

University of Strathclyde Glasgow

Efficient finite basis physics informed neural networks

University of Strathclyde,

Glasgow, UK

A thesis submitted for the degree of

Doctor of Philosophy

Sam Anderson

August 29, 2026

Declaration of Authenticity

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

The copyright of this thesis belongs to the author under the terms of the United Kingdom Copyright Acts as qualified by University of Strathclyde Regulation 3.50. Due acknowledgement must always be made of the use of any material contained in, or derived from, this thesis.

Signed: *Sam Anderson*
Date: August 29, 2026

Abstract

Physics-informed neural networks (PINNs) offer several advantages when compared to traditional numerical methods for solving PDEs, such as being a mesh-free approach and being easily extendable to solving inverse problems. One promising approach for allowing PINNs to scale to multi-scale problems is to combine them with domain decomposition; for example, finite basis physics-informed neural networks (FBPINNs) replace the global PINN network with many localised networks which are summed together to approximate the solution. In this work, we significantly accelerate the training of FBPINNs by linearising their underlying optimisation problem. We achieve this by employing extreme learning machines (ELMs) as their subdomain networks and showing that this turns the FBPINN optimisation problem into one of solving a linear system or least-squares problem.

Acknowledgements

I would like to first express my gratitude to my supervisors Victorita Dolean and Jennifer Pestana, without whom this thesis would not exist. Your constant support, guidance, and patience have made my journey a very pleasant one. The time you spared me in spite of both of your never-ending workloads is greatly appreciated.

Thank you also to Ben Moseley for providing me technical support and collaboration. I have improved immensely as a programmer under your guidance and the advice I received from you will be carried in my future projects. Thank you also to Philéas, I am grateful for your impactful contribution to this work.

I would not have completed this thesis without my peers at the University of Strathclyde, in particular Ann, Hywel, and Katie. More importantly than your vital aid at the beginning of my journey, I will always value your friendship throughout my PhD and beyond. Thank you all for your support.

Thank you also to the Department of Mathematics and Statistics and Faculty of Science of the University of Strathclyde, NMIS and AFRC for jointly funding this work.

Thank you to my parents, Lai and Hamish, for the eternal support and belief in me. Thank you to my sister, Jasmin, for always being there when I need you, and for being my oldest friend. I dedicate this thesis to you all, for everything you have done, and will do, for me.

Thanks to my friends Iain and Niall, for their constant support. Without my dogs, Luna and Maya, I would not have had the mental strength to complete this journey.

Finally, I want to acknowledge my partner, Vale, for pushing me forward, inspiring me, and always being there when I need you. Thank you, I love you.

Contents

List of Figures **iv**

List of Tables **xii**

1 Introduction **1**

2 Research Background **7**

2.1 Physics-Informed Neural Networks - PINNs 7

2.2 Applications 10

2.3 PINN limitations 13

2.4 PINN Extensions 15

2.5 FBPINNs 21

2.6 Extreme Learning Machines (ELMs) 23

2.7 Summary 26

3 ELM-FBPINN **27**

3.1 Introduction 27

3.2 The ELM-FBPINN method 28

3.3 Problems studied 30

3.3.1 1D problem – Damped harmonic oscillator 31

3.3.2 Multiscale Laplace in 2D 33

3.3.3 Common implementation details 35

3.4 Numerical Results 38

3.4.1 1D Damped harmonic oscillator 38

3.4.2 2D - Multiscale Laplace 60
3.5 Discussion and conclusions 74

4 Basis pre-training **79**
4.1 Introduction 79
4.2 Pre-Training Definition 80
4.3 Problem Definition 83
4.3.1 1D Damped Harmonic Oscillator 84
4.3.2 2D Multi-scale Laplace 86
4.3.3 Common Implementation 88
4.4 Numerical Results 90
4.4.1 1D Damped Harmonic Oscillator 91
4.4.2 2D Multi-scale Laplace 132
4.5 Discussion and Conclusions 156

5 Conclusion **160**

Bibliography **163**

List of Figures

2.1 Scopus search results by year and subject for 'physics informed neural network' and 'PINN' between 2019 and 2025. 12
2.2 Taxonomy of PINN extensions based on their primary methodological modification. 19

3.1 Generalised workflow of PINN like methods with domain decomposition. PINN, FBPINN, and ELM-FBPINN all follow this basic structure, with Step 2. only being relevant for FBPINN and ELM-FBPINN. 30
3.2 Left: Example domain decomposition with 6 subdomains and width factor, $\eta = 2.9$. Coloured bars indicate the support of each window function. Right: Exact solution of the damped harmonic oscillator for $\omega_0 = 80$, $u_0 = 1$, and $v_0 = 0$. 33
3.3 Left: Example two-dimensional domain decomposition with 10×10 subdomains and width factor of $\eta = 2.9$. Three window functions are highlighted in the lower-left region, while the orange colouring indicates the summed partition of unity. Right: Example exact solution of the multi-scale Laplace problem with $n_\omega = 4$ and $\omega_4 = 16$. 35
3.4 Example one-dimensional decompositions for varying width factors. Top Left: $\eta = 1$. Top Right: $\eta = 1.5$. Bottom: $\eta = 2.9$ 37
3.5 Logarithm of the $\hat{e}_{L^1}$ test error versus iteration count for the baseline 1D damped harmonic oscillator. 39

3.6 Model predictions versus the exact solution for the 1D damped harmonic oscillator. Top left: PINN; top right: FBPINN; bottom: ELM-FBPINN. 40
3.7 Logarithm of the $\hat{e}_{L^1}$ error vs iterations for the 1D damped harmonic oscillator for different numbers of basis functions, C. 42
3.8 Final $\hat{e}_{L^1}$ values for each model vs the number of basis functions, C. . . 43
3.9 Top: Condition number, $\kappa(\mathbf{M})$ and Bottom: rank($\mathbf{M}$) vs number of basis functions, C. 44
3.10 Basis functions of ELM-FBPINN, within a single subdomain. Top: $C = 8$ Bottom: $C = 32$. 45
3.11 Convergence curves for varying number of subdomains, J, for FBPINN and ELM-FBPINN. 46
3.12 Logarithm of final $\hat{e}_{L^1}$ values for varying number of subdomains, J, for FBPINN and ELM-FBPINN. 47
3.13 Exact solution and model predictions when $J = 8$. 48
3.14 Condition number, $\kappa(\mathbf{M})$ and rank($\mathbf{M}$) vs number of subdomains, J. . . 49
3.15 Convergence curves for varying number of subdomains, J, and solution frequency, ω_0 for FBPINN and ELM-FBPINN. 50
3.16 Convergence curves for varying number of subdomains, J, and solution frequency, ω_0 for FBPINN and ELM-FBPINN. 51
3.17 Convergence curves for increasing network layer depth, h and decreasing number of subdomains, J, for FBPINN and ELM-FBPINN. 52
3.18 Convergence curves for increasing network layer depth, h and decreasing number of subdomains, J, for ELM-FBPINN. 53
3.19 Hidden layer basis functions inside the 4th subdomain. Top: $h = 2$; Bottom: $h = 4$. Left: FBPINN: Right: ELM-FBPINN. 55
3.20 Condition number, $\kappa(\mathbf{M})$ and rank($\mathbf{M}$) vs depth of network layers, h. Note that while h is increased, number of subdomains, J, is decreased. . 56
3.21 Convergence curves for increasing network layer depth, h and decreasing number of basis functions, C, for all models. 57

3.22 Convergence curves for increasing network layer depth, h and decreasing number of basis functions, C, for ELM-FBPINN. 58
3.23 Condition number, $\kappa(\mathbf{M})$ and rank($\mathbf{M}$) vs depth of network layers, h. Note that while h is increased, number of basis functions, C, is decreased. 59
3.24 Convergence curves for baseline models for the 2D multiscale Laplace problem. 61
3.25 Baseline model predictions for the 2D multiscale Laplace problem. The rows correspond to the solutions obtained with PINN (top), FBPINN (middle) and ELM-FBPINN (bottom). For each method the predicted solution (left), exact solution (middle) and difference between the predicted and exact solutions (right) are shown. 63
3.26 Convergence curves for all models for varying number of basis functions C. 64
3.27 Final L^1 test loss ($\hat{e}_{L^1}$) vs number of basis functions (C) for all models. 65
3.28 Convergence curves for FBPINN and ELM-FBPINN for varying number of subdomains per axis (J). 67
3.29 Final L^1 test loss ($\hat{e}_{L^1}$) vs number of subdomains per axis (J). 68
3.30 Final L^1 test loss ($\hat{e}_{L^1}$) vs number of subdomains per axis (J) as the number of frequency components n_ω, is varied. 69
3.31 Final L^1 test loss ($\hat{e}_{L^1}$) vs number of subdomains per axis (J) as the number of frequency components n_ω, is varied. 70
3.32 Convergence curves for FBPINN and ELM-FBPINN for varying network depth (h) and number of subdomains per axis (J). 72
3.33 Final L^1 test loss ($\hat{e}_{L^1}$) for FBPINN and ELM-FBPINN for varying network depth (h) and number of subdomains per axis (J). 73
3.34 Convergence curves for FBPINN and ELM-FBPINN for varying network depth (h) and number of basis functions (C). 74
3.35 Final $\hat{e}_{L^1}$ for FBPINN and ELM-FBPINN for varying network depth (h) and number of basis functions (J). 75

