

Equation 7.28. To highlight the impact of the overlap term, we plot the difference between the window functions learned from a POUnet with and without the overlap term in Fig. 7.14. In practice, no overlap penalization results in very sharp partition boundaries, which represents successful POUnet training as developed in [142], but is ill-suited for downstream FBPINN training.

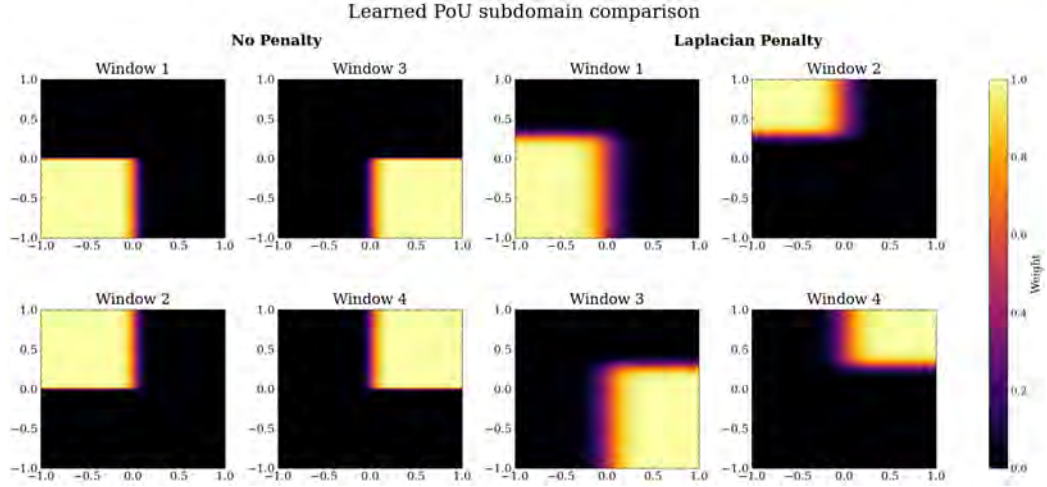


Figure 7.14: Comparison of the subdomains learned by a POUnet for the Helmholtz equation with no added penalty terms to enforce wider overlaps (left), and the subdomains learned by a POUnet with Laplacian penalty term (right). The Laplacian penalty term has a weight of $\lambda_{\text{Lap}} = 0.0001$. The subdomains with Laplacian penalty have a wider transition between subdomains, which aids in downstream FBPINN training.

7.2.1.3 ADD-FBPINNs

We propose adaptive domain decomposition via a hierarchical approach that relies on a POUnet to learn optimal subdomains on which an FBPINN is trained. The integration of POUnets and FBPINNs is motivated by the need for flexible, physics-informed domain decomposition for solving PDEs. A key benefit of our method is that no data, either of the forcing term or the solution, is needed. Since POUnets are a data-driven approach, we use a naive approximation given by a standard PINN to facilitate the POUnet training. POUnets allow the neural networks to focus on partitioning the problem domain, and we utilize the learned partitions from a POUnet

to define subdomains on which to train a FBPINN.

We present our approach in Algorithm 2 and provide an illustration in Fig. 7.15. This adaptive algorithm has the following stages:

1. **Initialization:** A baseline PINN is trained on the entire problem domain Ω to generate a low-fidelity approximation $\hat{\mathbf{u}}^{(1)}$. This solution serves as the first target, $\mathbf{y}_{\text{target}}$, for the POUnet. A schedule of increasing number of partitions $\{N_1, N_2, \dots, N_S\}$ is defined.
2. **Iterative partitioning and refinement:** The algorithm iterates through the schedule of partition numbers in the schedule. For each number of partitions $N = N_1, N_2, \dots, N_S$:
 - (a) **POUnet training:** A POUnet is trained to learn window functions for a set of N partitions, $\Phi^{(N)}$, using the most recent prediction $\mathbf{y}_{\text{target}}$.
 - (b) **Collapse detection:** The algorithm checks if any partition has “collapsed,” i.e., at least one window function has maximum value drop below the threshold τ .
 - **If a collapse occurs:** The current partition number is marked as N_{fail} . The algorithm reverts back to the last successful partition configuration (Φ^* at N_{prev}) and trains an FBPINN using the window functions Φ^* as subdomains to refine the approximate solution to the PDE. The new solution becomes the updated $\mathbf{y}_{\text{target}}$, and the POUnet is re-tried on N_{fail} . If the collapse occurs again, training terminates and returns the last successful result.
 - **If no collapse occurs:** The learned partitions $\Phi^{(N)}$ are accepted as the new best configuration Φ^* , and the algorithm proceeds to the next partition number in the schedule.

3. **Final FBPINN training:** Once the loop completes, either by exhausting the schedule or terminating early due to a repeated collapse, a final FBPINN is trained on the last successful number of partitions Φ^* . This yields the final solution.

Algorithm 2 ADD-FBPINN

```

1: Train PINN and obtain  $\hat{u}^{(1)}$ ;  $\mathbf{y}_{\text{target}} \leftarrow \hat{u}^{(1)}$ 
2: Schedule partition numbers  $\{N_1, N_2, \dots, N_S\}$ ; choose collapse threshold  $\tau$ 
3:  $\Phi^* \leftarrow \Phi^{(1)}$ ,  $N_{\text{prev}} \leftarrow N_1$ 
4: for  $N \in \{N_2, \dots, N_S\}$  do
5:   Train POUnet at  $N$  using  $\mathbf{y}_{\text{target}}$ ; obtain partitions  $\Phi^{(N)} = \{\phi_i^{(N)}\}_{i=1}^N$ 
6:   if  $\exists \alpha$  s.t.  $\max_{\mathbf{x}} \phi_i^{(N)}(\mathbf{x}) < \tau$  then ▷ Collapse detected at  $N$ 
7:     Train FBPINN at  $N_{\text{prev}}$  with  $\Phi^*$  to get  $\hat{u}^{(N_{\text{prev}})}$ 
8:      $\mathbf{y}_{\text{target}} \leftarrow \hat{u}^{(N_{\text{prev}})}$  ▷ Update target with a better solution
9:     Train POUnet at  $N$  using new  $\mathbf{y}_{\text{target}}$  and obtain new  $\Phi^{(N)}$  ▷ Retry
10:    POUnet training for  $N$ 
11:    if  $\exists \alpha$  s.t.  $\max_{\mathbf{x}} \phi_i^{(N)}(\mathbf{x}) < \tau$  then ▷ Collapse again at  $N$ 
12:      return last FBPINN (at  $N_{\text{prev}}$ ) and  $\Phi^*$  ▷ Stop
13:    end if
14:     $\Phi^* \leftarrow \Phi^{(N)}$ ,  $N_{\text{prev}} \leftarrow N$  ▷ Update partitions and proceed to next  $N$ 
15:  end if
16: end for
17: Train FBPINN at  $N_{\text{prev}}$  with  $\Phi^*$ 
18: return last FBPINN and  $\Phi^*$ 

```

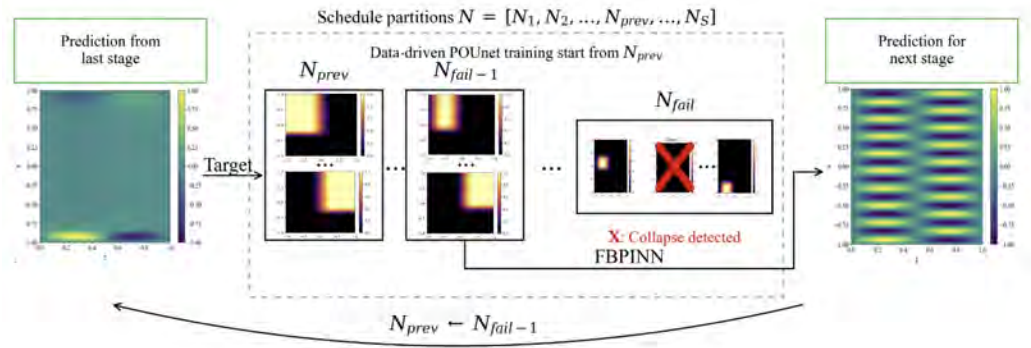


Figure 7.15: Graphical abstract of the ADD-FBPINN

A partition is fully effective when its maximum value is 1, and its influence diminishes as its maximum value approaches zero, because a value near zero means

the partition never achieves exclusive ownership of any point in the domain. And so, a collapse is defined as the maximum value of any partition dropping below a threshold τ . Setting $\tau = 1.0$ can result in false detection of a collapse so we take $\tau = 0.8$ to appropriately balance partition effectiveness and collapse detection. We also note that, to replicate the normalization of inputs to subdomains in FBPINN training (Equation 7.2), we treat learned subdomains as a rectangle that covers the extent of the subdomain.

In the following section, we test this approach on the 2D Helmholtz equation, 2D Poisson equation with multiscale features, and viscous Burgers equation. We tested four architectures of POUnets in our experiments and found that the separable RBF-POUnet most often results in the lowest final relative L_2 error, and so we present only those results and discuss the impacts of the standard RBF, MLP and separable MLP POUnet architectures in Appendix ??.

7.2.1.4 Sampling Methods

The placement and spatial coverage of the collocation points in the domain Ω are critical to the training of PINNs, as they strongly affect the convergence and accuracy [160]. For our 1D examples, we default to sampling these points from a uniform distribution over Ω . For the 2D examples, we default to the Hammersley [161] sampling approach as a low-discrepancy quasi-random sequence that provides more uniform coverage of the domain than standard random sampling.

During training, residual-based adaptive distribution (RAD) resampling [160] can be applied, which periodically resamples collocation points with probability proportional to the local PDE residual magnitude, concentrating points in regions of high error. Our experiments suggest that resampling points according to residual error results in improved FBPINN predictions. For brevity, a presentation of this approach is included in Appendix ??.

7.2.2 Results

7.2.2.1 Problems studied

1D Sine ODE

As an ablation test (Section 7.2.2.2) on the importance of subdomain overlap width and density of collocation points in the overlapping regions, we consider the equation

$$\frac{du}{dx} = \sin(\omega_1 x) + \sin(\omega_2 x), \quad u(0) = 0 \quad (7.27)$$

which has the exact solution

$$u(x) = \frac{1 - \cos(\omega_1 x)}{\omega_1} + \frac{1 - \cos(\omega_2 x)}{\omega_2}.$$

We consider the case where $\omega_1 = 1, \omega_2 = 45$.

Helmholtz equation

We consider the following two-dimensional Helmholtz problem. The equation in the spatial domain (x, y) is given by

$$\begin{aligned} u_{xx} + u_{yy} + k^2 u - q(x, y) &= 0 \\ u(x, y) &= 0, \quad x, y \in \partial\Omega, \quad \Omega = [-1, 1] \times [-1, 1] \end{aligned} \quad (7.28)$$

where $q(x, y)$ is the following forcing term with k set to 1 :

$$\begin{aligned} q(x, y) &= - (a_1 \pi)^2 \sin(a_1 \pi x) \sin(a_2 \pi y) \\ &\quad - (a_2 \pi)^2 \sin(a_1 \pi x) \sin(a_2 \pi y) \\ &\quad + k^2 \sin(a_1 \pi x) \sin(a_2 \pi y). \end{aligned}$$

With this, we have the closed-form, analytical solution

$$u(x, y) = \sin(a_1 \pi x) \sin(a_2 \pi y).$$

This problem is used as a benchmark test for the performance of our ADD-FBPINN method. For this purpose, we vary the values of a_1, a_2 to increase the difficulty of the problem. We also use this problem with $a_1 = 1, a_2 = 4$ in an ablation study with standard FBPINNs to gauge the impact of increasing overlap width between subdomains, detailed in Section 7.2.2.2.

Poisson equation with multiscale features

Next, we study a two-dimensional Poisson equation that is used in [141] given by

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega = [0, 1]^2, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{7.29}$$

where the source term $f(x_1, x_2)$ is

$$f(x_1, x_2) = \frac{2}{n} \sum_{i=1}^n (w_i \pi)^2 \sin(w_i \pi x_1) \sin(w_i \pi x_2), \tag{7.30}$$

which corresponds to the exact solution

$$u(x_1, x_2) = \frac{1}{n} \sum_{i=1}^n \sin(w_i \pi x_1) \sin(w_i \pi x_2). \tag{7.31}$$

In this case, multiscale features are present in the solution, and we can tune n and w_i to adjust the number of components and the frequency of each component.

Burgers equation

For the final benchmark, we employ two cases of Burgers equation. Case 1 is given by

$$\begin{aligned}
u_t + uu_x - (0.01/\pi)u_{xx} &= 0, \quad x \in [-1, 1], \quad t \in [0, 1] \\
u(0, x) &= -\sin(\pi x) \\
u(t, -1) &= u(t, 1) = 0
\end{aligned} \tag{7.32}$$

where ν is the viscosity.

We can strongly enforce boundary and initial conditions using the following ansatz:

$$u(t, x) = -\sin(\pi x)(1 - t) + (1 - x^2)t(NN_\theta(t, x))$$

We also consider the same PDE with a shifted initial condition, denoted Case 2,

$$\begin{aligned}
u(0, x) &= -\sin(\pi x + \frac{\pi}{4}) \\
u(t, -1) &= u(t, 1) \\
u_x(t, -1) &= u_x(t, 1).
\end{aligned} \tag{7.33}$$

For this example, we choose to enforce boundary conditions via a term in the loss function,

$$\mathcal{L}_{bc}(\theta) = \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} [\mathcal{NN}(t_i, -1, \theta) - \mathcal{NN}(t_i, 1, \theta)]^2 + \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} [\mathcal{NN}_x(t_i, -1, \theta) - \mathcal{NN}_x(t_i, 1, \theta)]^2$$

but keep strongly enforced initial conditions with the ansatz:

$$u(t, x) = -\sin(\pi x + \frac{\pi}{4})(1 - t) + \frac{\sqrt{2}}{2}(1 - x^2)t(NN_\theta(t, x))$$

7.2.2.2 Impact of subdomain overlap width

In this section, we perform an ablation study using standard FBPINNs with varying overlap widths. This ablation study aims first to show that increasing the overlap width can greatly improve the accuracy of FBPINNs. We demonstrate that there

exist cases where an FBPINN can fail to accurately capture the dynamics of the solution, yet when the same network is trained with a larger overlap region, the prediction from the FBPINN improves significantly. To do this, we fix the number of subdomains in a non-adaptive FBPINN via Equation 7.4 and vary only the overlap width. We select examples exhibiting multiscale or high-frequency behavior, which are known to improve the performance of a standard PINN [123], to isolate the effects of varying overlap widths on FBPINN accuracy. These are the 1D Sine equation 7.27, 2D Helmholtz 7.28, and 2D Poisson 7.29. The 2D Helmholtz equation and 2D Poisson equation are selected as benchmarks for our ADD-FBPINN, and each equation has multiple cases tested at varying levels of frequency/multiscale features. And so, for the ablation study, we selected the lowest frequency cases. Specific parameter choices are described below.

For each test case, we over-saturate the collocation points in the overlapping regions with the smallest width. The extra collocation points are sampled according to the following scheme. First, we assign each overlap a density, given by

$$\rho_{\text{base}} = \frac{N^d}{|\Omega|}, \quad N_{\text{extra}} = \rho_{\text{base}} \cdot \alpha \cdot |\Omega_{\text{overlap}}|. \quad (7.34)$$

where N denotes the number of collocation points in each dimension before oversampling, $|\Omega|$ denotes the size of the domain, $|\Omega_{\text{overlap}}|$ denotes the size of the overlap, and $\alpha \geq 1$ is an oversampling factor that determines how many extra collocation points will be placed in the overlapping region. For this study, we set $\alpha = 4.0$ to ensure that the overlapping regions are indeed oversaturated with collocation points.

We first test this on the 1D Sine ODE in Equation 7.27. The total width of the overlap between two adjacent subdomains is set via a parameter δ , as described in Equation 7.5. As the overlap width increases, the relative L2 error decreases, as seen in Table 7.3. On the other hand, simply adding collocation points to the overlapping regions does not significantly improve accuracy. In fact, we see that when the overlap

is wider, $\delta = 0.3$, we still see a reduction in error from the thinner overlap $\delta = 0.1$ even when the number of points in the overlapping regions is similar. For our method, this means the adaptive domain decomposition must capture a wide enough overlap to ensure high accuracy with the FBPINN. In Fig. 7.16, we plot the predicted and exact solutions for the cases with and without oversampling in the overlap. As the errors suggest, the mismatch in dynamics is not recovered from the case without oversampling. We also visualize the collocation points in the oversampled case in Fig. 7.17. In contrast, Fig. 7.18 depicts the predicted solutions with narrow overlap $\delta = 0.1$ and wide overlap $\delta = 0.3$, and the improvement in the prediction from an FBPINN with wider overlapping subdomains is clear.

This pattern continues for the 2D examples as well, with errors for the Helmholtz equation depicted in Table 7.4 and the 2D Poisson equation with multiscale features in Table 7.5. For Helmholtz, when the points are oversampled in a narrow overlapping region no meaningful improvement in accuracy is seen. For this case, we also plot the distribution of collocation points for the cases with narrow overlap with and without oversampling in the overlap region in Fig. 7.19. In all cases, we see that the reduction in error is linked specifically to wider overlapping regions, not simply the number of collocation points in each overlap.