List of Figures

4.1 Histograms showing the distribution of 250 randomly sampled initial conditions for the 1D damped harmonic oscillator. Left: Bimodal normal distribution for displacement, u_0. Right: Normal distribution for velocity, v_0. 86
4.2 Histogram of the uniform distribution used to generate the scaling coefficients, c, for the 2D multi-scale Laplace problem. Here the number of solutions has been set to 100. 87
4.3 Mean error, $\hat{e}_{L^1}$, vs Iterations for PT for the 1D harmonic oscillator and the baseline configuration. 91
4.4 PT predictions versus the exact solution for the 1D damped harmonic oscillator. Top left: $u_0 = 0.82$, $v_0 = -2.01$; Top right: $u_0 = -0.89$, $v_0 = -1.14$; Bottom: $u_0 = -1.18$, $v_0 = 2.62$. 92
4.5 Convergence curves for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for the 1D harmonic oscillator with the baseline configuration. 93
4.6 Hidden-layer basis functions for baseline configurations. Left: ELM-FBPINN; Right: PT-ELM-FBPINN. 95
4.7 Convergence curves for PT-ELM-FBPINN for varying PT iterations. A baseline ELM-FBPINN convergence curve is also plotted. 96
4.8 Final $\hat{e}_{L^1}$ values for PT-ELM-FBPINN as the number of PT iterations are varied. A baseline ELM-FBPINN result has been included for comparison. 97
4.9 Condition number, $\kappa(\mathbf{M})$ and rank($\mathbf{M}$) vs number of PT iterations. . . 99
4.10 Convergence curves for PT with varying number of pre-training solutions (P). 100
4.11 Convergence curves for PT-ELM-FBPINN with varying number of pre-training solutions (P). 101
4.12 Final $\hat{e}_{L^1}$ values for PT and PT-ELM-FBPINN with varying number of pre-training solutions (P). 101
4.13 Condition number, $\kappa(\mathbf{M})$ and rank($\mathbf{M}$) vs number of PT training solutions, P. 102

4.14 Convergence for each PT model with varying number of subdomains (J). 103
4.15 Convergence for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of subdomains (J). 104
4.16 Final error, $\hat{e}_{L^1}$ vs number of subdomains (J) for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN. 105
4.17 Basis functions for ELM-FBPINN and PT-ELM-FBPINN within the subdomain $\frac{J}{2}$. Top row: $J = 12$; Bottom row: $J = 128$. Left column: ELM-FBPINN; Right column: PT-ELM-FBPINN. 106
4.18 2-norm condition number and rank of the matrix $\mathbf{M}$ for both ELM-FBPINN and PT-ELM-FBPINN for varing number of subdomains, J. . 109
4.19 Convergence for PT with varying number of basis functions (C). 111
4.20 Convergence for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN with varying number of basis functions (C). 111
4.21 Final $\hat{e}_{L^1}$ values for PT and PT-ELM-FBPINN with varying number of basis functions, C. 112
4.22 SVD-based condition number and rank of the matrix $\mathbf{M}$ for ELM-FBPINN and PT-ELM-FBPINN for varying number of basis functions, C. 113
4.23 Convergence of PT for varying number of hidden layers, h, and number of subdomains, J. 115
4.24 Convergence of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of hidden layers, h, and number of subdomains, J. . . . 116
4.25 Convergence of ELM-FBPINN, and PT-ELM-FBPINN for varying number of hidden layers, h, and number of subdomains, J. 116
4.26 Predicted solutions for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN with number of hidden layers $h = 6$, and number of subdomains $J = 6$. . 117
4.27 Final $\hat{e}_{L^1}$ of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of hidden layers, h, and number of subdomains, J. 119
4.28 Hidden-layer basis functions for PT-ELM-FBPINN with $h = 4$ 119

4.29 SVD-based condition number and rank of the matrix $\mathbf{M}$ for ELM-FBPINN and PT-ELM-FBPINN for varying number of hidden layers, h and number of subdomains, J. 120
4.30 Convergence of PT models with varying number of hidden layers, h, and number of basis functions, C. 122
4.31 Convergence for FBPINN, ELM-FBPINN and PT-ELM-FBPINN with varying number of hidden layers, h and number of basis functions, C. . 122
4.32 Convergence for ELM-FBPINN and PT-ELM-FBPINN with varying number of hidden layers, h and number of basis functions, C. 124
4.33 Final $\hat{e}_{L^1}$ for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of hidden layers, h, and number of basis functions, C. 125
4.34 SVD-based condition number and rank of the matrix $\mathbf{M}$ for ELM-FBPINN and PT-ELM-FBPINN. 126
4.35 Convergence of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying solution frequency, ω_0, and number of subdomains, J. 129
4.36 Final $\hat{e}_{L^1}$ of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying solution frequency, ω_0, and number of subdomains, J. 130
4.37 Condition number and rank of matrix $\mathbf{M}$ for ELM-FBPINN and PT-ELM-FBPINN when varying solution frequency, ω_0 and number of subdomains, J. 131
4.38 Convergence of PT baseline model for the 2D multi-scale Laplace problem. 134
4.39 PT predictions versus the exact solution for the 2D multi-scale Laplace problem. Left: PT predicted solutions; Middle: exact solutions; Right: Difference. 135
4.40 Convergence of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN baseline models for the 2D multi-scale Laplace problem. 136
4.41 Convergence of each PT-ELM-FBPINN model with varying number of PT iterations for the 2D multi-scale Laplace problem. An ELM-FBPINN baseline result has also been plotted for comparison. 138

4.42 Final $\hat{e}_{L^1}$ for PT-ELM-FBPINN as PT iterations are varied for the 2D multi-scale Laplace problem. An ELM-FBPINN baseline has been included for comparison. 139
4.43 Convergence of PT models for varying number of subdomains per axis, J, for the 2D multi-scale Laplace problem. 140
4.44 Convergence of FBPINN, ELM-FBPINN and PT-ELM-FBPINN for varying number of subdomains, J, for the 2D multi-scale Laplace problem. . 141
4.45 Final $\hat{e}_{L^1}$ for varying number of subdomains per axis, J, for the 2D multi-scale Laplace problem. 142
4.46 Convergence for PT models for varying number of basis functions, C, for the 2D multi-scale Laplace problem. 143
4.47 Convergence for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of basis functions, C, for the 2D multi-scale Laplace problem. 144
4.48 Final $\hat{e}_{L^1}$ for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of basis functions, C, for the 2D multi-scale Laplace problem. 145
4.49 Convergence for PT models with varying number of hidden layers, h, and number of subdomains, J, for the 2D multi-scale Laplace problem. . 146
4.50 Convergence for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN with varying number of hidden layers, h, and number of subdomains per axis, J, for the 2D multi-scale Laplace problem. 147
4.51 Predicted solutions for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN with $h = 3$ hidden layers and $J = 8$ subdomains per axis for the 2D multi-scale Laplace problem. Top row: FBPINN; Middle row: ELM-FBPINN; Bottom row: PT-ELM-FBPINN. Left column: Predicted solution; Middle column: Exact solution; Right column: Difference. 148
4.52 Final $\hat{e}_{L^1}$ for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN with varying number of hidden layers, h, and number of subdomains per axis, J, for the 2D multi-scale Laplace problem. 149

4.53 Convergence of PT models for varying number of hidden layers, h, and number of basis functions, C, for the 2D multi-scale Laplace problem. . 150
4.54 Convergence of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of hidden layers, h, and number of basis functions, C, for the 2D multi-scale Laplace problem. 151
4.55 Final $\hat{e}_{L^1}$ of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of hidden layers, h, and number of basis functions, C, for the 2D multi-scale Laplace problem. 152
4.56 Convergence for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying n_ω and number of subdomains per axis, J, for the 2D multi-scale Laplace problem. 154
4.57 Final $\hat{e}_{L^1}$ for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying n_ω and number of subdomains per axis, J, for the 2D multi-scale Laplace problem. 154

List of Tables

2.1 Summary of major PINN extensions categorised by their primary methodological focus. 20

3.1 Baseline results for the 1D damped harmonic oscillator. Reported $\hat{e}_{L^1}$, $\hat{e}_{L^2}$ and $\hat{e}_{L^\infty}$ values correspond to mean ± range over random seeds, with scientific notation factorized for readability. Time values are mean ± 1 standard deviation. 41

3.2 Condition number $\kappa(\mathbf{M})$ and numerical rank of the ELM-FBPINN system matrix for the baseline configuration. 41

3.3 Final $\hat{e}_{L^1}$ values for each model at different number of basis functions, C. 43

3.4 SVD-based condition number and matrix rank for ELM-FBPINN as a function of C. 44

3.5 Final $\hat{e}_{L^1}$ values for each model at different number of basis functions, J. 47

3.6 SVD-based condition number and matrix rank for ELM-FBPINN as a function of J. 49

3.7 Final $\hat{e}_{L^1}$ values for each model at different solution frequency, ω_0. . . . 51

3.8 Final ELM-FBPINN $\hat{e}_{L^1}$ values and number of iterations for varying network depth, h and number of subdomains, J. 54

3.9 Final $\hat{e}_{L^1}$ values for varying network depth, h and number of subdomains, J. 54

3.10 SVD-based condition number and matrix rank as h and J are varied for ELM-FBPINN. 56

3.11 Final $\hat{e}_{L^1}$ values for varying network depth, h and number of basis functions, C. 57
3.12 Final ELM-FBPINN $\hat{e}_{L^1}$ values and number of iterations for varying network depth, h and number of basis functions, C. 58
3.13 SVD-based condition number and matrix rank as h and C are varied for ELM-FBPINN. 59
3.14 Baseline results for the 2D multiscale Laplace. 62
3.15 Final $\hat{e}_{L^1}$ values and average run time for varying number of basis functions, C. 66
3.16 Final $\hat{e}_{L^1}$ values for varying number of subdomains, J. 68
3.17 Final e_{L^1} values and average run time for varying n_ω. 71
3.18 Final $\hat{e}_{L^1}$ values and average run time for varying number of hidden layers (h) and number of subdomains per axis (J). 72
3.19 Final L^1 test loss ($\hat{e}_{L^1}$) values and number of iterations for varying network depth, h and number of subdomains, J. 73
3.20 Final e_{L^1} values for varying number of hidden layers (h) and basis functions (C). 75
3.21 Final $\hat{e}_{L^1}$ values and number of iterations for varying network depth, h and number of basis functions, C. 75

4.1 Baseline results for the 1D damped harmonic oscillator. Reported $\hat{e}_{L^1}$, $\hat{e}_{L^2}$ and $\hat{e}_{L^\infty}$ values are mean $\pm$ range over random seeds, while time values are mean $\pm$ 1 standard deviation. 94
4.2 Condition number $\kappa(\mathbf{M})$ and numerical rank of the ELM-FBPINN and PT-ELM-FBPINN system matrices for the baseline configuration. . . . 95
4.3 Final $\hat{e}_{L^1}$ values for varying PT iterations for PT-ELM-FBPINN. A baseline ELM-FBPINN result has also been included for comparison. 98
4.4 SVD-based condition number and matrix rank for PT-ELM-FBPINN as a function of PT iterations. A baseline ELM-FBPINN result has also been included for comparison. 99

List of Tables

4.5 Final $\hat{e}_{L^1}$ values and average run time for PT and PT-ELM-FBPINN for varying number of PT training solutions, P. 101
4.6 SVD-based condition number and matrix rank for PT-ELM-FBPINN as a function of the number of training solutions. 102
4.7 Final $\hat{e}_{L^1}$ values for each model at different numbers of subdomains, J. PT results are included for discussion purposes and are not a direct comparison to other models' performance. 104
4.8 SVD-based condition number and matrix rank for ELM-FBPINN and pre-trained ELM-FBPINN as a function of the number of subdomains. . 108
4.9 Final $\hat{e}_{L^1}$ values for each model at different number of basis functions, C; PT is separated for clarity. 112
4.10 Condition numbers and ranks of ELM-FBPINN and PT-ELM-FBPINN matrices for varying number of hidden units, C. 114
4.11 Final $\hat{e}_{L^1}$ values and number of iterations for varying network depth, h, and number of subdomains, J. 118
4.12 Final $\hat{e}_{L^1}$ values and average run time for varying network depth, h and number of subdomains, J. 118
4.13 Condition numbers and ranks of ELM-FBPINN and PT-ELM-FBPINN matrices for varying number of hidden layers, h, and number of subdomains, J. 121
4.14 Final $\hat{e}_{L^1}$ values and number of iterations for varying network depth, h and number of basis functions, C. 124
4.15 Final $\hat{e}_{L^1}$ values and average run time for varying network depth, h, and number of basis functions, C. 125
4.16 Condition numbers and ranks of ELM-FBPINN and Pre-trained ELM-FBPINN matrices for varying number of hidden units, h and number of basis functions C. 127
4.17 Final $\hat{e}_{L^1}$ values for each model at different solution frequencies, ω_0 and number of subdomains, J. 130

4.18 Condition numbers and ranks of ELM-FBPINN and Pre-trained ELM-FBPINN matrices for varying solution frequency, ω_0 and number of subdomains J. 131
4.19 Baseline results for the 2D multi-scale Laplace problem for PT, FBPINN, ELM-FBPINN and PT-ELM-FBPINN. PT has been included for reference and is not a direct comparison of performance against the other models. 136
4.20 PT-ELM-FBPINN final $\hat{e}_{L^1}$ values for varying numbers of pre-trainer iterations for the 2D multi-scale Laplace problem. 139
4.21 Final $\hat{e}_{L^1}$ values for varying number of subdomains per axis, J, for the 2D multi-scale Laplace problem. 142
4.22 Final $\hat{e}_{L^1}$ values for varying number of basis functions, C, for the 2D multi-scale Laplace problem. PT model results are included for reference and are not a direct comparison. 145
4.23 Final $\hat{e}_{L^1}$ values for varying number of hidden layers, h and varying number of subdomains, J, for the 2D multi-scale Laplace problem. PT results are for reference only, and are not a direct comparison. 150
4.24 Final $\hat{e}_{L^1}$ values and average run time for varying number of hidden layers, h, with decreasing number of basis functions, C, for the 2D multi-scale Laplace problem. 153
4.25 Final $\hat{e}_{L1}$ values and average run time for varying n_ω and number of subdomains, J, for the 2D multi-scale Laplace problem. 155

Chapter 1

Introduction

In recent decades, machine learning has undergone an extraordinary transformation, enabled by large-scale data storage capabilities, the wider availability of powerful GPUs and open source libraries. Once largely confined to pattern recognition and data-driven prediction tasks, it now plays a central role in all scientific fields ranging from healthcare and finance to climate modelling and scientific discovery. Large-scale neural networks have demonstrated remarkable capabilities as universal function approximators, capable of representing highly complex, nonlinear mappings. Although these results have been known since 1950, publicly visible systems such as large language models and generative AI models have further accelerated research interest and societal integration of machine learning technologies.

At a theoretical level, neural networks possess universal approximation properties, enabling them to represent broad classes of functions with arbitrary accuracy under some mathematical assumptions. At a more practical level, advances in optimisation algorithms, automatic differentiation, and hardware acceleration have made it feasible to build these approximations by training large models efficiently. However, these developments naturally raise an important question: can neural networks be leveraged not only to analyse data, but also to solve the fundamental equations governing physical systems?

Scientific Machine Learning (SciML) has emerged as a field that seeks to bridge ma-

chine learning with scientific computing[1]. Rather than replacing physics-based models with purely data-driven surrogates, SciML aims to integrate physical laws, structural priors, and governing constraints directly into learning algorithms. The central objective is to combine the expressive flexibility of neural networks with the reliability, interpretability, and consistency of established scientific models.

The scope of SciML is broad and rapidly expanding. Neural networks have been employed to discover previously unknown governing equations directly from observational data, enabling data-driven system identification in complex partial differential equations. They have been used to construct surrogate models that approximate expensive numerical simulations, significantly reducing computational cost in repeated-query settings through machine learning-based reduced-order modelling. In other contexts, ML techniques have been integrated into classical numerical pipelines in order to accelerate traditional solvers. More recently, operator learning frameworks have been developed to approximate mappings between infinite-dimensional function spaces, allowing neural networks to learn solution operators rather than individual solutions. Additionally, physical principles such as conservation laws, symmetries, and boundary conditions are increasingly embedded directly into optimisation procedures, ensuring that the learned models remain consistent with the known scientific structure. Through these diverse approaches, SciML seeks not only to improve computational efficiency, but also to rethink how data-driven and physics-based methods can be systematically combined.

Among the various approaches within SciML, Physics-Informed Neural Networks (PINNs) have attracted particularly significant attention. First formalised by Raissi, Perdikaris, and Karniadakis, PINNs incorporate the governing partial differential equation directly into the neural network loss function. Instead of relying solely on training data, the model is trained by minimising the residual of the differential equation in some norm, together with boundary and initial conditions as additional penalties. In this way, the solution of a PDE is reformulated as a continuous optimisation problem over the parameters of a neural network. By design, PINNs offer several attractive theoretical and practical properties. Because the governing equations are enforced

[1]The term was formally introduced in a 2019 U.S. Department of Energy report [1].

through automatic differentiation rather than mesh-based discretisation, PINNs are inherently mesh-free and do not require structured grids (however one has to sample points at which residuals are evaluated). Their formulation is, at least in theory, dimension-agnostic, making them attractive for high-dimensional problems where classical discretisation methods become prohibitively expensive. The implementation is often straightforward, leveraging modern automatic differentiation frameworks to compute derivatives of arbitrary order. Moreover, forward and inverse problems can be handled within a unified framework (since in both cases an optimisation problem has to be solved), allowing unknown physical parameters to be inferred alongside solution fields.

Despite these advantages, classical PINNs face several well-documented challenges. Neural networks trained via gradient descent exhibit a spectral bias toward low-frequency components, making high-frequency or multi-scale solutions difficult to resolve accurately and a prescribed accuracy is sometimes impossible to achieve. The resulting optimisation problem is often stiff and poorly conditioned, leading to slow convergence and sensitivity to hyperparameter choices. As problem complexity increases, particularly in multi-scale or highly oscillatory regimes, these issues become more pronounced, limiting scalability and reliability. These limitations motivate the search for structured architectural and optimisation strategies that retain the flexibility of PINNs while improving convergence behaviour and numerical stability.

The focus of this work is Finite Basis Physics-Informed Neural Networks (FBPINNs), a domain-decomposed variant of PINNs designed to improve scalability and convergence in oscillatory or multi-scale regimes. By partitioning the domain into subdomains and constructing local neural approximations, FBPINNs address some spectral bias and localisation challenges of global PINNs. A natural question then arises:

Can we further improve efficiency and stability by linearising part of the optimisation process?

Extreme Learning Machines (ELMs) provide an appealing framework in this direction. In ELM-style models, hidden-layer weights are randomly initialised and frozen, and only the output layer is optimised via linear least squares. This converts a non-

linear optimisation problem into a linear one, dramatically reducing computational complexity.

However, random feature spaces introduce their own limitations in expressivity and conditioning. This thesis investigates the following central questions:

1. How does linearising the optimisation of FBPINNs affect convergence speed, solution accuracy, and computational efficiency?
2. Can supervised pre-training of hidden-layer basis functions improve the performance of ELM-FBPINN models?
3. How does pre-training influence matrix conditioning, singular values, and rank properties of the resulting least-squares systems?
4. Under what regimes (frequency, dimensionality, depth, domain decomposition) does pre-training provide the greatest benefit?

These questions are investigated systematically through controlled ablation studies in both one-dimensional and two-dimensional PDE settings. The structure of the thesis is as follows:

Thesis Overview and Contributions

This thesis develops the progression from domain-decomposed PINNs, to linearised optimisation strategies, and finally to feature space approximation through supervised learning of the hidden-layer basis for scalable scientific machine learning. Each chapter builds on the previous one, gradually refining the balance between expressivity, stability, and computational efficiency.

Chapter 2: Research Background Chapter 2 provides an overview of the research background that has informed the direction of this work. It begins by formally introducing PINNs, and discusses some of the advantages and disadvantages of the method. Then, a number of alternative extensions to PINNs are presented, before a more in-depth introduction of FBPINNs and ELMs, the primary focus of this thesis, is given.