With these results, we conclude that overlap width rather than the density of collocation points in overlapping regions is a dominant factor controlling the accuracy with FBPINNs. This is in alignment with the findings in [123] and has a direct implication for our adaptive domain decomposition framework: it is insufficient for the POUnet to simply learn the optimal number and placement of subdomains. The learned partitions must also exhibit wide enough overlapping regions to ensure smooth blending between subdomains, which results in higher accuracy for the downstream FBPINN training. However, the POUnet architectures naturally reward sharp partitions, since sharp boundaries reduce the approximation error in the POUnet’s own training objec-

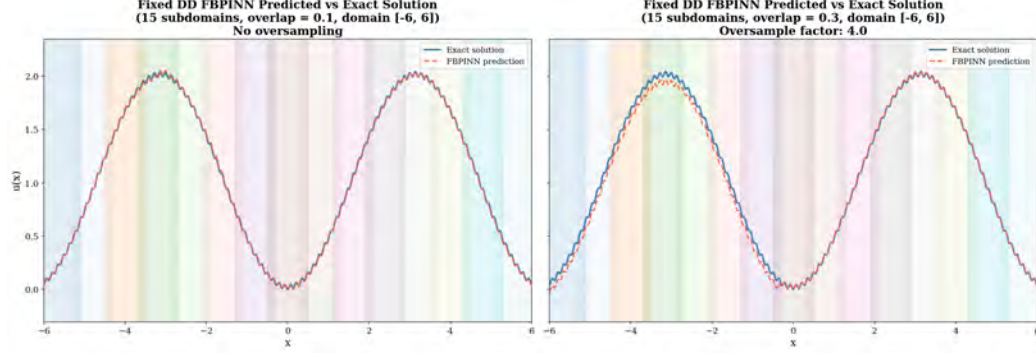


Figure 7.16: Predicted and exact results for the 1D Sine ODE with 2000 base points (left) and with 2000 base points + oversampling in the overlap (right).

tive (Equation 7.22), because points active in multiple subdomains are represented by a blend of local polynomial models from the two subdomains which may disagree and increase error. Left unconstrained, the POU-net learns sharp partitions with insufficient overlap widths that are well suited for function approximation but inadequate for FBPINN training. The adaptive domain decomposition results in the following section demonstrate that the Laplacian penalty regularization term increases the overlap regions between subdomains to allow for effective FBPINN training.

Table 7.3: Relative L2 errors and number of collocation points active in exactly 2 subdomains for the 1D Sine equation with $\omega_1 = 1$ and $\omega_2 = 45$ across a range of overlap widths $\delta = 0.1, 0.2, 0.3$ and 30 uniform subdomains. The second row denotes a case where more collocation points were placed in the overlapping regions with an enrichment factor of 4.0, showing that more collocation points does not substantially boost accuracy.

Overlap δ	Relative L2 error	Num points active in 2 subdomains
$\delta = 0.1$	7.38×10^{-3}	533
$\delta = 0.1$	3.89×10^{-2}	2206
$\delta = 0.2$	4.67×10^{-3}	1005
$\delta = 0.3$	3.59×10^{-3}	1509

7.2.2.3 ADD-FBPINN results

In this section, we present the results for our ADD-FBPINN approach. All hyperparameters for each model can be found in Appendix 7.3.

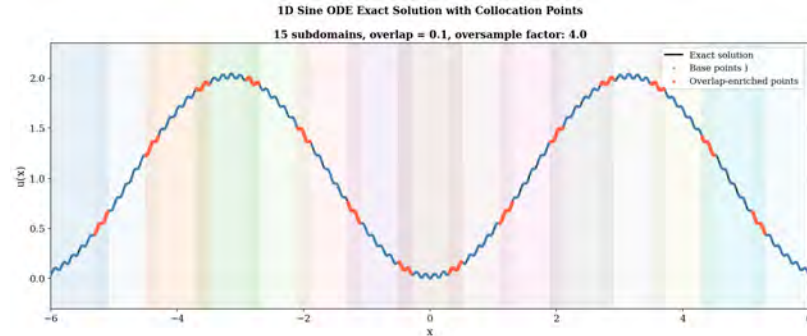


Figure 7.17: Collocation points used to solve the 1D Sine ODE with oversample factor of $\alpha = 4.0$. Collocation points are plotted along the exact solution. Blue points denote points in only one subdomain, and red points denote collocation points active in two subdomains.

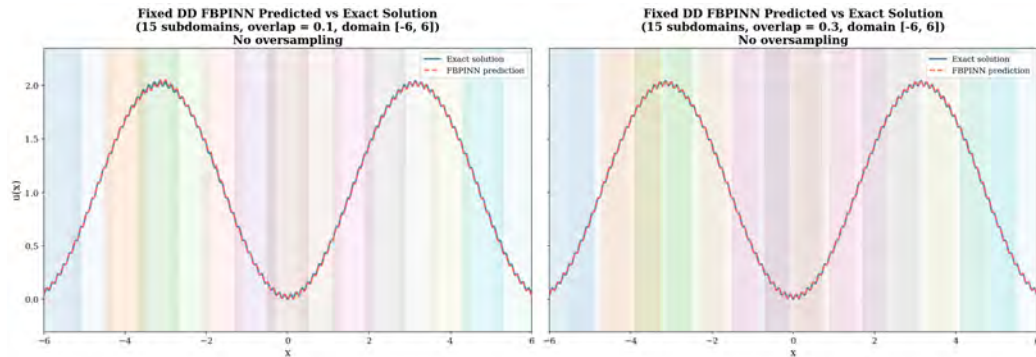


Figure 7.18: Predicted and exact results for the 1D Sine ODE with 2000 base points and an overlap width of $\delta = 0.1$ (left) and $\delta = 0.3$ (right). The wider overlap results in a more accurate solution.

Table 7.4: Relative L2 errors and number of collocation points active in exactly 2 subdomains for the 2D Helmholtz equation with $a_1 = 1$ and $a_2 = 4$ across a range of overlap widths $\delta = 0.09, 0.14, 0.19, 0.24$ and 25 uniform subdomains. The second row denotes a case where more collocation points were placed in the overlapping regions with an enrichment factor of 3.0, showing that more collocation points does not substantially boost accuracy.

Overlap δ	Relative L2 error	Num points active in 2 subdomains
$\delta = 0.09$	1.46×10^{-2}	3353
$\delta = 0.09$	1.96×10^{-2}	42,346
$\delta = 0.14$	1.29×10^{-2}	4887
$\delta = 0.19$	5.52×10^{-3}	6225
$\delta = 0.24$	1.50×10^{-3}	7370

Table 7.5: Relative L2 errors and number of collocation points active in exactly 2 subdomains for the 2D Poisson equation with multiscale features with $\omega_i = [1, 2, 4]$ across a range of overlap widths $\delta = 0.09, 0.14, 0.19, 0.24$ and 25 uniform subdomains. An oversampling factor of $\alpha = 4.0$ is used.

Overlap δ	Relative L2 error	Num points active in 2 subdomains
$\delta = 0.09$	1.53×10^{-1}	7440
$\delta = 0.09$	3.10×10^{-1}	44,266
$\delta = 0.14$	8.33×10^{-2}	8100
$\delta = 0.19$	9.07×10^{-2}	11,245
$\delta = 0.24$	2.84×10^{-1}	14,876

Helmholtz equation

We first evaluate our ADD-FBPINN on the 2D Helmholtz equation presented in Equation 7.28. The Helmholtz equation is a classic benchmark for domain decomposition methods [141, 162] because its solution complexity is closely linked to the wave numbers a_1 and a_2 . Importantly, anisotropic cases where $a_1 \neq a_2$ produce solution with very different length scales along each axis, which uniform domain decomposition cannot efficiently capture with a small number of subdomains. And so, we use this example to show that our adaptive method can learn decompositions that are aligned with challenging regions of the solution.

We consider multiple cases by varying the mode number a_1, a_2 with examples rang-

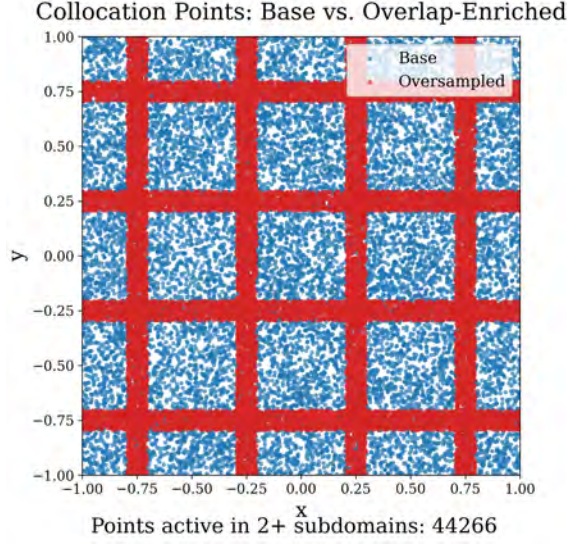


Figure 7.19: Distribution of collocation points for the 2D Helmholtz equation ablation study with an overlap width of $\delta = 0.09$. 10000 base collocation points are sampled over the entire domain using Hammersley sampling. Then, extra points are added to regions where 2 or more subdomains are active with an oversampling factor of $\alpha = 4.0$.

ing from low-to-high frequency, as well as isotropic and anisotropic cases. The cases are listed by mode numbers in Table 7.6 with relative L2 errors and number of subdomains learned for each case. For each case, we compare the relative L2 errors given by a baseline PINN, a FBPINN with fixed uniform domain decomposition, and the ADD-FBPINN. We select a range of overlap widths centered around the baseline overlap used in our main experiments. The fixed domain decomposition FBPINN always uses the same number of subdomains learned by the adaptive method to provide a fair comparison. For all cases, the separable RBF architecture is used for the POUnet, and the Laplacian penalty term with a weight of $\lambda_{\text{lap}} = 0.0001$ is used, selected based on results from a preliminary sweep over 0.01, 0.001, 0.0001 based on training stability and resulting overlap widths.

The baseline PINN’s performance drops in accuracy as the mode numbers increase, which is consistent with the well-documented failure mode of standard PINNs to capture the dynamics of high wave-number Helmholtz problems. Our ADD-FBPINN

improves the error of the fixed DD by one or two magnitudes of error in all cases, and improves error over standard PINNs by two or three magnitudes of error. In some cases, the optimal domain decomposition is still mostly uniform, but shifted slightly to align with regions of high error. For instance, see the subdomains learned from Case 3 in Fig. 7.20. The partition locations correspond to regions of higher error given by the initial PINN prediction at Stage 0, by nature of the data-driven POUnet. The initial PINN has a lower error in the bottom right corner, so the POUnet places one large subdomain in that region, while the work in the higher-error regions is split by the remaining subdomains. This demonstrates that the adaptive framework does not simply impose a uniform domain decomposition strategy, but actually allocates subdomains in direct response to where the baseline solution is least accurate.

The largest relative gains are found in the anisotropic cases, where $a_1 = 3, a_2 = 8$ and $a_1 = 1, a_2 = 9$. In particular, the baseline PINN completely fails to capture any meaningful dynamics in the highly anisotropic Case 4, but the adaptive framework is still able to recover and produce an accurate solution. The evolution of predictions from Stage 0, the initial PINN, to Stage 1, the FBPINN with learned subdomains, for Case 4 are depicted in Fig. 7.21. Rather than uniform subdomains, the POUnet adapts the size and placement of each subdomain to the local complexity of the solution, as seen in Fig. 7.21.

It is also worth noting that the fixed domain decomposition FBPINN, while consistently outperforming the baseline PINN, does not achieve competitive accuracy in most cases. This is partly due to the fact that we constrain the fixed domain decomposition FBPINN to the same number of subdivisions learned by the adaptive method, which is often a small number. Beyond subdomain count, the fixed domain decomposition is not guaranteed to align with the length scales of the solution. If more subdomains are used, the fixed decomposition errors improve at the cost of significant increase in computational cost. This highlights a key advantage of our ap-

proach: by learning subdomain placements aligned with features of the solution, the adaptive method achieves high accuracy with fewer subdomains. The POU-net only requires 2000 iterations to train, which is negligible compared to the cost of training an FBPINN on a larger number of domains.

Table 7.6: Relative L2 errors for a standard PINN, FBPINN, and ADD-FBPINN on the 2D Helmholtz equation. Results are shown for a range of wave numbers to highlight the ability of this method to capture both high-frequency isotropic cases and high-frequency anisotropic cases.

Case	Mode numbers	PINN	FBPINN	ADD-FBPINN	Number of subdomains
Case 1	$a_1 = 1, a_2 = 4$	2.35×10^{-2}	1.50×10^{-3}	5.88×10^{-4}	4
Case 2	$a_1 = 6, a_2 = 6$	7.89×10^{-2}	2.54×10^{-1}	6.41×10^{-3}	4
Case 3	$a_1 = 3, a_2 = 8$	9.89×10^{-1}	4.55×10^{-1}	3.89×10^{-3}	4
Case 4	$a_1 = 1, a_2 = 9$	9.90×10^{-1}	6.02×10^{-2}	4.73×10^{-3}	9

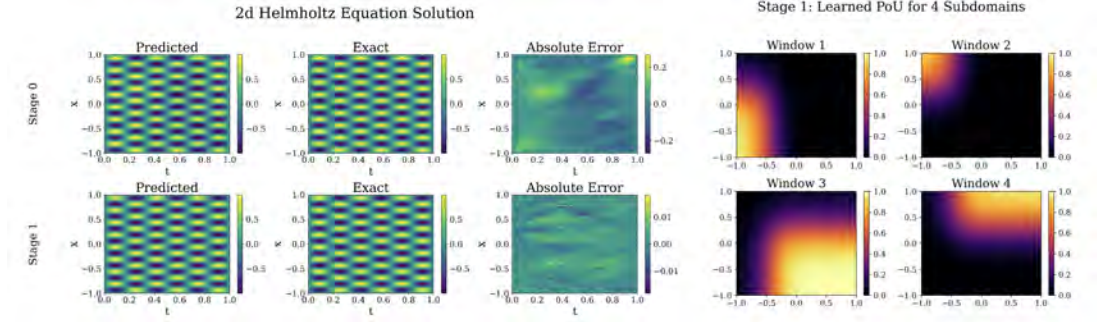


Figure 7.20: Results for the 2D Helmholtz equation with $a_1 = 3$ and $a_2 = 8$. On the left, we see the predicted, exact, and absolute error for both Stage 0 (initial PINN), and Stage 1 (ADD-FBPINN). On the right, we see the learned subdomains used as the domain decomposition in Stage 1.

Poisson equation in two dimensions with multiscale features

We use the 2D Poisson equation from Equation 7.29 to demonstrate the ability of adaptive domain decomposition to capture multiscale features. Unlike the Helmholtz equation, where frequency is determined by a pair of wave numbers, the Poisson equation is constructed as a superposition of n modes of frequencies $\{\omega_i\}_{i=1}^n$, like in [141]. This allows us to directly control the spread and number of scales present in

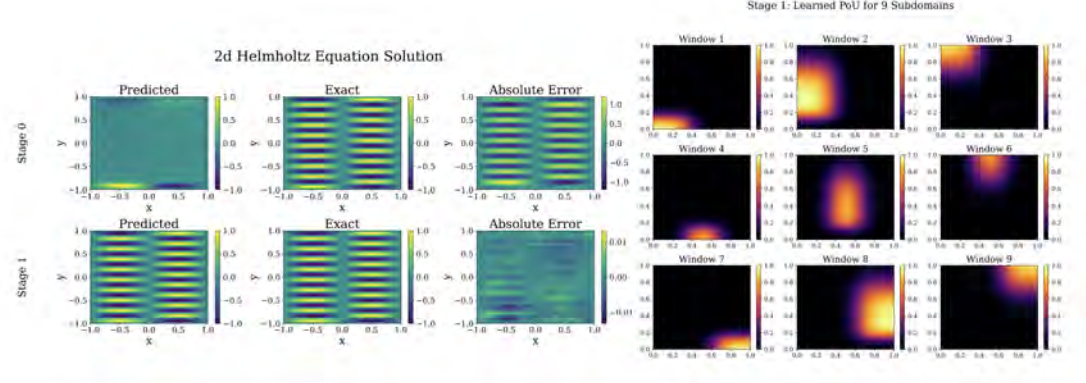


Figure 7.21: Results for the 2D Helmholtz equation with $a_1 = 1$ and $a_2 = 9$. On the left, we see the predicted, exact, and absolute error for both Stage 0 (initial PINN), and Stage 1 (ADD-FBPINN). On the right, we see the learned subdomains used as the domain decomposition in Stage 1.

the solution. We consider 4 cases of increasing frequency, and test our ADD-FBPINN against a baseline PINN and fixed DD FBPINN.

Final relative L2 errors are presented in Table 7.8. In Cases 1 and 2, both the fixed and ADD-FBPINNs offer modest improvements over the baseline PINN, which is consistent with the expectation that domain decomposition provides the most benefit as solution complexity increases. In Case 3, the fixed DD FBPINN only slightly outperforms than the baseline PINN despite using the same number of subdomains as the adaptive method. For Cases 1-3, the adaptive approach learns 4 subdomains that are more or less uniformly placed in the problem domain, similar to the partitions of the fixed DD FBPINN. So for these cases, placement is not the true gain of the adaptive approach, but optimal overlap width. While overlap width can be selected according to the problem in a standard FBPINN, the adaptive approach immediately finds the correct overlap without the need for hyperparameter tuning, as seen in Fig. 7.22.

A similar effect is present in Case 4 (Fig. 7.23), where the frequency is highest with $\omega_i = [1, 2, 4, 8, 16]$, the fixed FBPINN only slightly outperforms the baseline PINN, although both cases have higher errors than the ADD-FBPINNs. In this case,

however, the ADD-FBPINNs learns 16 subdivisions and favors a higher number of sharp partitions. Because the FBPINN does not reach the same accuracy as the ADD-FBPINN, this suggests that 16 uniformly-placed subdomains are not enough to resolve the full scale of the solution, whereas 16 learned subdomains can. By contrast, the adaptive approach reduces the error by roughly an order of magnitude using the same subdomain count, by concentrating subdomains where the multiscale structure of the solution demands it.

Table 7.7: Relative L2 errors for a standard PINN, FBPINN, and ADD-FBPINN on the 2D Poisson equation with multiscale features. Results are shown for a range of frequencies.

Case	Frequencies $\{\omega_i\}$
Case 1	$\omega = [1, 2]$
Case 2	$\omega = [1, 2, 4]$
Case 3	$\omega = [1, 2, 4, 8]$
Case 4	$\omega = [1, 2, 4, 8, 16]$

Burgers equation

We use viscous Burgers equation to highlight the ability of the ADD-FBPINN to accurately subdivide a problem domain not only in space, but also in time. This example also certifies that the POU net accurately captures challenging regions of the solution, specifically, the sharp gradient region that forms around $t \approx 0.3$. We utilize two test cases here: first, the standard benchmark version depicted in Equation 7.32 with the sharp gradient located at $x = 0.0$. Additionally, we consider Equation 7.33,

Table 7.8: Relative L2 errors for a standard PINN, FBPINN, and ADD-FBPINN on the 2D Poisson equation with multiscale features. Results are shown for a range of frequencies with training difficulty increasing for each case.

Case	PINN	FBPINN	ADD-FBPINN	Number of subdomains
Case 1	6.09×10^{-4}	2.89×10^{-4}	9.77×10^{-5}	4
Case 2	1.76×10^{-3}	1.12×10^{-3}	7.82×10^{-4}	4
Case 3	1.45×10^{-2}	1.33×10^{-2}	4.33×10^{-3}	4
Case 4	8.42×10^{-1}	5.84×10^{-1}	4.51×10^{-2}	16

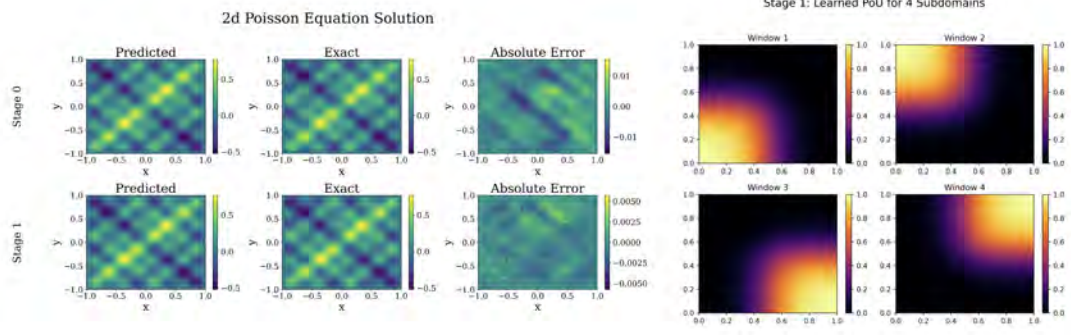


Figure 7.22: Results for the 2D Poisson equation with multiscale features for Case 3 with $\omega = [1, 2, 4, 8]$. On the left, we have predicted, exact, and absolute error for both Stage 0 (baseline PINN) and Stage 1 (ADD-FBPINN). On the right, we see the learned domain decomposition used in Stage 1. In this case, the subdivision is still uniform, but the adaptive method is able to learn the ideal overlap width.

a non-symmetric variant where the shock is located at $x = -0.25$.

We test these two cases to confirm that the POUnet does in fact learn subdomains that are aligned with challenging regions of the problem. In this case, the subdomains can locate the sharp gradient. The subdomains learned for each case are plotted in Fig. 7.26. In fact, the POUnet for the periodic equation locate the position of the sharp gradient at $x = -0.25$ but also adds a wider overlap at the time where the shock forms $t \approx 0.3$.

Relative L2 errors and number of subdomains learned are shown in Table 7.9. Note that, again, in the case where the shock is located at $x = -0.25$, the baseline PINN even outperforms the FBPINN. We compare the predicted solutions and errors for this case in Fig. 7.25. In Fig. 7.26(b), we see that the subdomains the adaptive FBPINN learns are aligned with the challenging region of the problem, whereas the fixed DD subdivisions are uniform. The ADD-FBPINN can even find correct placement for subdomains even for problems where the difficulty arises in space-time. Our approach offers feature-aligned domain decomposition without the need to define such subdomains by hand for each problem of interest.

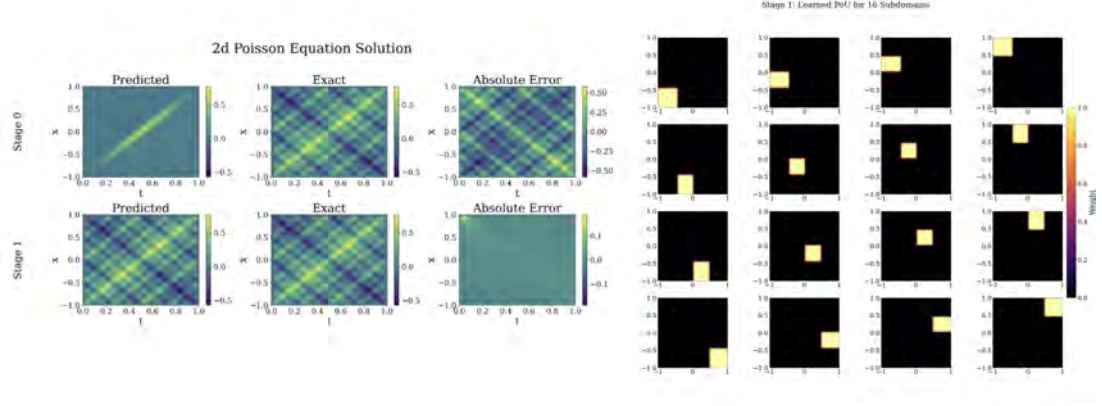


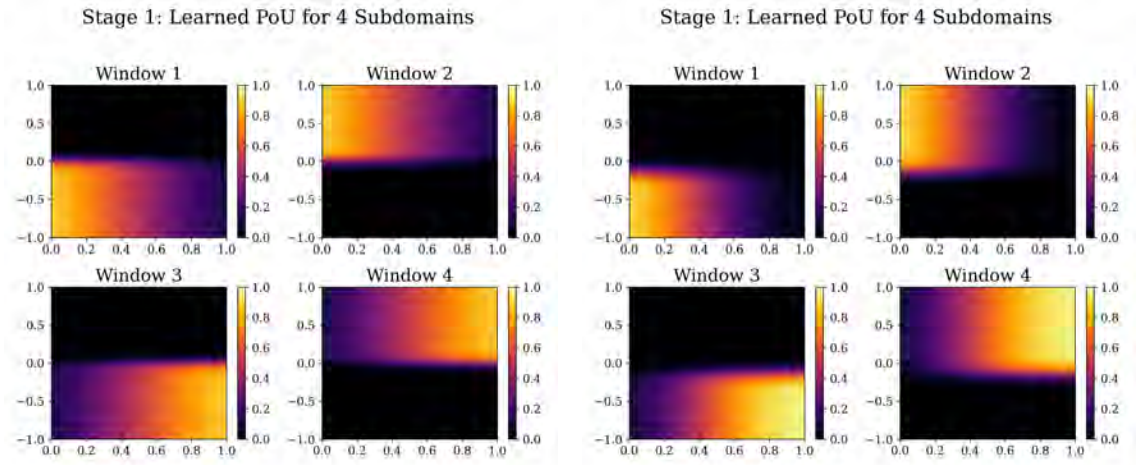
Figure 7.23: Results for the 2D Poisson equation with multiscale features for Case 3 with $\omega = [1, 2, 4, 8, 16]$. . On the left, we have predicted, exact, and absolute error for both Stage 0 (baseline PINN) and Stage 1 (ADD-FBPINN). On the right, we see the learned domain decomposition used in Stage 1. In this case, the subdivision favors more subdomains with sharper transitions.

Table 7.9: Relative L2 errors for a standard PINN, FBPINN, and ADD-FBPINN on Burgers equation. Results are shown for two test cases, one where the shock is located at $x = 0$ (Equation 7.32), and one where the shock is located at $x = -0.25$ (Equation 7.33).

Sharp gradient location	PINN	FBPINN	Adaptive DD FBPINN	Number of subdomains
$x = 0$	9.85×10^{-2}	4.59×10^{-2}	1.26×10^{-3}	4
$x = -0.25$	1.48×10^{-2}	3.11×10^{-2}	2.75×10^{-3}	4

7.2.3 Conclusions

We have developed an adaptive domain decomposition framework for FBPINNs in which the window functions are learned rather than prescribed. POUnets are trained on an initial coarse PINN solution and the window functions learned are subsequently used as the subdomains for FBPINN training. No data is needed for the method, and it is purely physics-informed. We have introduced separable POUnets to enforce regular shaped subdomains and a Laplacian penalty term to enforce wide enough overlapping regions between subdomains. Across three problems, the high frequency 2D Helmholtz equation, multiscale 2D Poisson equation, and viscous Burgers equation, the adaptive method consistently outperforms a baseline PINN and fixed FBPINN.



((a)) Subdomains learned for Burgers Equation 7.32 with sharp gradient at $x = 0$. ((b)) Subdomains learned for Burgers Equation 7.33 with sharp gradient at $x = -0.25$.

Figure 7.24: Subdomains learned for various cases of Burgers equation by a POU net.

The ADD-FBPINN recognizes key features of the solution and proposes subdomains such that the challenging regions of the problem are shared across multiple subdomains. Further, with the Laplacian penalty term, the adaptive approach can efficiently find the optimal overlap width between subdomains.

In future work we will consider adding local normalization of the input to the Laplacian penalty term, similar to the input normalization in FBPINNs that ensures the relative frequency each subdomain sees is low. We note that the size of the subdomains shrinks as the number of subdomains increases. When subdomains are smaller, the sharp transitions between subdomains learned by the POU loss occur over shorter distances, resulting in window functions having larger second derivatives. The Laplacian penalty term, then, grows as the size of subdomains decrease, which results in wider overlaps for small subdomains. At high subdomain counts, this can result in collapses of the subdomains. To further balance the sharpness enforced by the POU net with the overlap regularization, one can normalize the input of the Laplacian penalty term to $[-1, 1]$. Normalization allows the POU net to learn a larger number of subdomains without overlaps swallowing the unique subdomain territory.

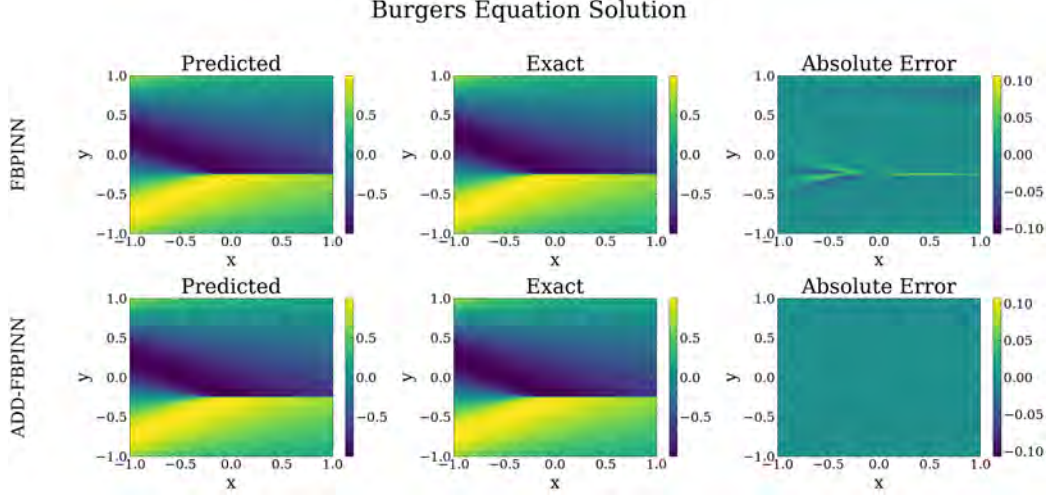
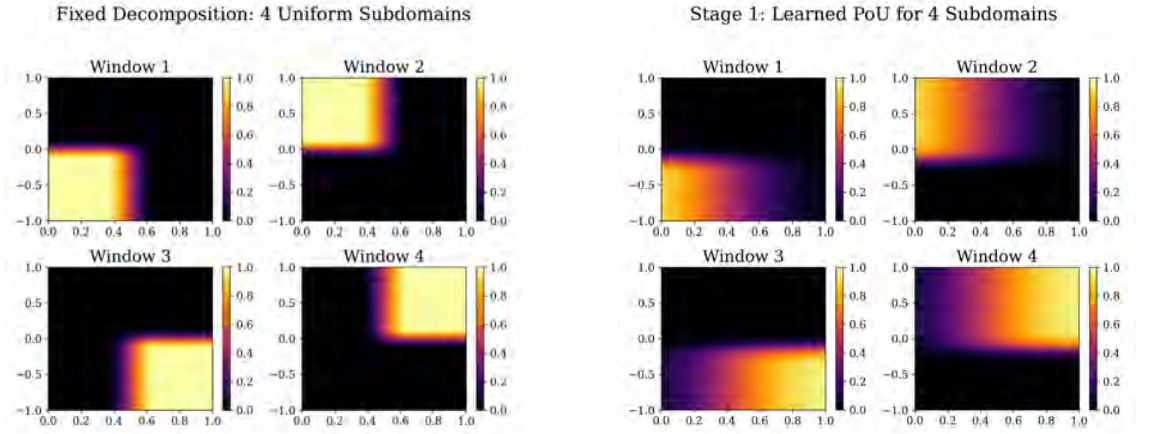


Figure 7.25: Predicted and exact solution and absolute error for Burgers equation comparing fixed domain decomposition with a standard FBPINN to that of an ADD-FBPINN.

This could be beneficial when the complexity of a solution demands a higher number of subdomains to reach sufficient accuracy, such as the highly anisotropic cases of the 2d Helmholtz problem.

Although the ADD-FBPINN framework can learn irregular shaped domains with the MLP POUnet, regular subdomains yielded the highest FBPINN accuracy for the problems studied, likely due to the underlying PDEs having axis-aligned solution features. Additionally, the current implementation treats learned subdomains as a rectangle that covers the extent of the subdomain for normalization. This provides sufficient normalization for approximately rectangular subdomains, but may be less suitable for irregular subdomains. Developing a normalization approach that considers the geometry of an irregular subdomain is an additional direction for future work.

In the context of this dissertation, this chapter highlighted the benefit of applying classical domain decomposition approaches to PINNs and KANs to improve their numerical stability for problems with complex geometries and high frequency features. In the following chapter, we explore the intersection of all three pillars of trustworthy



((a)) Fixed domain decomposition used for Burgers Equation 7.32 with sharp gradient at $x = -0.25$.

((b)) Subdomains learned for Burgers Equation 7.33 with sharp gradient at $x = -0.25$.

Figure 7.26: Comparison between the fixed domain decomposition subdomains and the learned subdomains for Burgers equation with shock located at $x = -0.25$. This case highlights the ability of the adaptive domain decomposition to perform feature-aligned domain decomposition automatically.

modeling in the machine learning setting. We explore an approach for adaptively selecting the loss weights to balance the loss landscape of physics-informed deep operator networks, in which our method offers a tradeoff in accuracy at the benefit of reduced computational cost.