Chapter 3: Linearised Optimisation via ELM-FBPINN While FBPINNs improve representation through localisation, their training still relies on nonlinear gradient-based optimisation, which can be slow and sensitive to hyperparameters. Chapter 3 addresses this limitation by introducing a partially linearised formulation inspired by Extreme Learning Machines (ELM). In the proposed ELM-FBPINN model, hidden-layer weights are randomly initialised and frozen, and only the output layer is optimised via a linear least-squares solve. This reformulation transforms the training problem from a fully nonlinear optimisation into a structured linear algebra problem. The chapter provides a detailed numerical study of its conditioning, rank properties, and convergence characteristics. A systematic empirical comparison between FBPINN and ELM-FBPINN reveals a clear trade-off: linearisation substantially accelerates optimisation and reduces computational cost, but the use of random fixed features can limit representational power and degrade stability in certain regimes.

Chapter 4: Pre-Trained Hidden-Layer Basis Functions Chapter 4 presents the central methodological contribution of this thesis: the introduction of supervised pre-training for hidden-layer basis functions within the ELM-FBPINN framework. The proposed PT-ELM-FBPINN approach retains the computational efficiency of the linear least-squares formulation but replaces random hidden-layer weights with weights that have been pre-trained on representative solution samples. By aligning the hidden-layer basis with the solution manifold prior to linearisation, the method improves matrix rank, stabilises conditioning behaviour, and accelerates convergence of the LSQR solver. The chapter conducts extensive ablation studies in both one-dimensional and two-dimensional PDE settings, systematically isolating the effects of domain decomposition, network width, hidden-layer depth, solution frequency, and pre-training duration. The empirical results demonstrate that pre-training consistently improves convergence speed and robustness, and in many regimes closes the performance gap between linearised solvers and fully trainable FBPINNs.

Publications

Below is a list of works that have been generated or are planned for publication, directly as a result of this thesis:

1. Anderson, S., Dolean, V., Moseley, B., & Pestana, J. (2024). ELM-FBPINN: efficient finite-basis physics-informed neural networks. Proceedings 28th International Conference on Domain Decomposition Methods (DD28) https://www.ddm.org/DD28/Proc-28.php - **Published**

2. Anderson, S., Dolean, V., Moseley, B. et al. ELM-FBPINNs: an efficient multilevel random feature method. Mach. Learn. Comput. Sci. Eng 2, 23 (2026). https://doi.org/10.1007/s44379-026-00071-1 - **Published**

3. PT-ELM-FBPINN: performing feature space approximation via supervised learning of hidden-layer basis functions to improve ELM-FBPINN performance - **in preparation**.

Publication 2 will further expand on publication 1, presenting a wider, more extensive numerical study of ELM-FBPINN. These works form the basis of Chapter 3. Publication 3 will present the full numerical study of PT-ELM-FBPINN, comparing directly to FBPINN and ELM-FBPINN. This work forms the basis of Chapter 4.

Chapter 2

Research Background

This chapter surveys the development of physics-informed neural networks (PINNs) as a framework for solving partial differential equations (PDEs), with particular emphasis on their formulation, practical applications, and known limitations. After introducing the core PINN methodology, the chapter reviews a broad range of extensions proposed to address challenges related to optimisation, expressivity, spectral bias, and scalability. Special attention is given to domain decomposition strategies and structured representations that draw inspiration from classical numerical methods. Within this context, finite basis physics-informed neural networks (FBPINNs) and extreme learning machines (ELMs) are examined in detail, as they offer complementary mechanisms for improving efficiency and robustness by combining partition-of-unity concepts with linear or weakly non-linear solvers.

2.1 Physics-Informed Neural Networks - PINNs

Physics-informed neural networks are a type of machine learning methodology for solving partial differential equations. Originally published in 2017 as a two-part article [2,3], before being republished as a single work in 2019 [4], the seminal work that introduced PINNs by Raissi, Perdikaris, and Karniadakis has become a widely-cited paper, with thousands of citations across a growing number of fields. This level of acknowledgement both highlights the influence of the publication, and signals the significant expansion of

research into PINNs and physics-informed machine learning (PIML). PINNs have become a very popular solver for PDEs, due to their ease of implementation and potential application to problems with high dimensionality. Crucially, they leverage the wealth of knowledge of the underlying physical systems, and by constraining the optimisation of the loss function with the physical laws, PINNs rely significantly less on high-quality, labelled data when compared with traditional machine learning models [4].

PINNs are designed to solve the following boundary value problem: find $u(\mathbf{x})$ such that

$$\mathcal{N}[u(\mathbf{x})] = f(\mathbf{x}),\ \mathbf{x} \in \Omega \subset \mathbb{R}^d,\ \text{with}\ \mathcal{B}_k[u(\mathbf{x})] = g_k(\mathbf{x}),\ \mathbf{x} \in \Gamma_k \subset \partial\Omega, \tag{2.1}$$

where $\mathcal{N}[u(\mathbf{x})]$ is some differential operator, $u(\mathbf{x})$ is the problem solution, and $\{\mathcal{B}_k[\cdot]\}$ is a set of K boundary conditions defined such that the solution is uniquely determined. PINNs use a neural network, $u(\mathbf{x}, \boldsymbol{\theta})$, to directly approximate the solution, i.e. $u(\mathbf{x}, \boldsymbol{\theta}) \approx u(\mathbf{x}) \in \mathbb{R}^{d_{out}}$, where d_{out} is the output dimension of the solution. The vector $\boldsymbol{\theta}$ contains all the network parameters, commonly known as the weights and biases. The PINN is trained through the optimisation of the following loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = \underbrace{\frac{\lambda_I}{N_I}\sum_{i=1}^{N_I}\left(\mathcal{N}[u(\mathbf{x}_i, \boldsymbol{\theta})] - f(\mathbf{x}_i)\right)^2}_{\mathcal{L}_{\text{PDE}}} + \underbrace{\sum_{k=1}^{K}\frac{\lambda_B^{(k)}}{N_B^{(k)}}\sum_{i=1}^{N_B^{(k)}}\left(\mathcal{B}_k[u(\mathbf{x}_i^{(k)}, \boldsymbol{\theta})] - g_k(\mathbf{x}_i^{(k)})\right)^2}_{\mathcal{L}_{\text{BC}}}, \tag{2.2}$$

where $\{\mathbf{x}_i\}_{i=1}^{N_I}$ is a set of collocation points sampled in the interior of the domain, $\{\mathbf{x}_i^{(k)}\}_{j=1}^{N_B^{(k)}}$ is a set of points sampled along each Γ_k, and λ_I and $\lambda_B^{(k)}$ are scalar positive weights which ensure the terms in the loss function are well balanced. The term $\mathcal{L}_{\text{PDE}}$ ensures the PDE is satisfied on the interior of the domain, directly penalising the residual of the underlying equation (hence the neural network is physics-informed). The boundary loss, $\mathcal{L}_{\text{BC}}$, ensures that the solution obeys the K boundary conditions. Using the output of this loss function, the network weights at each epoch are updated, typically using gradient-based optimisation strategies such as stochastic gradient de-

scent (SGD), Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) [5], and Adaptive Moment Estimation (Adam) [6].

The work was built upon earlier ideas presented in [7], which used Gaussian Processes (GP) to infer solutions to linear integro-differential operators. In this earlier work, the solution $u(\mathbf{x})$ was modelled as a Gaussian Process prior, and the governing linear differential operator was explicitly applied to the GP, yielding a transformed GP that represents the distribution of the PDE residual. By conditioning the joint GP on available observational data and enforcing consistency with the governing equation, the authors obtained a probabilistic solution to the PDE that naturally quantified uncertainty. Importantly, the linearity of the operator allowed the physics to be encoded analytically at the kernel level, leading to data-efficient inference in a fully mesh-free setting. While this GP-based method had its limitations, it established several foundational principles later adopted by PINNs: operator-aware learning, collocation-based enforcement of physics, and the use of physical constraints to dramatically reduce data requirements.

A critical technique employed by PINNs is the leveraging of automatic differentiation (AD) for evaluation of PDE residuals, as well as evaluating gradients used for training network weights. Not only does this allow for efficient, accurate calculation of differential operators, it bypasses the need for expensive mesh generation, which is replaced instead by a collocation method. Most popular Python backends (Tensorflow [8], Pytorch [9], JAX [10]) implement AD without any additional user effort, making PINNs a very accessible option for solving PDEs. Additionally, bespoke libraries have been developed specifically for physics-informed machine learning (PIML), including DeepXDE [11], Nvidia's Modulus [12], and SciANN [13].

The flexibility offered in the mesh-free environment makes PINNs adaptable for both forward and inverse problems. In [4], Raissi et al. demonstrated PINNs' ability to infer parameters even from incomplete, noisy data. Using a high-fidelity data set generated using a finite element method (FEM) solver, and sampling only 1% of the total points for input data, the authors were able to infer two unknown parameters and the entire pressure field of a set of 2D Navier–Stokes equations. Notably, the pressure

field was approximated without any input data of the pressure, only inferring the field through the input data of the velocity and the minimisation of the residual. This result demonstrates PINNs' potential for solving high-dimensional problems with low-volume, high-quality input data.

Despite the impressive results in these numerical demonstrations, the application of PINNs to complex problems highlighted several challenges and open questions that have shaped much of the subsequent PINN literature. These include the difficulty of balancing competing loss terms (a problem which is not unique to PINNs), the sensitivity of training to the choice of collocation points and network architecture, and the degradation of performance for problems exhibiting multiscale behaviour or high-frequency solutions. It should be noted that, despite their conceptual simplicity, PINNs introduce a non-trivial optimisation problem whose landscape is strongly influenced by the underlying physics. These observations foreshadowed later work on adaptive loss weighting, domain decomposition, operator learning, and hybrid methods combining PINNs with classical numerical solvers.

2.2 Applications

Having outlined the methodological foundations of PINNs and their early numerical demonstrations, it is natural to examine how these ideas have translated into practical applications across scientific and engineering domains. The following examples illustrate both the breadth of problems addressed by PINNs and the types of advantages most frequently reported in applied settings.