7.3 Model Hyperparameters

Table 7.10: Ablation Study Parameters

FBPINN Parameter	1d Sine ODE	2d Helmholtz	2d Poisson
Learning rate	0.001	0.0001	0.0001
Network size	[1, 64, 64, 64, 64, 1]	[2, 64, 64, 64, 64, 1]	[2, 64, 64, 64, 64, 1]
Activation function	tanh	tanh	tanh
Collocation points	2,000	40,000	40,000
Batch size	1,000	10,000	10,000
Iterations	40,000	40,000	40,000
RAD resample every	10,000	10,000	10,000
Subdomain count per dim	15	5	5

Table 7.11: PINN Parameters

PINN Parameters	2d Helmholtz	2d Poisson	Burgers
Learning rate	0.0001	0.0001	0.0001
Network size	[2, 64, 64, 64, 64, 1]	[2, 64, 64, 64, 64, 1]	[2, 64, 64, 64, 64, 1]
Activation function	tanh	tanh	tanh
Collocation points	40,000	40,000	40,000
Batch size	10,000	10,000	10,000
Iterations	40,000	40,000	40,000

Table 7.12: Fixed DD FBPINN Parameters

FBPINN Parameters	2d Helmholtz	2d Poisson	Burgers
Learning rate	0.0001	0.0001	0.0001
Network size	[2, 64, 64, 64, 64, 1]	[2, 64, 64, 64, 64, 1]	[2, 64, 64, 64, 64, 1]
Activation function	tanh	tanh	tanh
Collocation points	40,000	40,000	40,000
Batch size	10,000	10,000	10,000
Iterations	40,000	40,000	40,000
Number of subdomains	Varies	Varies	Varies

Table 7.13: ADD-FBPINN Parameters

FBPINN Parameters	2d Helmholtz	2d Poisson	Burgers
Learning rate	0.0001	0.0001	0.0001
Network size	[2, 64, 64, 64, 64, 1]	[2, 64, 64, 64, 64, 1]	[2, 64, 64, 64, 64, 1]
Activation function	tanh	tanh	tanh
Collocation points	40,000	40,000	40,000
Batch size	10,000	10,000	10,000
Iterations	40,000	40,000	40,000
RAD resampling every	10,000	10,000	10,000

Table 7.14: ADD-FBPINN POUnet Parameters

POUnet Parameters	2d Helmholtz	2d Poisson	Burgers
Learning rate	0.001	0.001	0.001
Architecture	Separable RBF	Separable RBF	Separable RBF
Iterations	2,000	2,000	2,000
Collapse threshold	$\tau = 0.8$	$\tau = 0.8$	$\tau = 0.8$
POU schedule	[1,4,9,16,25,36,49,64,81,100]	[1,4,9,16,25,36,49,64,81,100]	[1,4,9,16,25,36,49,64,81,100]

CHAPTER 8: EFFICIENT AND ACCURATE ADAPTIVE LOSS WEIGHTING FOR PHYSICS-INFORMED DEEP OPERATOR NETWORKS

In this chapter, we study accuracy and computational cost in the context of physics-informed deep operator networks. This work focuses on how accuracy can be improved by optimizing the balance of loss terms in a PI-DeepONet, and how the balance of the modeling pillars shifts when doing so. One direction of this research was to explore the possibilities of physics-informed operator learning, which has recently received a great deal of focus due to their ability to represent maps between infinite-dimensional Banach spaces [83, 86, 85, 84]. PI-DeepONets provide a perfect landscape for this study because they are extremely costly to train, but are desirable due to their ability to solve problems with little to no data.

Deep operator networks have shown success in a wide range of applications, see, for example, [163, 164, 165, 166, 167, 168]. Physics-informed DeepONets can train to satisfy the solution of a differential equation, in a manner analogous to PINNs [169, 170], with successful applications including crack propagation [171], heat conduction [167], and instability-wave prediction [172].

While PINNs and physics-informed DeepONets [169, 170] have shown tremendous promise across a wide range of applications, e.g., [171, 172, 167, 58], training PI-DeepONets can be difficult and can result in large errors when compared with exact solutions [89]. Recent work on efficient methods for improving the accuracy of PINNs and PI-DeepONets include schemes that implement causality [173], adaptive point selection methods [174], iterative methods to successively reduce the prediction in errors [175, 176, 177, 178, 179], and adaptive weighting schemes [151, 180, 152]. For a summary of advancements in training in PINNs, see [181].

In response to the challenge of physics-informed training, adaptive loss weighting has been shown to be crucial for training both PINNs and DeepONets [79]. Adaptively updating the loss weights improves training by incentivizing the network to balance contributions from the various loss terms. For PINNs, loss weights have been used to address gradient pathologies [182], enforce temporal causality [173, 112], and to balance residual decay rates [152]. Max-min approaches to loss weight selection have also shown promise [183] and facilitated the development of auxiliary networks that dynamically update loss weights [184, 185]. Despite the variety of existing approaches, these methods often depend on empirical findings, are task-specific, or require additional networks. In contrast, one method that has emerged for increasing training accuracy is the use of the Neural Tangent Kernel (NTK) in a weighting scheme in the loss function in PINNs [79] and PI-DeepONets [89].

The neural tangent kernel was presented in [186] as a theoretical framework for understanding the dynamics of a network during gradient descent. This work linked kernel regression to training neural networks by showing that, in the infinite width limit, a neural network converges to a deterministic kernel during training. Further, this work showed that the eigenvalues of the NTK characterize the rate at which the loss function converges, which was then exploited in a loss weight selection algorithm proposed in [79] and [89]. The NTK has been used to improve training for physics-informed DeepONets in practice [187], but a limiting disadvantage of this approach is the increase in computation time required. While the use of the NTK significantly increases the accuracy of a network, it greatly increases the computational cost at each iteration.

In this work, we present the use of the conjugate kernel (CK), an efficient approximation of the NTK, for adaptive weighting for training PI-DeepONets. While significantly less expensive to compute, we show that the CK provides almost equally accurate results for PI-DeepONets. We show that the CK can be used to offset the

training time increase caused by the computation of the NTK for a range of problems. To provide further best practices for utilizing the CK, we highlight a trend that emerges within the spectra of the CK during training. It has been shown for the NTK that a more isotropic kernel spectrum corresponds to improved loss convergence during training [?]. The evolution of the NTK spectrum during training was also utilized to compare the performance of PINNs with physics-informed Kolmogorov-Arnold networks [188], suggesting that less variation in the spectrum across iterations indicates more accurate and efficient loss convergence. Our work confirms these results via comparison of the NTK and CK spectra. We show that, in cases where a DeepONet finds an accurate solution operator, the eigenvalues of the CK or NTK matrix quickly (within 1000 iterations of training) stabilize. In cases where the DeepONet fails to train, the eigenvalues take longer to converge to a constant spread. This further suggests that eigenvalues during training can be used to highlight cases where the more computationally efficient CK might not perform sufficiently, and switching to the NTK is required.

8.1 The Neural Tangent Kernel and Conjugate Kernel

In this section, we detail the Neural Tangent Kernel and Conjugate Kernel for DeepONets.

The NTK (often called the empirical NTK) for a deep neural network (DNN) or a DeepONet is defined as the Gram matrix of the Jacobian of the DNN with respect to the network parameters [186], originally defined for infinitely-wide neural networks (NNs). The NTK after training a finite-width NN is referred to as the after kernel [189]. The CK [190, 191] is a “0-order” approximation to the NTK, defined as the Gram matrix of the Jacobian associated with the parameters of the last layer of a DNN [?]. Here, we are only interested in *kernels after training*, and so both the NTK and CK will denote empirical kernels obtained from finite-width NNs after some amount of training.

Mathematically, the NTK is represented as the kernel induced by the map

$$\Phi_{\text{NTK},k} := \frac{\partial f_{\text{NN}}^\theta}{\partial \theta_k}, \quad k = 1, 2, \dots, |\theta| \quad (8.1)$$

for a feedforward neural network f_{NN}^θ with trainable parameters θ . The (empirical) NTK is then given by

$$\text{NTK}(z, \tilde{z}) := \Phi_{\text{NTK}}(z)^\top \Phi_{\text{NTK}}(\tilde{z}). \quad (8.2)$$

Denoting the parameters in the last layer of f_{NN}^θ by θ^L , the CK is the kernel induced by the map

$$\Phi_{\text{CK},k} := \frac{\partial f_{\text{NN}}^\theta}{\partial \theta_k^L}, \quad k = 1, 2, \dots, |\theta^L|. \quad (8.3)$$

The CK is then defined as

$$\text{CK}(z, \tilde{z}) := \Phi_{\text{CK}}(z)^\top \Phi_{\text{CK}}(\tilde{z}). \quad (8.4)$$

If we define E as the contribution to the NTK for all but the last layer, we can write the relationship between the NTK and CK as $\text{NTK}(z, \tilde{z}) = \text{CK}(z, \tilde{z}) + E(z, \tilde{z})$.

The NTK has been shown to significantly improve the convergence of physics-informed training, e.g., [79, 89, 177, 192], however it is extremely computationally expensive to compute, thereby greatly increasing the training time of the NN or operator network. The CK has been shown to be an efficient approximation of the NTK, resulting in improved robustness, better conditioning, and increased accuracy in some test cases [80].

8.2 Results

8.2.1 Wave equation

We can generalize the wave equation presented in Sec. ?? to the operator learning problem by taking the general initial condition $u(x) = \sum_{n=1}^5 b_n \sin(n\pi x)$. To generate each initial condition u we generate a set $\{b_n\}_{n=1}^5$ of normally distributed random variables. We train with 1000 random initial conditions as the training set. The exact solution is given by $s(x, t) = \sum_{n=1}^5 b_n \sin(n\pi x) \cos(n\pi\sqrt{2}t)$.

In Table ?? we report the relative ℓ_2 errors between the exact solution and results. We note that the CK error is very close to the NTK error, however, it takes only about 50% of the computational time to achieve results of this accuracy. While the CK does not offer additional accuracy over the NTK in most cases, the lower computational time is a compelling argument for why one may consider using it. We plot the predicted solution for a representative initial condition and the distribution of weights throughout the problem domain in Fig. 8.2, which highlights that the CK weights target generally the same regions as the NTK weights.

To further inspect the performance of the CK as an alternative to the NTK, we compare the eigenvalues of each kernel at the end of training, also seen in Fig. 8.2. In particular, we plot the eigenvalues of the submatrices of the NTK/CK that correspond to particular loss terms, in this case, the residual, initial conditions, and boundary conditions. We will see across example problems that the eigenvalues of the CK appear to be roughly scaled down versions of the NTK eigenvalues. Further, we plot the eigenvalues at regular intervals during training in Fig. 8.3. We see that, in this case where the NTK and CK both train successfully, the eigenvalues quickly converge to some stabilized location. In the next section, we will see a case where this behavior is not present, indicating a failure to train.

	Fixed	NTK ($\alpha = 1.0$)	CK ($\alpha = 1.0$)	NTK ($\alpha = 0.5$)	CK ($\alpha = 0.5$)
Mean error	2.84% $\pm$ 1.87%	20.29% $\pm$ 9.14%	0.72% $\pm$ 0.60%	1.43% $\pm$ 0.88%	1.46% $\pm$ 0.92%
Runtime	2:36:53	6:24:41	3:08.46	6:24:10	3:08:33

Figure 8.1: Wave equation: Relative testing errors at 500 initial conditions, averaged over 5 different network initializations, and average elapsed time to run on a NVIDIA A100 GPU.

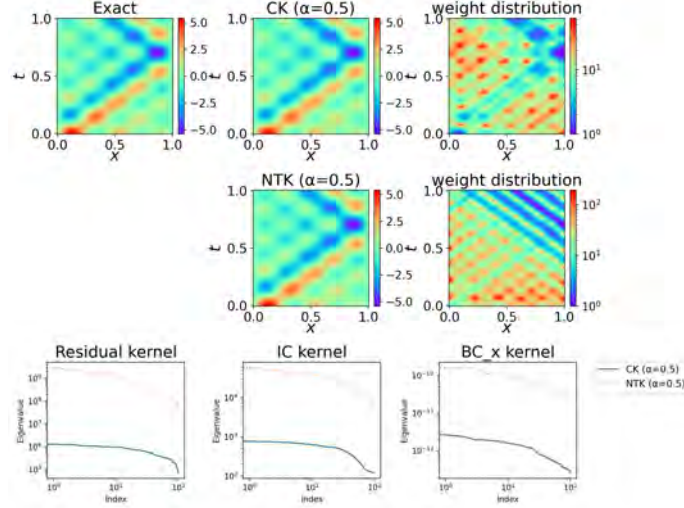


Figure 8.2: Results using a DeepONet to solve the wave equation. Top Row: Exact solution (left), Predicted solution using the CK (middle), distribution of weights throughout the space-time domain (right). Middle row: Predicted solution using the NTK ($\alpha = 0.5$) (middle) and distribution of weights throughout the space-time domain. Bottom row: Eigenvalues of the sub-matrices of the NTK and CK corresponding to various loss terms (residual, initial conditions, and boundary conditions, respectively) calculated at the end of training.

8.2.2 Burgers equation

The viscous one-dimensional Burgers equation with viscosity ν and periodic boundary conditions presented in [89, 169] presents a challenge for DeepONets. The system

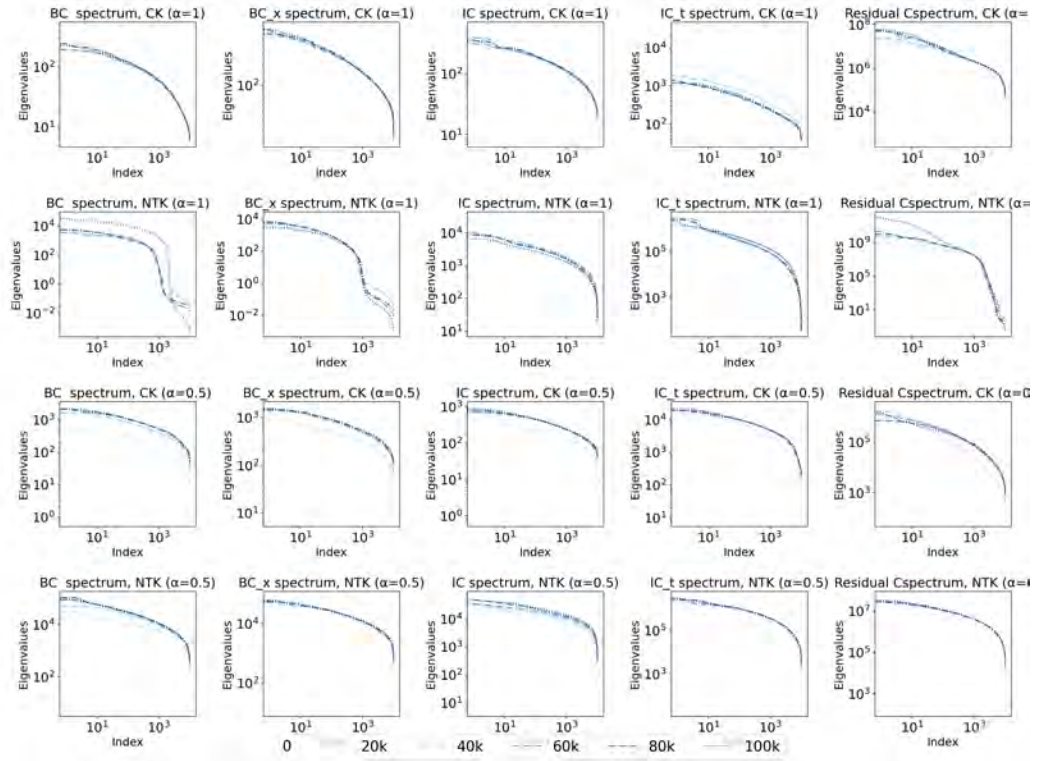


Figure 8.3: Spectrum of various kernels at regular intervals during training for the wave equation solved with a PI-DeepONet

is given by

$$\frac{\partial s}{\partial t} + s \frac{\partial s}{\partial x} - \nu \frac{\partial^2 s}{\partial x^2} = 0, \quad (x, t) \in [0, 1] \times [0, 1] \quad (8.5)$$

$$s(x, 0) = u(x), \quad x \in [0, 1], \quad (8.6)$$

$$s(0, t) = s(1, t), \quad t \in [0, 1], \quad (8.7)$$

$$\frac{\partial s}{\partial x}(0, t) = \frac{\partial s}{\partial x}(1, t), \quad t \in [0, 1], \quad (8.8)$$

where ν is the viscosity. To provide a direct comparison with [89], we follow the procedure as outlined to generate a training and testing set by selecting initial conditions $u(x)$ from a Gaussian random field, and train using the same network hyperparameters. The goal is to learn the solution operator mapping a set of initial conditions $u(x)$ to the set of associated solutions $s(x, t)$. The approximate solution given by the DeepONet is denoted by $\mathcal{G}_\theta$.

Results comparing the CK, the NTK, and fixed weights are shown in Table 8.1. The use of the CK not only leads to comparable or improved test accuracy compared to the NTK but requires only 42.3% of the training time.

Performance of the DeepONet when $\nu = 0.0001$ highlights a scenario in which the CK is not consistently strong enough to capture the full dynamics of the solution, with training succeeding only in some cases. To illustrate the performance of the CK weighting scheme accross multiple trials, we plot testing error box plots in Fig. 8.4. The CK results are more varied across multiple trained networks when compared to the NTK. To this point, we plot the spectrum of the kernels after training in Fig. 8.5, Fig. 8.6, Fig. 8.7. It has been shown that the eigenvalues of the NTK, and subsequently the CK, characterize the rate at which training error decreases [79]. In most cases, the eigenvalues of CK kernels appear to be "scaled-down" versions of the eigenvalues for the corresponding NTK kernels. However, in the case of $\nu = 0.0001$, the kernels computed for the initial conditions loss term break this trend.

We also inspect the behavior of kernel eigenvalues during training to gain insights on the performance of the CK weighting scheme. One might expect that in cases where training is robust and efficient, the eigenvalues remain mostly unchanged during training. We plot the evolution of kernel eigenvalues during training in Fig. 8.8 and Fig. 8.9. It is evident that, in the case of $\nu = 0.01$ in which the DeepONets consistently train well, the eigenvalues quickly converge to a stable spread (Fig. 8.8). In contrast, in the more challenging case where $\nu = 0.0001$, the eigenvalues take longer to reach their stable values (Fig. 8.9).

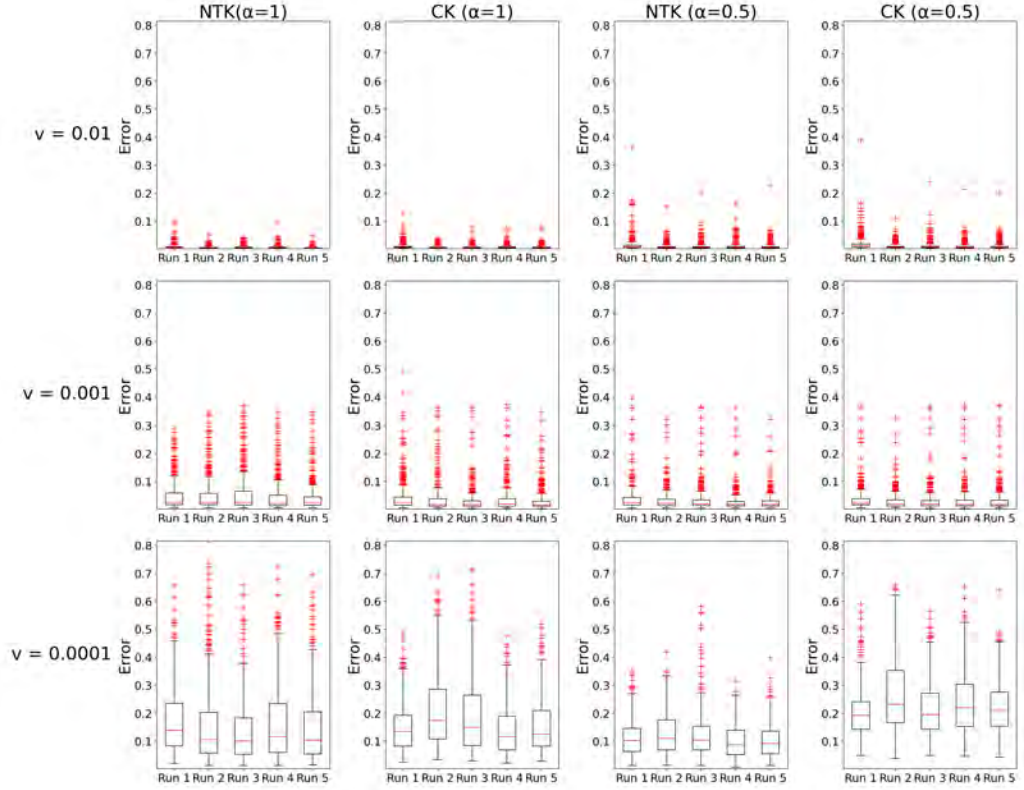


Figure 8.4: Burgers Equation: Testing errors on 500 initial conditions, with results for 5 randomly initialized networks

8.2.3 Stokes flow

It is shown in [89] that 2-dimensional Stokes flow over an object presents a challenge for DeepONets. In fact, convergence is only achieved when the modified DeepONet structure is used along with the NTK. Upon investigating the application of the CK

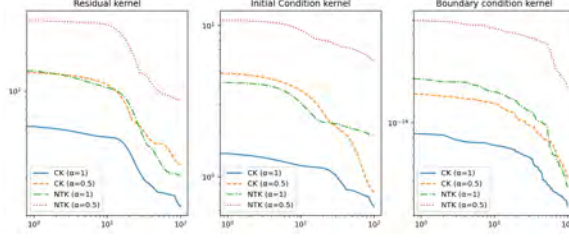


Figure 8.5: Burger's equation spectrum at the end of training with $\nu = 0.01$ for the sub-matrices of the NTK or CK corresponding to the residual, initial condition, or boundary condition loss terms, respectively.

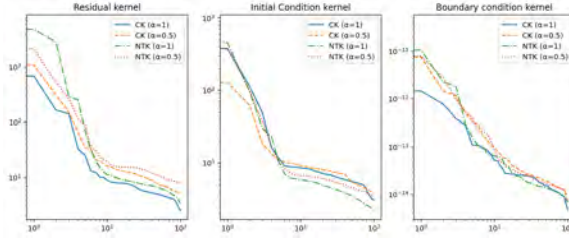


Figure 8.6: Burger's equation spectrum at the end of training with $\nu = 0.001$ for the sub-matrices of the NTK or CK corresponding to the residual, initial condition, or boundary condition loss terms, respectively.

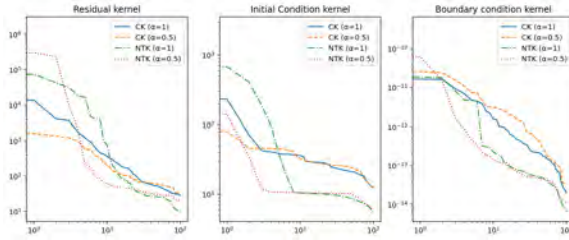


Figure 8.7: Burger's equation spectrum at the end of training with $\nu = 0.0001$ for the sub-matrices of the NTK or CK corresponding to the residual, initial condition, or boundary condition loss terms, respectively.

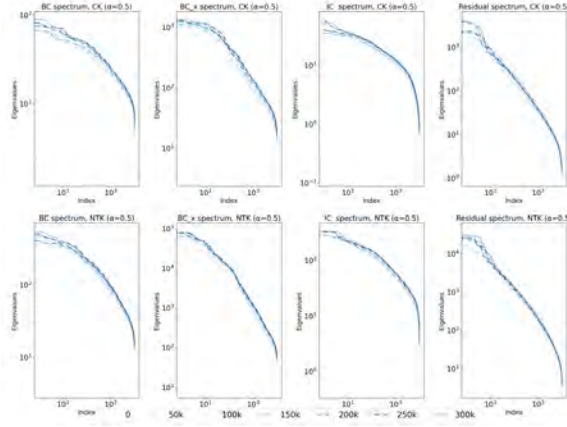


Figure 8.8: Eigenvalues of kernels during training for Burgers equation with viscosity $\nu = 0.01$. The eigenvalues in all cases converge quickly to stabilized value, indicating accurate and efficient training.

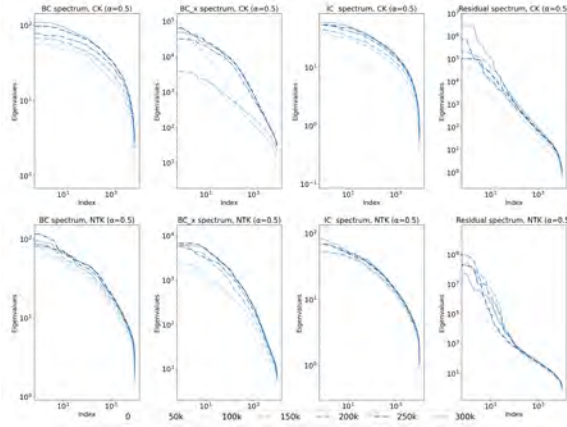


Figure 8.9: Eigenvalues of kernels during training for Burgers equation with viscosity $\nu = 0.0001$. The eigenvalues during training are more dispersed and stabilize slower, if at all, when compare to other cases. This highlights a difficulty in training.

to the same problem, we uncover a strategy for reducing computation time even in difficult-to-train cases. As such, we follow the problem setup in [89] to show a direct comparison for the CK.

In this example we consider Stokes flow of a fluid through a channel from left to right with an inlet and outlet, bounded above and below by walls with an obstacle in the middle of the computational domain. Our goal is to learn the operator $\mathcal{G}$ mapping different obstacle shapes to the associated PDE solution (u, v, p) , where $\mathbf{u} = (u, v)$ represents the velocity field, and p is the pressure. The associated PDE is given by

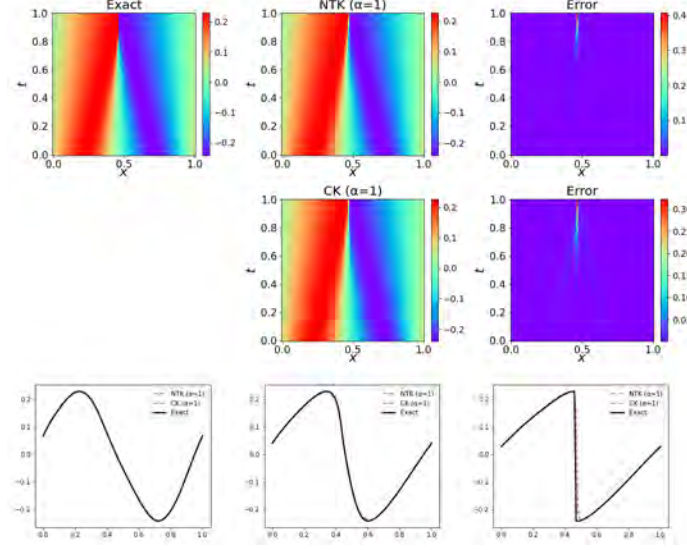


Figure 8.10: Sample results for Burgers equation with $\nu = 0.0001$

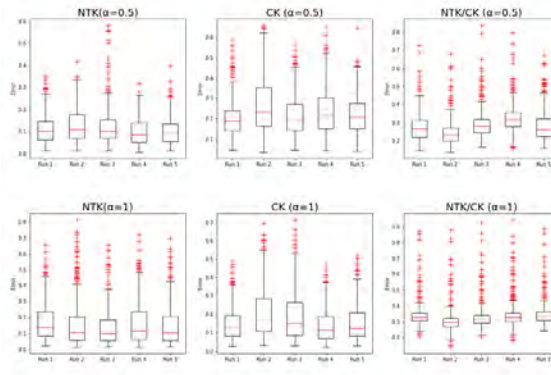


Figure 8.11: Burgers Equation: Testing errors on 500 initial conditions for $\nu = 0.0001$, with results for 5 randomly initialized networks. This plot includes errors for when the combination of NTK and CK is applied.

$$-\Delta \mathbf{u} + \nabla p = 0, \quad (x, y) \in \Omega \setminus \Gamma \quad (8.9)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (x, y) \in \Omega \setminus \Gamma \quad (8.10)$$

$$\mathbf{u} = \mathbf{0}, \quad (x, y) \in \Lambda_1 \cup \partial\Gamma \quad (8.11)$$

$$\mathbf{u} = \mathbf{g}, \quad (x, y) \in \Lambda_2 \quad (8.12)$$

$$p = 0, \quad (x, y) \in \Lambda_3 \quad (8.13)$$

Table 8.1: Results for Burgers equation for a range of viscosities ν . The mean and standard deviation of the relative ℓ_2 error are over 500 initial conditions in the test set. Average run times are computed using an NVIDIA A100 GPU.

ν	Fixed weights	NTK weights	CK weights
0.01	$3.65\% \pm 6.26\%$	$0.63\% \pm 0.50\%$ ($\alpha = 1$)	$0.66\% \pm 0.57\%$ ($\alpha = 1$)
0.001	$9.18\% \pm 8.03\%$	$2.80\% \pm 3.00\%$ ($\alpha = 0.5$)	$3.04\% \pm 3.82\%$ ($\alpha = 0.5$)
0.0001	$25.10\% \pm 9.53\%$	$11.44\% \pm 5.91\%$ ($\alpha = 0.5$)	$8.34\% \pm 3.62\%$ ($\alpha = 1$)
Run time (h)	1.582	6.071	2.568

where Δ is the Laplace operator, $\Omega = [0, 1] \times [0, 1]$ denotes the computational domain and Γ represents the obstacle, with boundary $\partial\Gamma$. The wall boundaries are given by Λ_1 , the inlet is given by Λ_2 and outlet by Λ_3 . The boundary conditions include a no-slip condition at the wall and obstacle boundaries, an initial velocity at the inlet of $\mathbf{g} = (\sin(\pi y), 0)$, and zero-pressure at the outlet. We aim to learn the operator $\mathcal{G}$ mapping obstacles $\partial\Gamma$ to PDE solutions (u, v, p) where the obstacles are ellipses centered at $(\frac{1}{2}, \frac{1}{2})$ with varying axis lengths $a, b > 0$:

$$\partial\Gamma(\phi) = \left(a \cos(\phi) + \frac{1}{2}, b \sin(\phi) + \frac{1}{2} \right), \quad \phi \in [0, 2\pi). \quad (8.14)$$

The DeepONet produces an approximate solution operator given by $\partial\Gamma \xrightarrow{\mathcal{G}_\theta} [\mathcal{G}_\theta^{(u)}, \mathcal{G}_\theta^{(v)}, \mathcal{G}_\theta^{(p)}]$ via minimizing the loss function 7.14:

$$\mathcal{L}(\theta) = \frac{2}{N^*} \sum_{i=1}^{N^*} \lambda_i \left| \mathcal{T}^{(i)}(\partial\Gamma^{(i)}, \mathcal{G}_\theta(\partial\Gamma^{(i)})(x_i, y_i)) \right|^2, \quad (8.15)$$

where $\mathcal{T}^{(i)}$ denotes the operators in the loss function; either enforcing the physics or boundary conditions. We train with a batch size of $N = 1000$ input closed curves $\{\partial\Gamma^{(i)} = [\partial\Gamma(\phi_1)^{(i)}, \partial\Gamma(\phi_2)^{(i)}, \dots, \partial\Gamma(\phi_m)^{(i)}]_{i=1}^N$ evaluated at evenly spaced grid points $\{\phi\}_{i=1}^m \in [0, 2\pi]$. For each input curve, we calculate the loss with training data sets $\{(x_{r,j}^{(i)}, y_{r,j}^{(i)})\}_{i=1}^Q$, $\{(x_{bc1,j}^{(i)}, y_{bc1,j}^{(i)})\}_{i=1}^P$, $\{(x_{bc2,j}^{(i)}, y_{bc2,j}^{(i)})\}_{i=1}^P$, $\{(x_{bc3,j}^{(i)}, y_{bc3,j}^{(i)})\}_{i=1}^P$ for

the residual and boundary conditions, calculated on $\Omega \setminus \Gamma$, $\Lambda_1 \cup \partial\Gamma$, Λ_2 , and Λ_3 respectively. For training, we set $P = 100$ and $Q = 2500$. In the composite loss equation 8.2.3, $N^* = 2NQP$. To generate the training dataset, a, b are randomly sampled from a uniform distribution $\mathcal{U}(0.05, 0.3)$ which creates our set of input curves $\{\partial\Gamma^{(i)}\}_{i=1}^N$ according to 8.2.3. The testing data set is generated in the same way for an additional 500 curves.

The individual loss terms are as follows. Residual loss is given by

$$\mathcal{L}_{res}^{(k)}(\theta) = \frac{1}{NQ} \sum_{i=1}^N \sum_{j=1}^Q \left| \mathcal{R}_\theta^k(\partial\Gamma^{(i)})(x_{r,j}^{(i)}, y_{r,j}^{(i)}) \right|^2, k = 1, 2, 3. \quad (8.16)$$

with individual PDE residuals calculated via

$$\mathcal{R}_\theta^{(1)}(\partial\Gamma)(x, y) = -\Delta\mathcal{G}_\theta^{(u)}(\partial\Gamma)(x, y) + \frac{\partial\mathcal{G}_\theta^{(p)}(\partial\Gamma)(x, y)}{x}, \quad (8.17)$$

$$\mathcal{R}_\theta^{(2)}(\partial\Gamma)(x, y) = -\Delta\mathcal{G}_\theta^{(v)}(\partial\Gamma)(x, y) + \frac{\partial\mathcal{G}_\theta^{(p)}(\partial\Gamma)(x, y)}{y}, \quad (8.18)$$

$$\mathcal{R}_\theta^{(3)}(\partial\Gamma)(x, y) = \frac{\partial\mathcal{G}_\theta^{(u)}(\partial\Gamma)(x, y)}{x} + \frac{\partial\mathcal{G}_\theta^{(v)}(\partial\Gamma)(x, y)}{y}. \quad (8.19)$$

Loss for the boundary conditions are then calculated via

$$\mathcal{L}_{BC}^{(1)}(\theta) = \frac{1}{NP} \sum_{i=1}^N \sum_{j=1}^P \left[\left| \mathcal{G}_\theta^{(u)}(\partial\Gamma^{(i)})(x_{bc1,j}^{(i)}, y_{bc1,j}^{(i)}) \right|^2 + \left| \mathcal{G}_\theta^{(v)}(\partial\Gamma^{(i)})(x_{bc1,j}^{(i)}, y_{bc1,j}^{(i)}) \right|^2 \right] \quad (8.20)$$

$$\mathcal{L}_{BC}^{(2)}(\theta) = \frac{1}{NP} \sum_{i=1}^N \sum_{j=1}^P \left[\left| \mathcal{G}_\theta^{(u)}(\partial\Gamma^{(i)})(x_{bc2,j}^{(i)}, y_{bc2,j}^{(i)}) - \sin(\pi y_{bc2,j}^{(i)}) \right|^2 + \left| \mathcal{G}_\theta^{(v)}(\partial\Gamma^{(i)})(x_{bc2,j}^{(i)}, y_{bc2,j}^{(i)}) \right|^2 \right] \quad (8.21)$$

$$\mathcal{L}_{BC}^{(2)}(\theta) = \frac{1}{NP} \sum_{i=1}^N \sum_{j=1}^P \left| \mathcal{G}_\theta^{(p)}(\partial\Gamma^{(i)})(x_{bc3,j}^{(i)}, y_{bc3,j}^{(i)}) \right|^2 \quad (8.22)$$

$$(8.23)$$

We train 5 randomly initialized networks and test over 500 input curves. Predicted solutions for a representative boundary condition from the testing set found by using the NTK to select loss weights can be found in Fig. 8.12. For comparison, results using a DeepONet with CK guided weights can be seen in Fig. 8.13. We also test the performance of applying a combination of both CK and NTK weights. Here, we use NTK weights on the residual loss terms, and CK weights on the boundary loss terms. A combination of CK and NTK weights offers improved performance over just using the CK, but still offers a reduction in computation time from using the NTK exclusively. The distribution of testing errors across the testing dataset for each run are illustrated with box plots in Fig. 8.14. The average of average testing errors over 5 networks are depicted in Fig. 8.15 and Fig. 8.16. For Stokes flow, utilizing the CK with a value of $\alpha = 0.5$ performs consistently well across the board, whereas using $\alpha = 1$ is less consistent: sometimes achieving a low error and other times failing to capture the dynamics completely.