Owing to the mesh-free flexibility, the potential to approximate solutions of high-dimensional problems, ease of implementation, and the leveraging of AD, PINNs have been applied in a wide range of fields, both for forward and inverse problems. In biomedicine, PINNs have been used to infer unknown parameters in ODEs that represent the glucose-insulin regulatory system [14]. The unique systems biology-informed neural network (SINN) was able to infer the hidden dynamics of the system of ordinary differential equations (ODEs), even on noisy measurements. It was also noted that the

regularisation of the model was implicitly enforced through the minimisation of the residual ODE loss term. In [15], the authors presented a novel method for estimating electrical activation in the heart using a PINN to represent the activation time and conduction velocity. They noted that PINNs outperformed more traditional machine learning methods, Gaussian process regression, and a linear interpolation method. For error estimation they also trained several PINNs at once, observing that even on an 8-core laptop processor they were able to train each sample in roughly one minute. In 2023, Calicchia et al. used a PINN to reconstruct the pressure field around a swimming fish, similarly to the 2019 result by Raissi et al. but this time on an non-trivial object in motion [16]. Compared to a traditional method, Queen 2.0 [17], the authors found PINN to be more flexible, particularly for the application of boundary conditions, and more accurate when reducing the resolution of the input particle image velocimetry (PIV) data. PINNs have also been studied in geophysics [18–20], optics [21, 22], heat transfer [23, 24], and nuclear physics [25, 26]. Though this is a non-exhaustive list of areas in which PINNs have been utilised, the growing interest across all fields of science is clear, with Figure 2.1 demonstrating the number of papers related to PINNs from 2019 to 2025, and a pie chart showing the percentage of papers written for each subject area. These plots were generated on the Scopus database by searching "TITLE-ABS-KEY ("physics informed neural network" OR "pinn") AND PUBYEAR AFT 2018 AND PUBYEAR BEF 2026 " [27].

Across these application domains, several common themes emerge. PINNs are frequently highlighted for their ability to incorporate sparse, noisy, or indirect measurements, their flexibility in handling complex geometries without meshing, and their natural suitability for inverse problems where parameters or hidden fields must be inferred. At the same time, reported successes are often problem-dependent, and many studies emphasise careful tuning of network architectures, loss-weighting strategies, and sampling procedures. These observations suggest that while PINNs provide a powerful and unifying framework, their effectiveness in practice remains closely tied to both the structure of the underlying physical problem and the numerical characteristics of the resulting optimisation task.

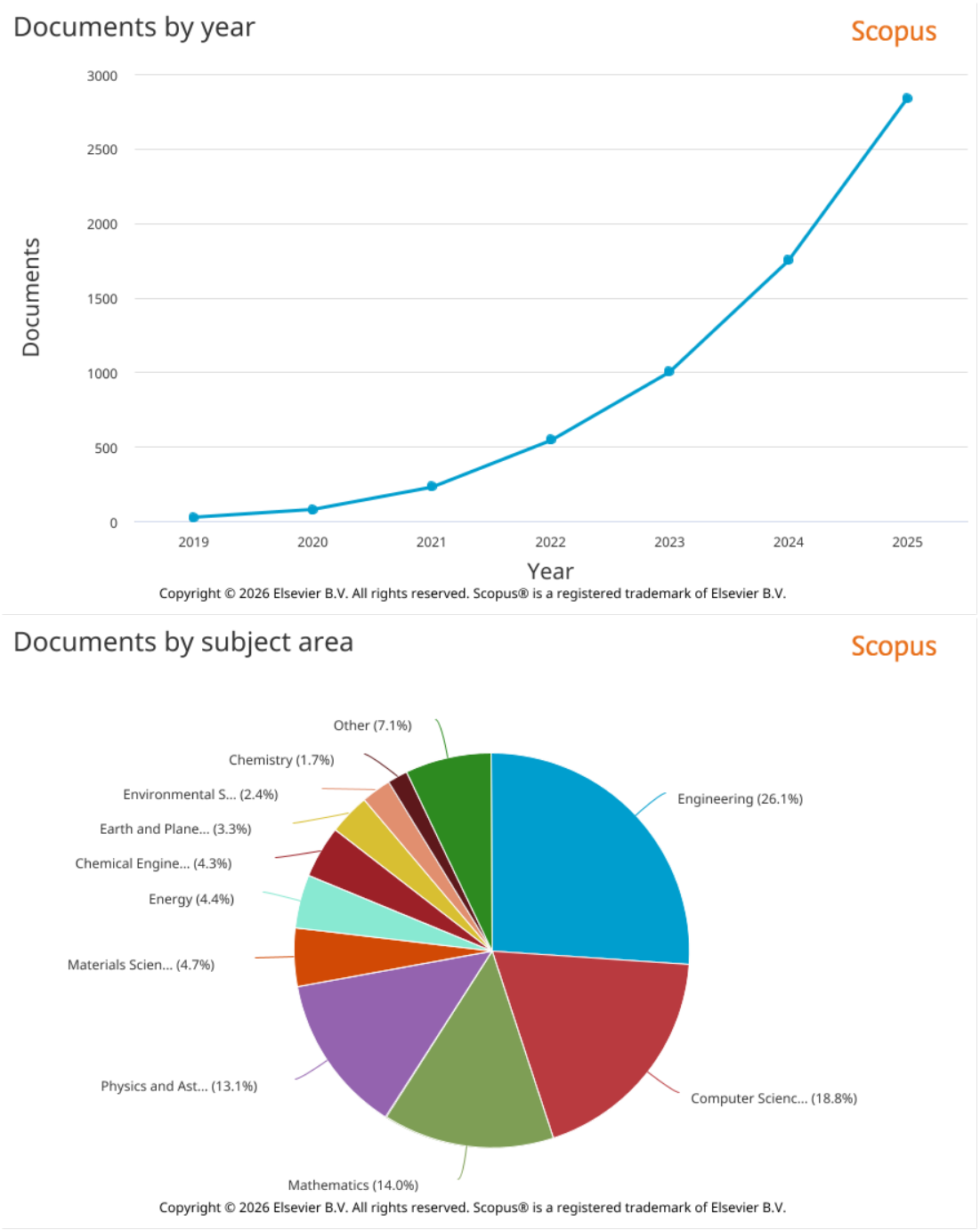


Figure 2.1: Scopus search results by year and subject for 'physics informed neural network' and 'PINN' between 2019 and 2025.

2.3 PINN limitations

In the following sections, we introduce and discuss some limitations of PINNs, before discussing some extensions that look to resolve these limitations. There are a number of ways in which PINNs present additional problems for solving PDEs. These include the non-convex loss landscape, resulting in difficult convergence for gradient-descent based solvers; imbalanced contributions from the PDE and boundary-condition terms in the loss function (a problem that is not unique, but is present in PINNs); differential operators causing stiff and ill-conditioned training dynamics; spectral bias, whereby neural networks preferentially learn low-frequency components before high-frequency components; and scalability for large-scale problems leading to computationally expensive training. These issues have all been addressed through various PINN extensions that seek to either alter the loss function, the neural network architecture, or the input data. While not an exhaustive list, we present a number of these extensions before introducing FBPINNs and ELMs.

Despite their broad applications and ease of implementation, PINNs suffer from a number of limitations. The original PINN described in [4] was built upon a fully connected multilayer perceptron (MLP). Efforts to ensure sufficient approximation capabilities often result in deep neural networks with considerably wide layers, i.e. many neurons. In [4], Raissi et al. adjusted the architecture of the MLP depending on the problem they were solving. For example, to solve the 1D non-linear Schödinger equation, the MLP used had 5 layers, each with 100 neurons, while for the discrete-time Allen–Cahn equation, 5 layers with 200 neurons per layer were used. The authors also conducted a study on increasing the layers and neurons, using Burger's equation as the example problem. The results indicated that increasing layers and neurons shows improvement on the relative L^2 error of the predicted solution, as well as a general trend of reduced errors when predicting unknown variables. These improvements were attributed to the increase in expressivity of the MLP. While sufficient width and depth of networks is necessary, larger models will result in increased training times, particularly as the domain size and solution complexity increase [28]. Balancing network size

with training efficiency is not the only issue that can affect a model's ability to learn. In [29], Wang et al. demonstrated that, due to the PINN loss function landscape being highly non-convex, gradient-based optimisers struggle to converge on the stiff optimisation problem. This work was inspired by prior work on deep neural networks by Glorot and Bengio [30], in which they showed that without proper initialisation, the activation functions used in each layer of deep neural networks tend to saturate quickly, resulting in vanishing gradients, and slow training.

Furthermore, this training difficulty is exacerbated by neural networks' tendency to learn low frequency features of solutions much faster than the high frequency features; a phenomenon known as spectral bias [31]. While not fully understood, there have been some key insights described in recent literature on this aspect of neural network training [32–35]. Through studying the neural tangent kernel (NTK) [36], Jacot et al. demonstrated that there is a link between the size of eigenvalues of the NTK matrix and training time of the associated MLP. PINNs have also been proven to exhibit spectral bias under the same NTK analysis [37], and this property is present in problems with multi-scale solutions [38]. In [39], the authors used a natural gradient descent algorithm, among other techniques aimed at improving computational cost, to significantly speed up the convergence of PINNs. To address spectral bias, a significant research effort into extending the architecture of PINNS has been undertaken across a broad range of applications, many of which will be discussed later in Section 2.4.

In addition to architectural and optimisation challenges, the formulation of the PINN loss function itself introduces further practical difficulties. The relative weighting of the PDE residual and boundary or initial condition terms can strongly influence training dynamics, and poor choices may lead to solutions that satisfy only a subset of the imposed constraints. While fixed scalar weights are commonly used, they often require extensive manual tuning and are highly problem-dependent. This issue becomes particularly pronounced in problems with heterogeneous physics, sharp gradients, or strongly-varying solution scales. not a pinn problem only

Another limitation arises from the enforcement of boundary conditions through soft penalty terms rather than exact satisfaction. Although this weak enforcement offers

flexibility, it may result in boundary inaccuracies, especially when boundary layers or singularities are present. In contrast to classical numerical methods, where boundary conditions are imposed exactly at the discrete level, PINNs rely on optimisation to reconcile competing objectives, which may not always yield physically consistent solutions.