For Stokes' flow, the spectrum for the conjugate kernels in the cases that fail to train differ significantly from the spectrum of the corresponding NTK. This is seen in Fig. 8.17, where we plot the eigenvalues of the CK and NTK at the end of training for two cases: one in which a solution is reached and one where training fails. Further, the rate at which the eigenvalues stabilize during training correlates to the success of training, highlighted in Fig. 8.18. Thus, the eigenvalues of the kernels provide insight as to (1) when switching from the CK to NTK weights is necessary, and (2) which loss terms need the added robustness of the NTK.

We also calculate the weight distribution over the domain during training for the PDE residual loss as in [89], with weight distribution plotted in Fig. 8.19:

$$\lambda_r^{(k)}(\partial\mathbf{\Gamma})(x, y) = \frac{\|\mathbf{H}\|_\infty}{\|\nabla_\theta \mathcal{R}_\theta^{(k)}(\partial\mathbf{\Gamma})(x, y)\|_2^2}, k = 1, 2, 3 \quad (8.24)$$

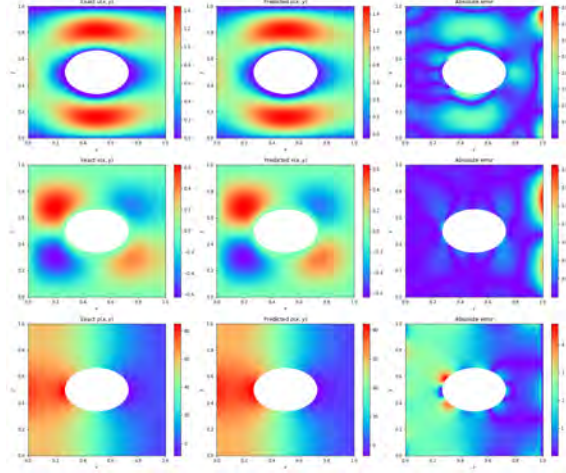


Figure 8.12: Stokes flow sample results exact, predicted, and absolute error for a representative boundary condition from the testing set using the NTK with $\alpha = 0.5$.

8.3 Conclusions

We have shown that the CK is an efficient approximation of the NTK, and can result in similar training accuracy for PINNs and physics-informed DeepONets, while significantly reducing the computational time. In the cases presented, the CK results have errors quite similar to the NTK in most cases, including outperforming the NTK for the smallest viscosity for the viscous Burgers equation. While the accuracy is similar between the two weighting schemes, using the CK rather than the NTK cuts the training time by roughly two thirds, and for this reason, the CK should be considered as an alternative for the NTK when training physics-informed DeepONets.

This work also highlighted the importance of the NTK/CK eigenvalues as a signal during training as to whether the network will reach an accurate solution or not. We showed that, consistent with all example problems, the eigenvalues converge quickly in cases where the network trains successfully, and are more spread out in more challenging cases. This insight can be utilized to flag cases in early training that will require the additional robustness of the NTK.

This work highlighted an important trade-off in computational cost, modeling error and numerical stability. In the next chapter, I synthesize all the results from this

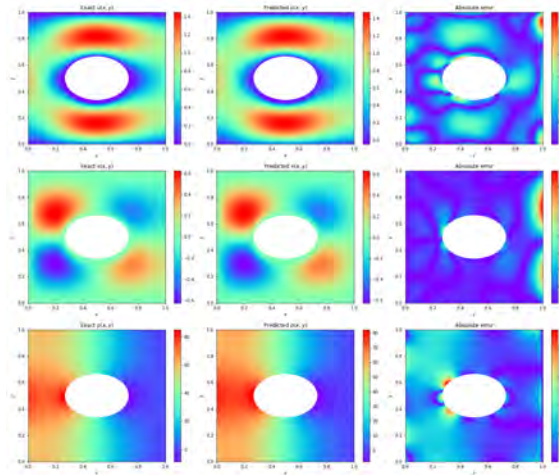


Figure 8.13: Stokes flow sample results exact, predicted, and absolute error for a representative boundary condition from the testing set using the CK with $\alpha = 0.5$.

dissertation to provide a clear picture of the interplay between classical and modern modeling approaches, as well as the importance of prioritizing not only numerical stability, but also modeling error and computational cost when building reliable models.

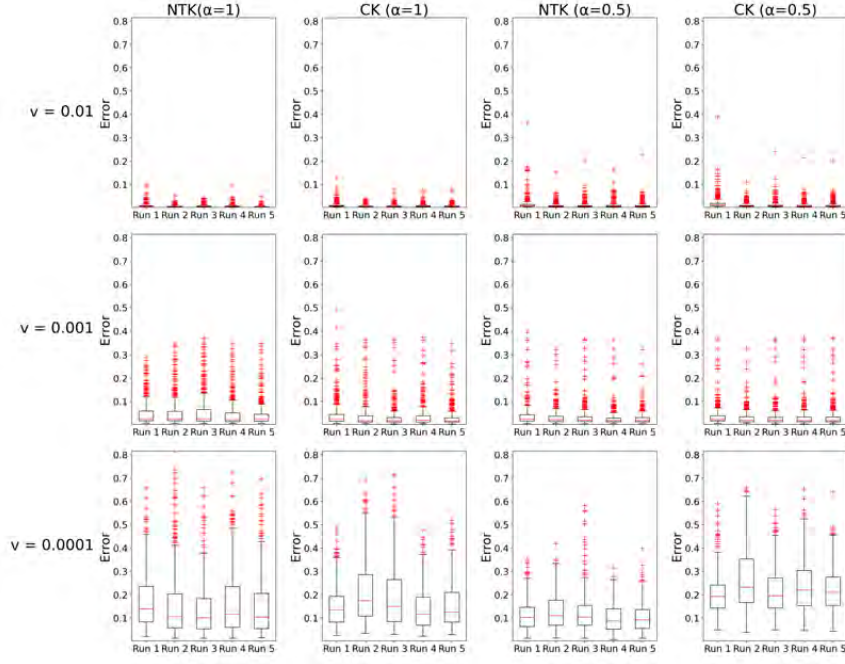


Figure 8.14: Stokes flow: Box plots of testing errors over 100 various obstacles over 5 network runs. The points denoted by red crosses are outliers, which extend past 1.5x the inter-quartile range of the box.

	NTK	NTK/CK	CK
Mean error u	$2.75\% \pm 0.89\%$	$36.49\% \pm 4.75\%$	$60.29\% \pm 4.14\%$
Mean error v	$8.14\% \pm 3.3\%$	$2.02\% \pm 43.52\%$	$0.22\% \pm 76.80\%$
Mean error p	$5.48\% \pm 3.29\%$	$40.31\% \pm 4.46\%$	$68.35\% \pm 4.29\%$
Runtime	2:54:00	2:22:12	1:46:12

Figure 8.15: Stokes flow: Mean testing errors at 500 testing ellipse sizes, averaged over 5 different network initializations, and average elapsed time to run on a NVIDIA A100 GPU. The NTK/CK case computes NTK weights for the residual loss terms, and CK weights for the boundary loss terms. All cases have $\alpha = 1$.

	NTK	NTK/CK	CK
Mean error u	$4.71\% \pm 1.89\%$	$5.31\% \pm 2.92\%$	$3.34\% \pm 1.24\%$
Mean error v	$11.93\% \pm$	$10.61\% \pm$	$8.42\% \pm 1.60\%$
Mean error p	2.41%	2.92%	$5.05\% \pm 1.99\%$
	$7.10\% \pm 2.88\%$	$7.29\% \pm 3.71\%$	
Runtime	2:54:00	2:22:48	1:46:48

Figure 8.16: Stokes flow: Mean testing errors at 500 testing ellipse sizes, averaged over 5 different network initializations, and average elapsed time to run on a NVIDIA A100 GPU. The NTK/CK case computes NTK weights for the residual loss terms, and CK weights for the boundary loss terms. All cases have $\alpha = 0.5$.

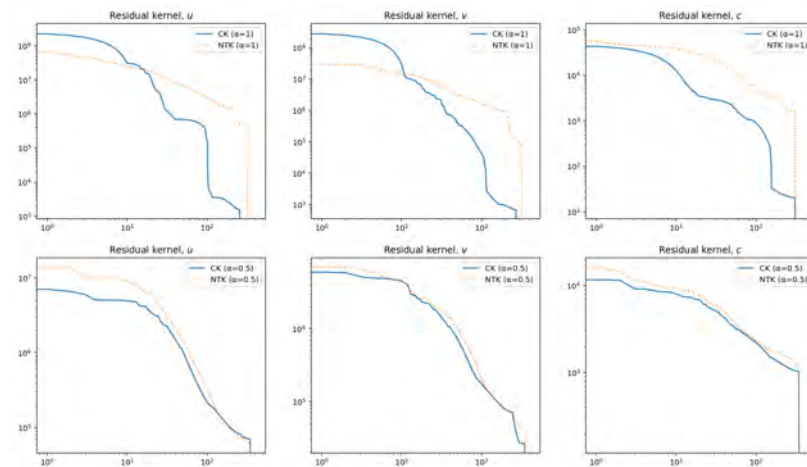


Figure 8.17: Stokes flow. Top row: Spectrum of residual kernels with respect to u, v, c using the NTK and CK for a case in which the DeepONet fails to reach a solution. Bottom row: Spectrum of residual kernels with respect to u, v, c using the NTK and CK for a case in which the DeepONet trains successfully.

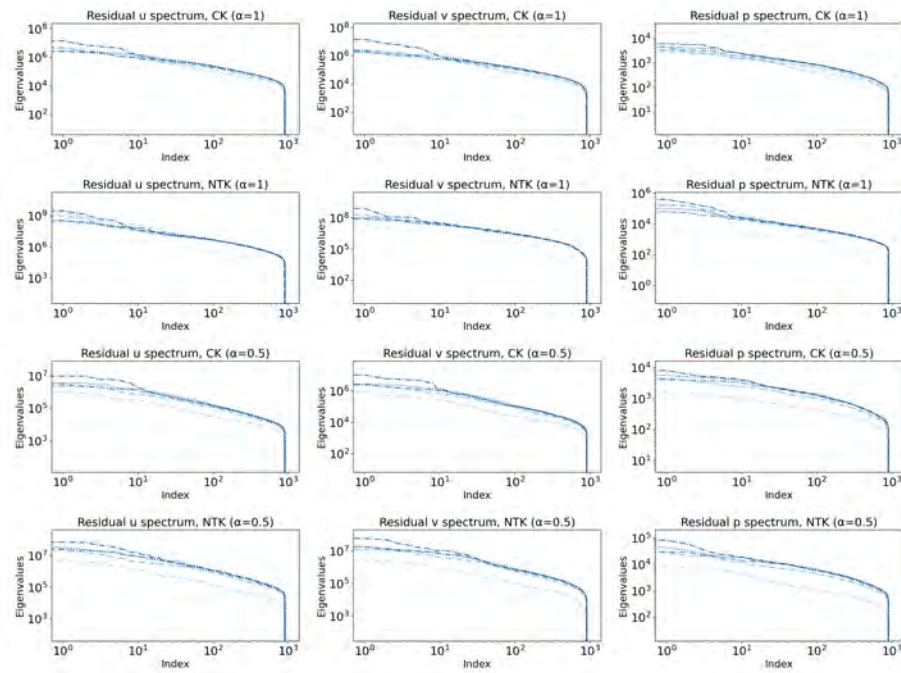


Figure 8.18: Eigenvalues during training for Stokes' flow

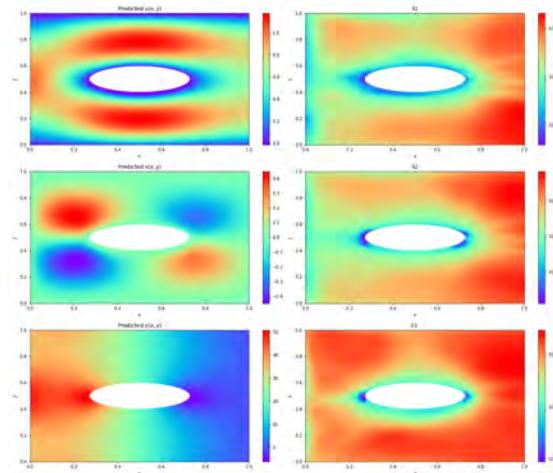


Figure 8.19: Stokes flow sample results for the NTK with $\alpha = 1$ and weight distribution for the residual kernels

CHAPTER 9: DISCUSSION AND FUTURE WORK

The work presented in this dissertation suggests that trustworthy scientific computing requires careful consideration of numerical accuracy, modeling error, and computational cost. Across a range of classical and machine learning-based modeling approaches, we have demonstrated that improving one of these aspects often introduces new considerations for the others. Rather than treating these pillars as independent objectives, reliable scientific computing requires understanding their interactions and balancing them according to the needs of the application.

The first part of this dissertation focused on these modeling objectives through the lens of classical numerical methods and model reduction. In Chapter 2, we showed that classical stability analysis for numerical SDE schemes is dependent on the benchmark chosen, with more sophisticated themes performing worse than simplistic ones depending on the benchmark used. Thus, we established the need for careful consideration when benchmarking novel schemes as well. In Chapter 3, we introduced full probability law modeling error analysis for a hierarchy of coarse-grained approaches, showing the benefits of expanding upon traditional equilibrium stability analysis. Again, we see that there is often more to the picture than standard benchmarking and analysis provide. In Chapter 4, we revisit the coarse-graining analysis, this time through the lens of relaxation time error. In future work, we hope to develop an algorithm that selects optimal resolved variables for coarse-graining that uses this estimate as an objective function.

The second part of this dissertation focused on developing and testing novel machine learning methods for fluid simulations. In Chapter 6, we developed a novel stacking approach for PINNs that leveraged multifidelity training to iteratively refine

the solution. In Chapter 7, we explored the classical approach of domain decomposition to improve the training of PINNs and KANs. And in Chapter 8, we focused on the use of adaptive loss weighting via the conjugate kernel for PI-DeepONets, which allowed us to explore a setting in which accuracy, efficiency, and modeling error have a noticeable trade-off. Future work regarding these projects will be to incorporate more rigorous numerical analysis regarding their performance and also applying these approaches to large-scale problems.

REFERENCES

- [1] P. E. Kloeden, E. Platen, and H. Schurz, *Numerical solution of SDE through computer experiments*. Springer Science & Business Media, 2012.
- [2] Y. SAITO and T. MITSUI, “Discrete approximations for stochastic differential equations,” *Trans. Japan SIAM*, vol. 2, pp. 1–16, 1992. In Japanese.
- [3] A. Rößler, “Second order runge–kutta methods for itô stochastic differential equations,” *SIAM Journal on Numerical Analysis*, vol. 47, no. 3, pp. 1713–1738, 2009.
- [4] C. J. Stephen, B. L. Green, Y. N. D. Lekhai, L. Weng, P. Hill, S. Johnson, A. C. Frangeskou, P. L. Diggle, Y.-C. Chen, M. J. Strain, E. Gu, M. E. Newton, J. M. Smith, P. S. Salter, and G. W. Morley, “Deep three-dimensional solid-state qubit arrays with long-lived spin coherence,” *Phys. Rev. Appl.*, vol. 12, p. 064005, Dec 2019.
- [5] Q. He, M. Perego, A. A. Howard, G. E. Karniadakis, and P. Stinis, “A hybrid deep neural operator/finite element method for ice-sheet modeling,” *Journal of Computational Physics*, vol. 492, p. 112428, 2023.
- [6] J. Wu, Y. Gan, P. Huang, and L. Shen, “A coarse-grained approach to modeling gas transport in swelling porous media,” *International Journal of Rock Mechanics and Mining Sciences*, vol. 183, p. 105918, 11 2024.
- [7] Z. Xiang, D. Cao, J. Lan, W. Wang, and D. P. Broom, “Multiscale simulation and modelling of adsorptive processes for energy gas storage and carbon dioxide capture in porous coordination frameworks,” *Energy Environ. Sci.*, vol. 3, pp. 1469–1487, 2010.
- [8] 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*, vol. 378, pp. 686–707, 2019.
- [9] S. Wang, H. Wang, and P. Perdikaris, “Learning the solution operator of parametric partial differential equations with physics-informed deepnets,” *CoRR*, vol. abs/2103.10974, 2021.
- [10] Y. Liu, J. Zhang, S. Yang, J. li, and Q. Liu, “Prospective on applying machine learning in computational fluid dynamics (cf) simulation of metallurgical reactors,” *Ironmaking Steelmaking: Processes, Products and Applications*, 09 2024.
- [11] Z. Mao, A. Jagtap, and G. Karniadakis, “Physics-informed neural networks for high-speed flows,” *Computer Methods in Applied Mechanics and Engineering*, 03 2020.

- [12] R. I. Mukhamediev, A. Symagulov, Y. Kuchin, K. Yakunin, and M. Yelis, “From classical machine learning to deep neural networks: A simplified scientometric review,” *Applied Sciences*, vol. 11, no. 12, p. 5541, 2021.
- [13] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, “Machine learning and the physical sciences,” *Reviews of Modern Physics*, vol. 91, no. 4, p. 045002, 2019.
- [14] T. M. Khan and A. Robles-Kelly, “Machine learning: Quantum vs classical,” *IEEE Access*, vol. 8, pp. 219275–219294, 2020.
- [15] T. Hudson, X. H. Li, and S. Murphy, “Numerical stability revisited: A family of benchmark problems for the analysis of explicit stochastic differential equation integrators,” 2026.
- [16] G. Robert and R. Tweedie, “Exponential convergence of langevin diffusions and their discrete approximation,” *Bernoulli*, vol. 2, pp. 341–363, 1996.
- [17] J. C. Mattingly, A. M. Stuart, and D. J. Higham, “Ergodicity for sdes and approximations: locally lipschitz vector fields and degenerate noise,” *Stochastic processes and their applications*, vol. 101, no. 2, pp. 185–232, 2002.
- [18] W. Fang and M. B. Giles, “Adaptive euler–maruyama method for sdes with nonglobally lipschitz drift,” *The Annals of Applied Probability*, vol. 30, no. 2, pp. 526–560, 2020.
- [19] C. Kelly and G. J. Lord, “Adaptive euler methods for stochastic systems with non-globally lipschitz coefficients,” *Numerical Algorithms*, vol. 89, no. 2, pp. 721–747, 2022.
- [20] G. J. Lord, *Stochastic Differential Equations*. Wiley-VCH, 2014.
- [21] E. Cancès, F. Legoll, and G. Stoltz, “Theoretical and numerical comparison of some sampling methods for molecular dynamics,” *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 41, no. 2, pp. 351–389, 2007.
- [22] F. Legoll, T. Lelièvre, and U. Sharma, “An adaptive parareal algorithm: application to the simulation of molecular dynamics trajectories,” *SIAM Journal on Scientific Computing*, vol. 44, no. 1, pp. B146–B176, 2022.
- [23] X. Han, P. E. Kloeden, X. Han, and P. E. Kloeden, “Numerical methods for ordinary and stochastic differential equations,” *Random Ordinary Differential Equations and Their Numerical Solution*, pp. 101–108, 2017.
- [24] X. Han, M. Valorani, and H. N. Najm, “Explicit time integration of the stiff chemical langevin equations using computational singular perturbation,” *The Journal of Chemical Physics*, vol. 150, no. 19, 2019.

- [25] X. Han and H. N. Najm, “Modeling fast diffusion processes in time integration of stiff stochastic differential equations,” *Communications on Applied Mathematics and Computation*, vol. 4, no. 4, pp. 1457–1493, 2022.
- [26] H. Schurz, “Preservation of probabilistic laws through euler methods for ornstein-uhlenbeck process,” *Stochastic analysis and applications*, vol. 17, no. 3, pp. 463–486, 1999.
- [27] Y. Saito and T. Mitsui, “Stability analysis of numerical schemes for stochastic differential equations,” *SIAM Journal on Numerical Analysis*, vol. 33, no. 6, pp. 2254–2267, 1996.
- [28] Y. Saito and T. Mitsui, “Mean-square stability of numerical schemes for stochastic differential systems,” *Vietnam J. Math*, vol. 30, pp. 551–560, 2002.
- [29] A. Tocino and M. Senosiain, “Mean-square stability analysis of numerical schemes for stochastic differential systems,” *Journal of Computational and Applied Mathematics*, vol. 236, no. 10, pp. 2660–2672, 2012.
- [30] H. Schurz, “Asymptotical mean square stability of an equilibrium point of some linear numerical solutions with multiplicative noise,” *Stochastic Analysis and Applications*, vol. 14, no. 3, pp. 313–353, 1996.
- [31] G. N. Milstein, *Numerical integration of stochastic differential equations*, vol. 313. Springer Science & Business Media, 2013.
- [32] G. Maruyama, “Continuous Markov processes and stochastic equations,” *Rend. Circ. Mat. Palermo (2)*, vol. 4, pp. 48–90, 1955.
- [33] D. Down, S. P. Meyn, and R. L. Tweedie, “Exponential and uniform ergodicity of markov processes,” *The Annals of Probability*, vol. 23, no. 4, pp. 1671–1691, 1995.
- [34] G. J. Lord, “Analysis of numerical methods suitable for computing lyapunov exponents,” *School of Mathematical Sciences*, 1995.
- [35] S. P. Meyn and R. L. Tweedie, *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.
- [36] L. Hörmander, “Hypoelliptic second order differential equations,” *Acta Math.*, vol. 119, pp. 147–171, 1967.
- [37] R. M. Gray and L. D. Davisson, *An Introduction to Statistical Signal Processing*. Cambridge University Press, 2004.
- [38] W. Rüemelin, “Numerical treatment of stochastic differential equations,” *SIAM Journal on Numerical Analysis*, vol. 19, no. 3, pp. 604–613, 1982.
- [39] K. Itô, P. Henry Jr, *et al.*, *Diffusion processes and their sample paths*. Springer Science & Business Media, 2012.

- [40] N. Ikeda and S. Watanabe, *Stochastic differential equations and diffusion processes*, vol. 24. Elsevier, 2014.
- [41] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, Í. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.
- [42] S. Takada, “Coarse-grained molecular simulations of large biomolecules,” *Current Opinion in Structural Biology*, vol. 22, no. 2, pp. 130–137, 2012. Theory and simulation/Macromolecular assemblages.
- [43] H. I. IngÅ³lfsson, C. A. Lopez, J. J. Usitalo, D. H. de Jong, S. M. Gopal, X. Periole, and S. J. M. A. J. M., –248, 2014.
- [44] W. G. Noid, “Perspective: Coarse-grained models for biomolecular systems,” *The Journal of Chemical Physics*, vol. 139, p. 090901, 09 2013.
- [45] A. Majda, I. Timofeyev, and E. Vanden-Eijnden, “A mathematical framework for stochastic climate models,” *Communications on Pure and Applied Mathematics*, vol. 54, pp. 891 – 974, 08 2001.
- [46] M. Buzzicotti, B. A. Storer, H. Khatr, S. M. Griffies, and H. Aluie, “Spatio-temporal coarse-graining decomposition of the global ocean geostrophic kinetic energy,” *Journal of Advances in Modeling Earth Systems*, vol. 15, no. 6, p. e2023MS003693, 2023. e2023MS003693 2023MS003693.
- [47] L. Scott, A. Borissova, A. Di Renzo, and M. Ghadiri, “Application of coarse-graining for large scale simulation of fluid and particle motion in spiral jet mill by cfd-dem,” *Powder Technology*, vol. 411, p. 117962, 2022.
- [48] W. G. Noid, “Perspective: Advances, challenges, and insight for predictive coarse-grained models,” *The Journal of Physical Chemistry B*, vol. 127, no. 19, pp. 4174–4207, 2023. PMID: 37149781.
- [49] M. J. del Razo, D. Crommelin, and P. G. Bolhuis, “Data-driven dynamical coarse-graining for condensed matter systems,” *The Journal of Chemical Physics*, vol. 160, Jan. 2024.
- [50] D. Aristoff, M. Johnson, and D. Perez, “Arbitrarily accurate, nonparametric coarse graining with markov renewal processes and the mori-zwanzig formulation,” 2023.

- [51] M. H. Duong, A. Lamacz, M. A. Peletier, A. Schlichting, and U. Sharma, “Quantification of coarse-graining error in langevin and overdamped langevin dynamics,” *Nonlinearity*, vol. 31, p. 4517â4566, Aug. 2018.
- [52] M. A. Katsoulakis, P. Plechac, L. Rey-Bellet, and D. K. Tsagkarogiannis, “Coarse-graining schemes and a posteriori error estimates for stochastic lattice systems,” 2006.
- [53] T. E. E. Hudson and X. H. Li, “Dynamical properties of coarse-grained linear sdes,” *Multiscale Modeling and Simulation : A SIAM Interdisciplinary Journal*, vol. 22, pp. 406–435, February 2024.
- [54] H. Mori, “Transport, collective motion, and brownian motion*),” *Progress of Theoretical Physics*, vol. 33, pp. 423–455, 03 1965.
- [55] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, vol. 23 of *Applications of Mathematics*. Berlin, Heidelberg: Springer-Verlag, 1st ed., 1992.
- [56] V. Panaretos and Y. Zemel, *An Invitation to Statistics in Wasserstein Space*. 01 2020.
- [57] *Entropy, Relative Entropy, and Mutual Information*, ch. 2, pp. 13–55. John Wiley Sons, Ltd, 2005.
- [58] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, “Physics-informed machine learning,” *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.
- [59] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, *et al.*, “Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence,” tech. rep., USDOE Office of Science (SC), Washington, DC (United States), 2019.
- [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 whatâs next,” *Journal of Scientific Computing*, vol. 92, no. 3, p. 88, 2022.
- [61] X. Jin, S. Cai, H. Li, and G. E. Karniadakis, “NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations,” *Journal of Computational Physics*, vol. 426, p. 109951, 2021.
- [62] M. Raissi, A. Yazdani, and G. E. Karniadakis, “Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations,” *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.

- [63] 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*, vol. 37, no. 12, pp. 1727–1738, 2021.
- [64] A. S. Joglekar and A. G. R. Thomas, “Machine learning of hidden variables in multiscale fluid simulation,” *Machine Learning: Science and Technology*, vol. 4, p. 035049, sep 2023.
- [65] J. J. J. Molina, K. Ogawa, and T. Taniguchi, “Stokesian processes: Inferring Stokes flows using physics-informed Gaussian processes,” *Machine Learning: Science and Technology*, 2023.
- [66] S. L. Brunton, “Applying machine learning to study fluid mechanics,” *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1718–1726, 2021.
- [67] P. Dubois, T. Gomez, L. Planckaert, and L. Perret, “Machine learning for fluid flow reconstruction from limited measurements,” *Journal of Computational Physics*, vol. 448, p. 110733, 2022.
- [68] D. Shu, Z. Li, and A. B. Farimani, “A physics-informed diffusion model for high-fidelity flow field reconstruction,” *Journal of Computational Physics*, vol. 478, p. 111972, 2023.
- [69] D. Liu and Y. Wang, “Multi-fidelity physics-constrained neural network and its application in materials modeling,” *Journal of Mechanical Design*, vol. 141, no. 12, 2019.
- [70] Y. Chen, L. Lu, G. E. Karniadakis, and L. Dal Negro, “Physics-informed neural networks for inverse problems in nano-optics and metamaterials,” *Optics express*, vol. 28, no. 8, pp. 11618–11633, 2020.
- [71] Z. Fang and J. Zhan, “Deep physical informed neural networks for metamaterial design,” *IEEE Access*, vol. 8, pp. 24506–24513, 2019.
- [72] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, “Physics-informed neural networks for high-speed flows,” *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112789, 2020.
- [73] G. S. Misyris, A. Venzke, and S. Chatzivasileiadis, “Physics-informed neural networks for power systems,” in *2020 IEEE Power & Energy Society General Meeting (PESGM)*, pp. 1–5, IEEE, 2020.
- [74] B. Huang and J. Wang, “Applications of physics-informed neural networks in power systems-a review,” *IEEE Transactions on Power Systems*, vol. 38, no. 1, pp. 572–588, 2022.
- [75] C. Moya and G. Lin, “DAE-PINN: a physics-informed neural network model for simulating differential algebraic equations with application to power networks,” *Neural Computing and Applications*, vol. 35, no. 5, pp. 3789–3804, 2023.

- [76] M. E. Bento, “Physics-guided neural network for load margin assessment of power systems,” *IEEE Transactions on Power Systems*, 2023.
- [77] Z. Wang, T. Hong, H. Li, and M. A. Piette, “Predicting city-scale daily electricity consumption using data-driven models,” *Advances in Applied Energy*, vol. 2, p. 100025, 2021.
- [78] L. McClenny and U. Braga-Neto, “Self-adaptive physics-informed neural networks using a soft attention mechanism,” *arXiv preprint arXiv:2009.04544*, 2020.
- [79] S. Wang, X. Yu, and P. Perdikaris, “When and why PINNs fail to train: A neural tangent kernel perspective,” *Journal of Computational Physics*, vol. 449, p. 110768, 2022.
- [80] S. Qadeer, A. Engel, A. Howard, A. Tsou, M. Vargas, P. Stinis, and T. Chiang, “Efficient kernel surrogates for neural network-based regression,” *arXiv preprint arXiv:2310.18612*, 2023.
- [81] J. Hou, Y. Li, and S. Ying, “Enhancing PINNs for solving PDEs via adaptive collocation point movement and adaptive loss weighting,” *Nonlinear Dynamics*, vol. 111, no. 16, pp. 15233–15261, 2023.
- [82] Z. Xiang, W. Peng, X. Liu, and W. Yao, “Self-adaptive loss balanced physics-informed neural networks,” *Neurocomputing*, vol. 496, pp. 11–34, 2022.
- [83] 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.
- [84] 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*, vol. 3, no. 3, pp. 218–229, 2021.
- [85] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, “Neural operator: Graph kernel network for partial differential equations,” *arXiv preprint arXiv:2003.03485*, 2020.
- [86] G. Wen, Z. Li, K. Azizzadenesheli, A. Anandkumar, and S. M. Benson, “U-FNO—an enhanced Fourier neural operator-based deep-learning model for multiphase flow,” *Advances in Water Resources*, vol. 163, p. 104180, 2022.
- [87] L. Lu, P. Jin, and G. E. Karniadakis, “DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators,” *arXiv preprint arXiv:1910.03193*, 2019.
- [88] A. A. Howard, M. Perego, G. E. Karniadakis, and P. Stinis, “Multifidelity deep operator networks for data-driven and physics-informed problems,” *Journal of Computational Physics*, vol. 493, p. 112462, 2023.

- [89] S. Wang, H. Wang, and P. Perdikaris, “Improved architectures and training algorithms for deep operator networks,” *Journal of Scientific Computing*, vol. 92, no. 2, p. 35, 2022.
- [90] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, “Kan: Kolmogorov-arnold networks,” *arXiv preprint arXiv:2404.19756*, 2024.
- [91] S. Rigas, M. Papachristou, T. Papadopoulos, F. Anagnostopoulos, and G. Alexandridis, “Adaptive training of grid-dependent physics-informed kolmogorov-arnold networks,” *IEEE Access*, 2024.
- [92] F. M. Rohrhofer, S. Posch, C. Gößnitzer, and B. Geiger, “On the role of fixed points of dynamical systems in training physics-informed neural networks,” *Transactions on Machine Learning Research*, vol. 2023, no. 1, p. 490, 2023.
- [93] S. Wang and P. Perdikaris, “Long-time integration of parametric evolution equations with physics-informed DeepONets,” *Journal of Computational Physics*, vol. 475, p. 111855, 2023.
- [94] 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*, vol. 4, no. 1, p. 015013, 2023.
- [95] 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*, 2019.
- [96] B. Moseley, A. Markham, and T. Nissen-Meyer, “Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations,” *Advances in Computational Mathematics*, vol. 49, no. 4, p. 62, 2023.
- [97] V. Dolean, A. Heinlein, S. Mishra, and B. Moseley, “Multilevel domain decomposition-based architectures for physics-informed neural networks,” *arXiv preprint arXiv:2306.05486*, 2023.
- [98] A. D. Jagtap, D. Mitsotakis, and G. E. Karniadakis, “Deep learning of inverse water waves problems using multi-fidelity data: Application to Serre–Green–Naghdi equations,” *Ocean Engineering*, vol. 248, p. 110775, 2022.
- [99] W. Chen and P. Stinis, “Feature-adjacent multi-fidelity physics-informed machine learning for partial differential equations,” *arXiv preprint arXiv:2303.11577*, 2023.
- [100] L. Lu, R. Pestourie, S. G. Johnson, and G. Romano, “Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport,” *Physical Review Research*, vol. 4, no. 2, p. 023210, 2022.

- [101] S. De, M. Reynolds, M. Hassanaly, R. N. King, and A. Doostan, “Bi-fidelity modeling of uncertain and partially unknown systems using DeepONets,” *Computational Mechanics*, vol. 71, no. 6, pp. 1251–1267, 2023.
- [102] P.-Y. Chuang and L. A. Barba, “Experience report of physics-informed neural networks in fluid simulations: pitfalls and frustration,” *arXiv preprint arXiv:2205.14249*, 2022.
- [103] P.-Y. Chuang and L. A. Barba, “Predictive limitations of physics-informed neural networks in vortex shedding,” *arXiv preprint arXiv:2306.00230*, 2023.
- [104] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, “Characterizing possible failure modes in physics-informed neural networks,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 26548–26560, 2021.
- [105] Y.-T. Liu, C.-Y. Wu, T. Chen, and Y. Yao, “Multi-fidelity surrogate modeling for chemical processes with physics-informed neural networks,” in *Computer Aided Chemical Engineering*, vol. 52, pp. 57–63, Elsevier, 2023.
- [106] M. Penwarden, S. Zhe, A. Narayan, and R. M. Kirby, “Multifidelity modeling for physics-informed neural networks (PINNs),” *Journal of Computational Physics*, vol. 451, p. 110844, 2022.
- [107] M. Aliakbari, M. Mahmoudi, P. Vadasz, and A. Arzani, “Predicting high-fidelity multiphysics data from low-fidelity fluid flow and transport solvers using physics-informed neural networks,” *International Journal of Heat and Fluid Flow*, vol. 96, p. 109002, 2022.
- [108] M. Ramezankhani, A. Nazemi, A. Narayan, H. Voggenreiter, M. Harandi, R. Seethaler, and A. S. Milani, “A data-driven multi-fidelity physics-informed learning framework for smart manufacturing: a composites processing case study,” in *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*, pp. 01–07, IEEE, 2022.
- [109] F. Regazzoni, S. Pagani, A. Cosenza, A. Lombardi, and A. Quarteroni, “A physics-informed multi-fidelity approach for the estimation of differential equations parameters in low-data or large-noise regimes,” *Rendiconti Lincei*, vol. 32, no. 3, pp. 437–470, 2021.
- [110] 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*, vol. 28, no. 5, 2020.
- [111] M. Penwarden, A. D. Jagtap, S. Zhe, G. E. Karniadakis, and R. M. Kirby, “A unified scalable framework for causal sweeping strategies for physics-informed neural networks (PINNs) and their temporal decompositions,” *arXiv preprint arXiv:2302.14227*, 2023.