Sampling strategies for collocation points also play a critical role in PINN performance. Uniform random sampling may fail to adequately resolve regions of high solution complexity, while adaptive or residual-based sampling introduces additional computational overhead and algorithmic complexity. Moreover, as the dimensionality of the problem increases, the number of collocation points required to accurately represent the solution can grow rapidly, undermining one of the original motivations of PINNs for high-dimensional problems.

Finally, scalability remains a significant challenge. Training PINNs for large-scale, time-dependent, or three-dimensional problems can be computationally expensive, often requiring long training times and careful hardware utilisation. These limitations have motivated a growing body of research aimed at improving optimisation robustness, incorporating domain decomposition, enriching network architectures, and developing hybrid approaches that combine PINNs with classical numerical solvers.

2.4 PINN Extensions

Given the limitations of PINNs, significant research effort has been directed towards extending PINNs to improve precision, efficiency, scalability, and address the spectral bias phenomenon. This has been approached in a number of ways, from alteration of the machine learning model architecture, to adjustments to the PINN loss function. In this section, a selection of various PINN extensions will be discussed, with finite-basis physics-informed neural networks (FBPINNs) and extreme learning machines (ELMs) further expanded on.

From a methodological perspective, most PINN extensions can be broadly categorised according to the primary aspect they modify: (i) the *representation* of the

solution through architectural changes, (ii) the *training dynamics* via loss reweighting or feature mappings, and (iii) the *numerical formulation* through variational principles or domain decomposition. This taxonomy provides a useful lens through which the rapidly growing body of PINN variants can be interpreted.

Many extensions of PINNs have been focused on the representation model architecture. With all neural networks exhibiting universal approximation properties, including the foundational MLPs [40], machine learning research in recent years has taken advantage of more complicated architectures and procedures. By leveraging the machine learning research from the last decade, PINNs can be more expressive, training accuracy and speed can both improve, and the spectral bias phenomenon can be mitigated.

For example, Gao et al. developed PhyGeoNet [41], which combines convolutional neural networks (CNN) with PINNs. While powerful, CNN architecture is designed for image-like data, i.e. uniformly distributed points within a rectangular domain. This hinders CNN's ability to resolve approximations on non-uniform domains. To circumvent this limitation and apply the CNN to parametric PDEs with irregular geometries, the authors transformed the solution space to a rectangular mesh first. They showed that PhyGeoNet accurately converged on solutions, agreeing with simulated test data, while also outperforming PINN in both accuracy and training time. This methodology demonstrates not just an architectural extension to PINN, but also a pre-processing of the domain, which is commonly seen in recent research. CNN is a popular choice of model architecture, and has been combined with PINN across a wide range of fields, including heat transfer [42], mechanical engineering [43], fluids [41, 44], and marine biology [45].

Recurrent Neural Networks (RNNs) are another powerful architecture that has seen some usage in the application of PINNs. In [46], Ren et al. developed a physics-informed convolutional-recurrent network (PhyCRNet) for solving spatiotemporal PDEs. This architecture leverages the convolutional long short-term memory (ConvLSTM) RNN to allow for spatial and temporal dimensions to be solved simultaneously, with the hidden and cell states being recurrently used for each time step, t. This is combined with an encoder and decoder to allow the model to approximate in a low-dimensional latent

space, before reconstruction. At each time step, the model reuses the previous time step's hidden state, cell state, and solution. The model is then optimised by minimising an aggregated average PINN loss function across all time steps. The authors found that PhyCRNet was able to evaluate each of their problem examples more accurately than PINN, particularly on new test examples. The authors also demonstrated the ability to generalise well to randomly-sampled initial conditions, in contrast to PINN, which was unable to generalise.

Yang et al. developed a physics-informed generative adversarial network (PI-GAN) [47]. GANs are a type of neural network that aims to train a generator model's ability to produce new simulated data that accurately represents a real data set. This is performed using adversarial training by also using a discriminator network that learns to distinguish between real and simulated data. PI-GANs take advantage of this two player zero-sum formulation for predicting solutions to stochastic differential equations (SDEs), ensuring that the simulated solutions obey the underlying physical law. PI-GAN was able to approximate Gaussian processes well with the accuracy of the predictions increasing with additional input data.

Using kernel regression theory, and considering the infinite width limit of neurons in an MLP, neural tangent kernel (NTK) theory by Jacot et al. [36] shows that the training dynamics of the neural network depends on the eigenvalues of the NTK matrix. In practice, this results in large eigenvalues corresponding to faster decay in the gradients, which leads to faster training for low frequency features. Inversely, smaller eigenvalues correspond to slower gradient decay and slower learning for high frequencies. This is one explanation for the spectral bias observed in the training of MLPs. In [37], Wang et al. showed, using the NTK, that PINNs also exhibit spectral bias in the same way that MLPs do, but with the additional complexity of loss term imbalance between the residual and boundary losses. They suggested a novel algorithm for updating balancing terms in the loss function and found that this improved final accuracy when compared to a default optimiser.

Tancik et al. [48] showed that they could use Fourier feature mapping on the input vector, which mitigates spectral bias by rescaling the eigenvalues seen in the NTK

matrix. This rescaling shifts the eigenvalues to be more balanced across frequencies, which improves training convergence on the higher frequencies. This was then extended to PINNs in [38], with the results again showing that the input feature mapping improves convergence speed and accuracy. When testing their method on a simple high-frequency sinusoidal toy problem, they found that increasing the frequency coefficient of their feature transformation correlated with significantly faster convergence. Notably, they found that higher-frequency inputs showed higher frequencies of their solution being resolved in early iterations, which contrasts with PINN and general MLP training dynamics.

Typically, PINNs use standard activation functions like ReLu, tanh, and sigmoid functions, as in traditional machine learning applications. In [49], Uddin et al. replaced the typical activation functions, with wavelet functions – specifically the Morlet, the Mexican hat and the Gaussian wavelet functions. The wavelets-based PINNs were tested on the Blasius equation, as well as a pair of coupled problems, and Burger's equation. In each test case, the wavelets-based PINNs outperformed PINNs with tanh activation functions. The authors note that similarly to many choices in neural network training, some trial and error may be required to find the optimal wavelets choice for a given problem.

Based upon the radial basis functions, first described in 1988 by Broomhead and Lowe [50], physics-informed radial basis network (PIRBN) also look to replace the general activation functions with radial basis functions. Specifically, Bai et al. [51] focused on Gaussian functions, and tested the model on a number of high-frequency, multi-scale function approximations. In all cases, PIRBN was more stable and more accurate than baseline PINN. They also analysed the PIRBN with NTK theory, proving that despite changing the activation functions, the NTK matrix should remain unchanged during training, and the PIRBN should remain a local approximator. This is ideal when attemping to solve for higher frequencies as the network's individual approximations per point are robust against specific changes in weights.

Variational PINNs (VPINNs) [52] and *hp*-VPINNs [53], take inspiration from traditional finite element methods and try to tackle the accuracy and training costs of

PINN through use of the variational form of the loss function, with *hp*-VPINNs also introducing *hp*-refinement of the domain. In [52], Kharazmi et al. showed that by introducing the variational form of the loss function, and using the Petrov–Galerkin method, they were able to improve the precision of their approximations compared to a regular PINN. The *hp*-PINNs also required less data for solutions containing shocks and sharp turning points, while still improving overall performance.

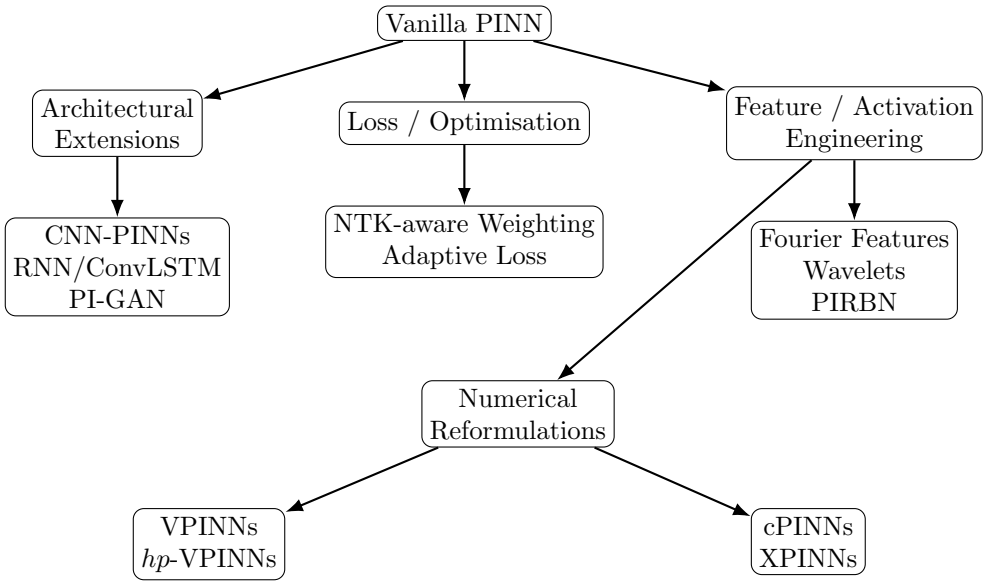


Figure 2.2: Taxonomy of PINN extensions based on their primary methodological modification.

Extended PINNs (XPINNs) [54] are an extension of conservative PINNs (cPINNs) [55], both of which employ domain decomposition to improve the training time and accuracy. Both methods divide the problem domain into smaller subdomains, approximating each with separate neural network and enforcing continuity at the boundary interfaces. While cPINNs were designed specifically for conservation laws, XPINNs can be applied to any generic PDE, can be decomposed in time as well as space, and have simpler interface boundary conditions. This division into subdomains allows for the individual neural networks to solve lower frequency sub-problems, in turn leading to simpler model architecture needs. In [54], XPINN was demonstrated to be effective in approximating the viscous Burger's equation with two subdomains and a complex

interface boundary between the two neural networks. Alhubail et al. demonstrated XPINNs' ability to solve highly heterogenous fluid problems, which vanilla PINN is unable to do [56]. However, they did note that without imposing continuity on the pressure and flux, the model did not converge properly. Additionally, they identified the training time as lengthy, with 50,000 iterations required in order to stop observing significant drops in the loss value, with no guarantee that the models were converging to a correct solution. In [57], Dong et al. combined XPINN with three different feature mapping architectures, including a wavelet-based multi-scale neural network. By applying these models to a variety of multi-scale fluid, flow, and pressure dynamics problems, they showed that XPINN in all three configurations outperformed PINN significantly in both convergence time and final approximation precision. They also noted that the wavelet-based network outperformed the other two models, both of which were Fourier based.