- [112] R. Matthey and S. Ghosh, “A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations,” *Computer Methods in Applied Mechanics and Engineering*, vol. 390, p. 114474, 2022.
- [113] C. L. Wight and J. Zhao, “Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks,” *arXiv preprint arXiv:2007.04542*, 2020.
- [114] A. Heinlein, A. Klawonn, M. Lanser, and J. Weber, “Combining machine learning and domain decomposition methods for the solution of partial differential equations—a review,” *GAMM-Mitteilungen*, vol. 44, no. 1, pp. Paper No. e202100001, 28, 2021.
- [115] A. Klawonn, M. Lanser, and J. Weber, “Machine learning and domain decomposition methods - a survey,” *Computational Science and Engineering*, vol. 1, p. 2, Sept. 2024.
- [116] K. Shukla, A. D. Jagtap, and G. E. Karniadakis, “Parallel physics-informed neural networks via domain decomposition,” *Journal of Computational Physics*, vol. 447, p. 110683, 2021.
- [117] Z. Hu, A. D. Jagtap, G. E. Karniadakis, and K. Kawaguchi, “When do extended physics-informed neural networks (XPINNs) improve generalization?,” *SIAM Journal on Scientific Computing*, vol. 44, no. 5, pp. A3158–A3182, 2022.
- [118] V. Dolean, A. Heinlein, S. Mishra, and B. Moseley, “Finite Basis Physics-Informed Neural Networks as a Schwarz Domain Decomposition Method,” in *Domain Decomposition Methods in Science and Engineering XXVII* (Z. Dostál, T. Kozubek, A. Klawonn, U. Langer, L. F. Pavarino, J. A. A. Åstek, and O. B. Widlund, eds.), (Cham), pp. 165–172, Springer Nature Switzerland, 2024.
- [119] V. Dolean, A. Heinlein, S. Mishra, and B. Moseley, “Multilevel domain decomposition-based architectures for physics-informed neural networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 429, p. 117116, Sept. 2024.
- [120] A. Heinlein, A. A. Howard, D. Beecroft, and P. Stinis, “Multifidelity domain decomposition-based physics-informed neural networks for time-dependent problems,” *arXiv preprint arXiv:2401.07888*, 2024.
- [121] Y. Shang, A. Heinlein, S. Mishra, and F. Wang, “Overlapping schwarz preconditioners for randomized neural networks with domain decomposition,” *Computer Methods in Applied Mechanics and Engineering*, vol. 442, p. 118011, 2025.
- [122] S. Anderson, V. Dolean, B. Moseley, and J. Pestana, “ELM-FBPINN: efficient finite-basis physics-informed neural networks,” Sept. 2024. *arXiv:2409.01949* [cs, math].

- [123] B. Moseley, A. Markham, and T. Nissen-Meyer, “Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations,” *Advances in Computational Mathematics*, vol. 49, July 2023.
- [124] A. N. Kolmogorov, “On the representations of continuous functions of many variables by superposition of continuous functions of one variable and addition,” in *Dokl. Akad. Nauk USSR*, vol. 114, pp. 953–956, 1957.
- [125] A. A. Aghaei, “fkan: Fractional kolmogorov–arnold networks with trainable Jacobi basis functions,” *Neurocomputing*, vol. 623, p. 129414, 2025.
- [126] D. W. Abueidda, P. Pantidis, and M. E. Mobasher, “Deepokan: Deep operator network based on kolmogorov arnold networks for mechanics problems,” *Computer Methods in Applied Mechanics and Engineering*, vol. 436, p. 117699, 2025.
- [127] 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*, vol. 431, p. 117290, 2024.
- [128] Y. Wang, J. Sun, J. Bai, C. Anitescu, M. S. Eshaghi, X. Zhuang, T. Rabczuk, and Y. Liu, “Kolmogorov–arnold-informed neural network: A physics-informed deep learning framework for solving forward and inverse problems based on kolmogorov–arnold networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 433, p. 117518, 2025.
- [129] R. Genet and H. Inzirillo, “Tkan: Temporal kolmogorov-arnold networks,” *arXiv preprint arXiv:2405.07344*, 2024.
- [130] M. Kiamari, M. Kiamari, and B. Krishnamachari, “Gkan: Graph kolmogorov-arnold networks,” *arXiv preprint arXiv:2406.06470*, 2024.
- [131] G. D. Carlo, A. Mastropietro, and A. Anagnostopoulos, “Kolmogorov-arnold graph neural networks,” 2024.
- [132] R. Bresson, G. Nikolentzos, G. Panagopoulos, M. Chatzianastasis, J. Pang, and M. Vazirgiannis, “Kagnns: Kolmogorov-arnold networks meet graph learning,” 2024.
- [133] S. SS, “Chebyshev polynomial-based kolmogorov-arnold networks: An efficient architecture for nonlinear function approximation,” *arXiv preprint arXiv:2405.07200*, 2024.
- [134] A. D. Bodner, A. S. Tepsich, J. N. Spolski, and S. Pourteau, “Convolutional kolmogorov-arnold networks,” *arXiv preprint arXiv:2406.13155*, 2024.

- [135] Q. Qiu, T. Zhu, H. Gong, L. Chen, and H. Ning, “Relu-kan: New kolmogorov-arnold networks that only need matrix addition, dot multiplication, and relu,” *arXiv preprint arXiv:2406.02075*, 2024.
- [136] M. Cheon, “Kolmogorov-arnold network for satellite image classification in remote sensing,” *arXiv preprint arXiv:2406.00600*, 2024.
- [137] Z. Huang, J. Cui, L. Yu, L. F. Herbozo Contreras, and O. Kavehei, “Abnormality detection in time-series bio-signals using kolmogorov-arnold networks for resource-constrained devices,” *medRxiv*, pp. 2024–06, 2024.
- [138] B. Azam and N. Akhtar, “Suitability of kans for computer vision: A preliminary investigation,” *arXiv preprint arXiv:2406.09087*, 2024.
- [139] 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.
- [140] S. Rigas and M. Papachristou, “jaxKAN: A JAX-based implementation of Kolmogorov-Arnold Networks,” May 2024.
- [141] V. Dolean, A. Heinlein, S. Mishra, and B. Moseley, “Multilevel domain decomposition-based architectures for physics-informed neural networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 429, p. 117116, 2024.
- [142] K. Lee, N. A. Trask, R. G. Patel, M. A. Gulian, and E. C. Cyr, “Partition of unity networks: deep hp-approximation,” 2021. <https://arxiv.org/abs/2101.11256>.
- [143] E. Ka, L. Leclercq, and S. V. Ukkusuri, “Adaptive domain decomposition physics-informed neural networks for traffic state estimation with sparse sensor data,” 2026. <https://arxiv.org/abs/2605.08028>.
- [144] Y. Xiong, H. Wei, Z. Ma, Z. Ding, and Y. Peng, “Dadd-pinn: Dual adaptive domain decomposition physics-informed neural networks,” *Mathematics*, vol. 14, no. 4, 2026.
- [145] J. Botvinick-Greenhouse, W. H. Ali, M. Benosman, and S. Mowlavi, “AB-PINNs: Adaptive-basis physics-informed neural networks for residual-driven domain decomposition,” *Machine Learning: Science and Technology*, 2026.
- [146] A. Bonfanti, I. Medina, R. List, B. Staeves, R. Santana, and M. Ellero, “PINN balls: Scaling second-order methods for PINNs with domain decomposition and adaptive sampling,” in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.
- [147] K. Tang, J. Zhai, X. Wan, and C. Yang, “Adversarial adaptive sampling: Unify PINN and optimal transport for the approximation of PDEs,” 2023. <https://api.semanticscholar.org/CorpusID:258967218>.

- [148] N. Trask, A. Henriksen, C. Martinez, and E. Cyr, “Hierarchical partition of unity networks: fast multilevel training,” in *Mathematical and Scientific Machine Learning*, pp. 271–286, PMLR, 2022.
- [149] A. Rodriguez, A. Chattopadhyay, P. Kumar, L. F. Rodriguez, and V. Kumar, “Partition of unity physics-informed neural networks (pou-pinns): An unsupervised framework for physics-informed domain decomposition and mixtures of experts,” *arXiv preprint arXiv:2412.06842*, 2024.
- [150] S. Wang, X. Yu, and P. Perdikaris, “When and why PINNs fail to train: A neural tangent kernel perspective,” *Journal of Computational Physics*, vol. 449, p. 110768, 2022.
- [151] L. D. McClenny and U. M. Braga-Neto, “Self-adaptive physics-informed neural networks,” *Journal of Computational Physics*, vol. 474, p. 111722, Feb. 2023.
- [152] W. Chen, A. Howard, and P. Stinis, “Self-adaptive weighting and sampling for physics-informed neural networks,” *Machine Learning: Science and Technology*, vol. 7, no. 2, p. 025050, 2026.
- [153] I. Lagaris, A. Likas, and D. Fotiadis, “Artificial neural networks for solving ordinary and partial differential equations,” *IEEE Transactions on Neural Networks*, vol. 9, no. 5, p. 987â1000, 1998.
- [154] S. Dong and N. Ni, “A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks,” *Journal of Computational Physics*, vol. 435, p. 110242, 2021.
- [155] S. Wang, S. Sankaran, and P. Perdikaris, “Respecting causality for training physics-informed neural networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 421, p. 116813, 2024.
- [156] N. Trask, A. Henriksen, C. Martinez, and E. Cyr, “Hierarchical partition of unity networks: fast multilevel training,” in *Proceedings of Mathematical and Scientific Machine Learning* (B. Dong, Q. Li, L. Wang, and Z.-Q. J. Xu, eds.), vol. 190 of *Proceedings of Machine Learning Research*, pp. 271–286, PMLR, 15–17 Aug 2022.
- [157] R. S. Govindaraju and B. Zhang, *Radial-Basis Function Networks*, pp. 93–109. Dordrecht: Springer Netherlands, 2000.
- [158] S. A. Billings and G. L. Zheng, “Radial basis function network configuration using genetic algorithms,” *Neural Networks*, vol. 8, no. 6, pp. 877–890, 1995.
- [159] E. C. Cyr, M. A. Gulian, R. G. Patel, M. Perego, and N. A. Trask, “Robust training and initialization of deep neural networks: An adaptive basis viewpoint,” 2019. <https://arxiv.org/abs/1912.04862>.

- [160] C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu, “A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 403, p. 115671, Jan. 2023.
- [161] J. M. Hammersley, “Monte Carlo methods for solving multivariable problems,” in *Annals of the New York Academy of Sciences*, vol. 86, pp. 844–874, 1960.
- [162] G. E. K. Ameya D Jagtap, “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*, vol. 28, no. 5, pp. 2002–2041, 2020.
- [163] V. Kumar, S. Goswami, D. J. Smith, and G. E. Karniadakis, “Real-time prediction of multiple output states in diesel engines using a deep neural operator framework,” *arXiv preprint arXiv:2304.00567*, 2023.
- [164] S. Goswami, K. Kontolati, M. D. Shields, and G. E. Karniadakis, “Deep transfer operator learning for partial differential equations under conditional shift,” *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1155–1164, 2022.
- [165] M. Yin, E. Zhang, Y. Yu, and G. E. Karniadakis, “Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems,” *Computer methods in applied mechanics and engineering*, vol. 402, p. 115027, 2022.
- [166] Q. He, M. Perego, A. A. Howard, G. E. Karniadakis, and P. Stinis, “A hybrid deep neural operator/finite element method for ice-sheet modeling,” *Journal of Computational Physics*, vol. 492, p. 0112428, 2023.
- [167] S. Koric and D. W. Abueidda, “Data-driven and physics-informed deep learning operators for solution of heat conduction equation with parametric heat source,” *International Journal of Heat and Mass Transfer*, vol. 203, p. 123809, 2023.
- [168] L. Liu, K. Nath, and W. Cai, “A causality-DeepONet for causal responses of linear dynamical systems,” *arXiv preprint arXiv:2209.08397*, 2022.
- [169] S. Wang, H. Wang, and P. Perdikaris, “Learning the solution operator of parametric partial differential equations with physics-informed DeepONets,” *Science advances*, vol. 7, no. 40, p. eabi8605, 2021.
- [170] S. Goswami, A. Bora, Y. Yu, and G. E. Karniadakis, “Physics-informed deep neural operators networks,” *arXiv preprint arXiv:2207.05748*, 2022.
- [171] 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*, vol. 391, p. 114587, 2022.

- [172] Y. Hao, P. C. Di Leoni, O. Marxen, C. Meneveau, G. E. Karniadakis, and T. A. Zaki, “Instability-wave prediction in hypersonic boundary layers with physics-informed neural operators,” *Journal of Computational Science*, p. 102120, 2023.
- [173] S. Wang, S. Sankaran, and P. Perdikaris, “Respecting causality is all you need for training physics-informed neural networks,” *arXiv preprint arXiv:2203.07404*, 2022.
- [174] C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu, “A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 403, p. 115671, 2023.
- [175] M. Ainsworth and J. Dong, “Galerkin neural networks: A framework for approximating variational equations with error control,” *SIAM Journal on Scientific Computing*, vol. 43, no. 4, pp. A2474–A2501, 2021.
- [176] M. Ainsworth and J. Dong, “Galerkin neural network approximation of singularly-perturbed elliptic systems,” *Computer Methods in Applied Mechanics and Engineering*, vol. 402, p. 115169, 2022.
- [177] A. A. Howard, S. H. Murphy, S. E. Ahmed, and P. Stinis, “Stacked networks improve physics-informed training: applications to neural networks and deep operator networks,” *arXiv preprint arXiv:2311.06483*, 2023.
- [178] Z. Aldirany, R. Cottureau, M. Laforest, and S. Prudhomme, “Multi-level neural networks for accurate solutions of boundary-value problems,” *arXiv preprint arXiv:2308.11503*, 2023.
- [179] Y. Wang and C.-Y. Lai, “Multi-stage neural networks: Function approximator of machine precision,” *arXiv preprint arXiv:2307.08934*, 2023.
- [180] S. J. Anagnostopoulos, J. D. Toscano, N. Stergiopoulos, and G. E. Karniadakis, “Residual-based attention in physics-informed neural networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 421, p. 116805, 2024.
- [181] J. D. Toscano, V. Oommen, A. J. Varghese, Z. Zou, N. Ahmadi Daryakenari, C. Wu, and G. E. Karniadakis, “From pinns to pikans: Recent advances in physics-informed machine learning,” *Machine Learning for Computational Science and Engineering*, vol. 1, no. 1, p. 15, 2025.
- [182] S. Wang, Y. Teng, and P. Perdikaris, “Understanding and mitigating gradient flow pathologies in physics-informed neural networks,” *SIAM Journal on Scientific Computing*, vol. 43, no. 5, pp. A3055–A3081, 2021.
- [183] D. Liu and Y. Wang, “A dual-dimer method for training physics-constrained neural networks with minimax architecture,” *Neural Networks*, vol. 136, p. 112â125, Apr. 2021.

- [184] Y. Song, H. Wang, H. Yang, M. L. Taccari, and X. Chen, “Loss-attentional physics-informed neural networks,” *Journal of Computational Physics*, vol. 501, p. 112781, 2024.
- [185] G. Zhang, H. Yang, F. Zhu, Y. Chen, and x. zheng, “Dasa-pinns: Differentiable adversarial self-adaptive pointwise weighting scheme for physics-informed neural networks,” *SSRN Electronic Journal*, 01 2023.
- [186] A. Jacot, F. Gabriel, and C. Hongler, “Neural tangent kernel: Convergence and generalization in neural networks,” *Advances in neural information processing systems*, vol. 31, 2018.
- [187] E. Williams, A. Howard, B. Meuris, and P. Stinis, “What do physics-informed deepoanets learn? understanding and improving training for scientific computing applications,” 2024.
- [188] S. A. Faroughi and F. Mostajeran, “Neural tangent kernel analysis to probe convergence in physics-informed neural solvers: Pikans vs. pinns,” 2025.
- [189] P. M. Long, “Properties of the after kernel,” *ArXiv*, vol. abs/2105.10585, 2021.
- [190] Z. Fan and Z. Wang, “Spectra of the conjugate kernel and neural tangent kernel for linear-width neural networks,” *Advances in neural information processing systems*, vol. 33, pp. 7710–7721, 2020.
- [191] Z. Wang, A. Engel, A. Sarwate, I. Dumitriu, and T. Chiang, “Spectral evolution and invariance in linear-width neural networks,” *arXiv preprint arXiv:2211.06506*, 2022.
- [192] J. Bai, G.-R. Liu, A. Gupta, L. Alzubaidi, X.-Q. Feng, and Y. Gu, “Physics-informed radial basis network (pirbn): A local approximating neural network for solving nonlinear partial differential equations,” *Computer Methods in Applied Mechanics and Engineering*, vol. 415, p. 116290, 2023.