Category	**Representative Methods**	**Main Motivation**
Architectural extensions	CNN-PINNs, RNN/ConvLSTM PINNs, PI-GANs	Improve expressivity, exploit spatial or temporal structure, enable stochastic modelling
Feature and activation engineering	Fourier features, wavelet PINNs, PIRBN	Mitigate spectral bias, improve convergence on high-frequency or multi-scale solutions
Loss formulation and optimisation	NTK-aware weighting, adaptive loss balancing	Address stiff optimisation landscapes and loss imbalance
Variational formulations	VPINNs, *hp*-VPINNs	Improve numerical consistency, reduce training cost, leverage FEM principles
Domain decomposition	cPINNs, XPINNs	Improve scalability, handle heterogeneity, reduce model complexity per subdomain

Table 2.1: Summary of major PINN extensions categorised by their primary methodological focus.

A summary of the key developments on the original PINN methodology described in this section is given in Figure 2.2 and Table 2.1. In addition to these extensions, finite-basis physics-informed neural networks (FBPINNs) and physics-informed extreme

learning machines (ELMs) are particularly interesting, as they depart from standard end-to-end training paradigms and instead introduce structured basis representations that more closely resemble classical numerical discretisations. These methods are discussed in detail in the following sections.

2.5 FBPINNs

Finite basis physics-informed neural networks (FBPINNs) can be viewed as a natural continuation of domain decomposition ideas in PINNs, but with a formulation that is more closely aligned with classical numerical analysis concepts, particularly partition-of-unity and finite element methods (FEM).

Similarly to the domain decomposition techniques utilised in XPINNs and *hp*-VPINNs, finite basis physics-informed neural networks (FBPINNs) also look to divide the global problem domain to reduce the local problem complexity for subdomain networks [28, 58]. The primary difference from the previous techniques is the use of continuous, partition-of-unity, window functions, that streamline and simplify subdomain boundary continuity. Instead of additional loss terms that ensure continuity at the subdomain interface boundaries, FBPINNs employ smooth window functions to confine each subdomain network. More specifically, the global solution domain, Ω, is decomposed into J overlapping subdomains, $\Omega = \cup_{j=1}^{J} \Omega_j$, and a neural network $u_j(\mathbf{x}, \boldsymbol{\theta}_j) \in \mathbb{R}^{d_{out}}$, is assigned to each subdomain. The subdomain overlap is essential to guarantee rapid transmission of information between subdomains while preserving the locality of each subdomain approximation. Furthermore, the domain decomposition chosen is problem dependent, with decomposition complexity matching the solution complexity. (FB)PINNs can be adapted to solve both regular and irregular global domains. The global PDE solution approximation is then given by:

$$u(\mathbf{x}, \boldsymbol{\theta}) = \mathcal{C} \left[\sum_{j=1}^{J} \omega_j(\mathbf{x}) \cdot \text{unnorm}(u_j(\text{norm}(\mathbf{x}), \boldsymbol{\theta}_j)) \right]. \tag{2.3}$$

Here $\mathcal{C}$ is a hard constraining operator that appropriately enforces boundary condi-

tions on the global solution. The window functions, $\omega_j(\mathbf{x})$, are used to locally confine each subdomain network to its subdomain, i.e. $\text{supp}\big(\omega_j\big) \subset \Omega_j$, and they typically obey a partition of unity, such that $\sum_{j=1}^{J} \omega_j \equiv 1$ on Ω. This ensures that the local subdomain solutions are appropriately weighted when recombined into the global solution, preserving scale while providing smooth transitions across overlapping subdomains. Each input is normalised within the subdomain, using some 'norm' function, making each local problem lower frequency and thus mitigating spectral bias. The subsequent subdomain solutions are then transformed back to the global solution domain via an 'unnorm' function. The same loss function as PINNs can be used to train FBPINNs by substituting (2.3) into (2.2) and noting that (2.3) simply defines a custom neural network architecture for the PINN. Note that the constraining function, $\mathcal{C}$ on $u(\mathbf{x}, \boldsymbol{\theta})$ eliminates the need for boundary terms in the loss function, if chosen appropriately.

The use of smooth window functions satisfying a partition-of-unity property draws a strong parallel with classical and generalised FEM, where global solutions are assembled from local basis functions with compact support. In this sense, FBPINNs can be interpreted as replacing fixed polynomial basis functions with learnable, data-driven neural basis functions, while preserving key numerical properties such as locality and continuity.

By breaking the global PINN optimisation problem into many smaller (coupled) local optimisation problems, Moseley et al. showed that FBPINNs can significantly improve the accuracy of PINNs when solving problems with highly multi-scale solutions [28]. The reason for this improvement in accuracy is in part due to the domain decomposition of the highly oscillatory global function into smaller, normalised, low-frequency functions. These lower frequency functions inside the subdomains are much simpler for the neural networks to approximate, which addresses the problem of the spectral bias. The speed of convergence also sees significant improvement, by taking advantage of the parallelisation of networks across GPU cores. Each subdomain network can be run on separate threads in both the forward and backward passes, considerably reducing the training time. FBPINNS were also extended to a multilevel domain decomposition architecture, inspired by classical methods [58]. Both standard

and multilevel FBPINNS were tested on a variety of problems including low and high frequency 1D and 2D problems with sinusoidal solutions, a (2+1)-dimensional Burger's equation, and a multi-scale Laplace problem in 2D. In all cases, the FBPINN outperformed comparison PINNs. The authors note that the FBPINN is more data efficient, with FBPINN converging faster, with smaller networks. It should be noted that the use of separate neural networks in each subdomain will likely result in a larger number of parameters, leading to an increase in memory usage and FLOPs, despite the improved run time.

Despite these advantages, the optimisation of FBPINNs remains fundamentally non-linear and gradient-based, which motivates further investigation into alternative training paradigms that more closely resemble the linear or weakly non-linear solvers commonly used in classical PDE discretisations.

2.6 Extreme Learning Machines (ELMs)

One such alternative paradigm is offered by extreme learning machines (ELMs), in which the iterative gradient-based training is replaced with a direct solution of a linear least-squares problem.

Despite the improvements in performance, FBPINNs still suffer from drawbacks that are common amongst PINN methods. The global loss function is largely unchanged from the original PINN loss, and the highly non-convex nature is still a challenge when using gradient descent based solvers. Compared to traditional methods like FEM, which reduce the PDE optimisation problem to a linear system of equations, FBPINNs are computationally expensive. One promising method to accelerate the training are extreme learning machines (ELMs) [59]. An ELM is a neural network where only the weights and biases of its last layer are trained; the remaining parameters are randomly initialised and frozen. This transforms the training regime of the neural network from a gradient descent based optimisation to a linear least-squares problem for the output weights, similarly to FEM.

From a numerical perspective, ELMs can be interpreted as fixed-basis methods,

where the randomly initialised hidden-layer features act as a non-adaptive basis, and the optimisation reduces to computing the optimal coefficients in this basis.

Consider a shallow neural network architecture, where the randomly initialised, frozen, parameters are the weights, $\mathbf{w}_c$, and biases, b_c, $c = 1, \ldots, C$, of the hidden layer. To simplify the introduction of ELMs, it is useful to consider the approximation of a function $g(\mathbf{x})$, $\mathbf{x} \in \Omega \subset \mathbb{R}^d$. In this case, the ELM approximant, $u_{\text{ELM}}(\mathbf{x}, \mathbf{a}) \in \mathbb{R}^{d_{out}}$, is given by

$$u_{\text{ELM}}(\mathbf{x}, \mathbf{a}) = \sum_{c=1}^{C} a_c \sigma(\mathbf{w}_c \cdot \mathbf{x} + b_c) \ , \tag{2.4}$$

where $\mathbf{a} = [a_1, \ldots, a_C]^T$ contains the trainable weights of its last layer. Given a set of collocation points $\{\mathbf{x}_i\}_{i=1}^N$, the ELM approximant is trained using the loss function:

$$\mathcal{L}(\mathbf{a}) = \frac{1}{2} \sum_{i=1}^{N} \big(u_{\text{ELM}}(\mathbf{x}_i, \mathbf{a}) - g(\mathbf{x}_i)\big)^2. \tag{2.5}$$

Substituting (2.4) into (2.5), we see that minimising $\mathcal{L}(\mathbf{a})$ is equivalent to solving the least-squares problem

$$\min_{\mathbf{a} \in \mathbb{R}^C} \frac{1}{2} \|\mathbf{M}\mathbf{a} - \mathbf{b}\|_2^2, \quad \mathbf{M} = \begin{bmatrix} \sigma(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & \sigma(\mathbf{w}_C \cdot \mathbf{x}_1 + b_C) \\ \vdots & \ddots & \vdots \\ \sigma(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & \sigma(\mathbf{w}_C \cdot \mathbf{x}_N + b_C) \end{bmatrix}. \tag{2.6}$$

The resulting optimisation problem is convex and admits a unique solution when $\mathbf{M}$ has full column rank, allowing the use of robust numerical linear algebra techniques such as the QR decomposition or singular value decomposition. The algorithm for training an ELM is summaried in Algorithm 1.

Algorithm 1 ELM

1: Randomly assign input weights $\mathbf{w}_c$ and biases b_c, $c = 1, \ldots, C$.
2: Calculate the hidden layer output matrix $\mathbf{M}$.
3: Calculate the output weight vector $\mathbf{a}$ by solving the least-squares problem (2.6).

In [59], Huang et al. demonstrated that ELM training time on a sinc function problem was 170 times faster than the comparison back-propagation algorithm, while the

network attained a significantly improved test loss value. Similarly, the ELM was 10000 times faster than the support vector regression (SVR) model employed for the same problem. This general trend continued for all subsequent tests, which demonstrates ELMs' ability to improve training time when compared to gradient-based solvers. While ELMs were originally designed for single layer feed forward networks, they have also been extended to multilayer networks [60, 61].

Physics-informed extreme learning machines (PIELM) [62], introduced by V. Dwivedi and B. Srinivasan, combine ELM and PINN. Taking advantage of the mesh-free approach, PIELM can be applied to problems involving complex geometry, while reducing the overall training time with efficient numerical methods for optimising the linearised loss function. In [62], PIELM was evaluated on a number of linear and quasi-linear PDEs, and was able to solve these steady and time-dependent PDEs efficiently. However, the authors also noted that PIELM exhibited spectral bias on problems with sharp gradients. To address this limitation the authors introduced distributed PIELM (DPIELM), which decomposes the global domain into localised subdomains.

This observation highlights that, while ELMs significantly reduce optimisation complexity, spectral bias may still arise when the fixed random feature basis is insufficient to represent high-frequency or strongly localised solution components.

While DPIELM is a promising methodology, the decomposition schedule is similar to that of XPINNs, and requires continuity conditions at the subdomain boundaries. FBPINNs smoothly combine the local solutions via window functions that obey a partition of unity. This streamlines the process of reconstructing the global solution, and allows for each local network to learn more information about neighbouring subdomains. From the definition of FBPINN, it is clear that the ELM methodology is compatible, and could address the gradient-based optimisation issues seen in [28] and [58].

This compatibility suggests a natural hybrid direction in which finite-basis, partition-of-unity architectures are combined with ELM-style linear solvers, offering the potential to retain the numerical advantages of FBPINNs while substantially reducing training cost and optimisation stiffness.

2.7 Summary

This chapter has reviewed the foundations of PINNs, their applications across a wide range of scientific domains, and the principal limitations that arise from their reliance on large, fully connected neural networks and gradient-based optimisation. A variety of PINN extensions have been discussed, highlighting recurring strategies such as architectural modifications, feature engineering, variational formulations, and domain decomposition. These developments reflect an increasing convergence between physics-informed machine learning and classical numerical analysis.

In particular, FBPINNs introduce a structured, partition-of-unity framework that alleviates spectral bias and improves scalability by decomposing complex global problems into simpler local sub-problems. However, FBPINNs still rely on non-linear optimisation and inherit many of the challenges associated with gradient-based training. Extreme learning machines (ELMs), by contrast, offer a fundamentally different training paradigm in which optimisation reduces to the solution of a linear least-squares problem, providing substantial gains in computational efficiency.

Motivated by these complementary strengths, the remainder of this thesis focuses on combining the partition-of-unity structure and mesh-free flexibility of FBPINNs with the efficient linear solvers enabled by ELMs. This hybrid approach, referred to as ELM-FBPINN, aims to mitigate the optimisation difficulties inherent to gradient-based PINN methods while retaining their expressive power and flexibility.

Chapter 3

ELM-FBPINN

3.1 Introduction

In this chapter, the performance of combining FBPINN with ELM is studied, with the aim of alleviating the optimisation difficulties commonly encountered in gradient-descent-based physics-informed neural networks without sacrificing accuracy. The new model, called ELM-FBPINN, is compared to a vanilla PINN model, as well as to FBPINN, allowing us to isolate the effect of replacing trainable subdomain networks by randomly initialised, fixed-feature models. All models are evaluated on the same benchmark problems: a 1D damped harmonic oscillator and a 2D multiscale Laplace problem, chosen to test both temporal dynamics and spatial multiscale behaviour.

We attempt to answer the following questions: can replacing the subdomain networks with ELM address the training difficulties seen in gradient-descent-based models? Can model prediction accuracy compare to or exceed that of FBPINN? What are the implications of this replacement in terms of computational efficiency and robustness? Is a single layer of basis functions sufficient in a randomly initialised, frozen network?

Section 3.2 defines the ELM-FBPINN methodology, including the construction of the partition of unity framework and the resulting linear least-squares problem. Section 3.3 describes the problems studied, including the 1D damped harmonic oscillator and the 2D multiscale Laplace problem, and discusses the boundary conditions and source terms used in each case. The common implementation for the models studied is also

described, ensuring a fair and consistent comparison across architectures.

We then present numerical results in Section 3.4, first for the 1D damped harmonic oscillator (Section 3.4.1) and then for the 2D multiscale Laplace problem (Section 3.4.2), focusing on accuracy, convergence behaviour, and sensitivity to hyperparameters. Finally, our conclusions and directions for future work are discussed in Section 3.5.

3.2 The ELM-FBPINN method

In this work, the subdomain networks in the FBPINN method are replaced with ELMs to solve (2.1). The key idea is to retain the domain decomposition and partition-of-unity structure of FBPINNs, while replacing gradient-based training on each subdomain by a linear least-squares solve. More specifically, an ELM is placed in each subdomain and, using (2.4), subdomain network functions denoted by $u_j(\mathbf{x}, \boldsymbol{\theta}_j)$ are substituted in (2.3), such that the global ELM-FBPINN solution, $\hat{u}(\mathbf{x}, \mathbf{a}) \in \mathbb{R}^{d_{out}}$ is given by

$$\hat{u}(\mathbf{x}, \mathbf{a}) = \sum_{j=1}^{J} \omega_j(\mathbf{x}) \sum_{c=1}^{C} a_{j,c}\, \sigma(\mathbf{w}_{j,c} \cdot \mathbf{x} + b_{j,c}) \ . \tag{3.1}$$

In the ELM-FBPINN, the hidden-layer weights $\mathbf{w}_{j,c}$ and biases $b_{j,c}$ are randomly initialised and frozen, while only the output coefficients $a_{j,c}$ are determined by solving a least-squares problem. This removes the need for backpropagation and iterative optimisation, which are the main computational bottlenecks in (FB)PINNs. For simplicity, we use the same basis (activation) functions σ for each ELM. Note, the output dimension d_{out} can be arbitrary, but in this thesis, $d_{out} = 1$ for all tests.

Similar to (FB)PINNs, we use the following loss function to train the ELM-FBPINN:

$$\mathcal{L}(\mathbf{a}) = \lambda_I \sum_{i=1}^{N_I} \big(\mathcal{N}[\hat{u}(\mathbf{x}_i, \mathbf{a})] - f(\mathbf{x}_i)\big)^2 + \lambda_B \sum_{k=1}^{K} \sum_{i=1}^{N_B^{(k)}} \big(\mathcal{B}_k[\hat{u}(\mathbf{x}_i^{(k)}, \mathbf{a})] - g_k(\mathbf{x}_i^{(k)})\big)^2, \tag{3.2}$$

where $\{\mathbf{x}_i\}$ and $\{\mathbf{x}_i^{(k)}\}$ are as defined for the PINN loss function (2.2). The scaling weights λ_I and λ_B are chosen to ensure a good balance between the residual and

boundary terms.

Now, assuming that $\mathcal{N}$ and $\mathcal{B}$ are *linear* operators, we can substitute (3.1) into (3.2) and rearrange to obtain the least-squares problem

$$\min_{\mathbf{a}\in\mathbb{R}^{CJ}} \lambda_I \|\mathbf{M}\mathbf{a}-\mathbf{c}\|_2^2 + \lambda_B \|\mathbf{B}\mathbf{a}-\mathbf{g}\|_2^2. \tag{3.3}$$

This linearity assumption is satisfied for a broad class of elliptic and time-harmonic PDEs and is crucial for enabling an efficient ELM-based formulation. If the operators are non-linear then further considerations must be made, particularly through the introduction of an iterative solver, such as a Newton's method.

It is now convenient to introduce the index $\ell = (j-1)C + c$ to map from index pairs (j, c) in (3.2) to a single index, and $p = \sum_{j=1}^{k-1} N_B^{(j)} + i$ to map from pairs (i, k) in (3.2) to a single index. Additionally, we let $N_B = \sum_{k=1}^{K} N_B^{(k)}$ and define $\Psi_{j,c}(\mathbf{x}) = \sigma(\mathbf{w}_{j,c}\cdot\mathbf{x} + b_{j,c})$.

The vector $\mathbf{c} = \{f(\mathbf{x}_n)\}_{n=1}^{N_I}$ contains evaluations of the PDE right-hand side, f, at internal collocation points, while $\mathbf{g} = \{g_p\}_{p=1}^{N_B}$, $g_p = g_k(\mathbf{x}_i^{(k)})$, contains evaluations of the boundary condition right-hand sides at boundary collocation points. The vector $\mathbf{a} = \{a_\ell\}_{\ell=1}^{JC}$, $a_\ell = a_{j,c}$, contains the output-layer weights of all subdomain ELMs.

The matrices $\mathbf{M}$ and $\mathbf{B}$ are defined as $\mathbf{M} = \{m_{n,\ell}\}$, with $m_{n,\ell} = \mathcal{N}[(\omega_j\Psi_{j,c})(\mathbf{x}_n)]$, and $\mathbf{B} = \{b_{p,\ell}\}$, with $b_{p,\ell} = \mathcal{B}_k[(\omega_j\Psi_{j,c})(\mathbf{x}_i^{(k)})]$, for $\ell = 1,\ldots,JC$, $n = 1,\ldots,N_I$, and $p = 1,\ldots,N_B$.

After solving this least-squares problem, the solution can be evaluated at a set of test points $\{\mathbf{x}_{\text{test}}^q\}_{q=1}^{N_T}$ by computing $\mathbf{u}_{\text{test}} = \mathbf{M}_{\text{sol}}\mathbf{a}$, where $\mathbf{M}_{\text{sol}} = \{m_{q,\ell}\}$ and $m_{q,\ell} = (\omega_j\Psi_{j,c})(\mathbf{x}_{\text{test}}^q)$.

Using this linear system formulation in (3.3), the same ELM algorithm in Algorithm 1 can be applied, solving (3.3) instead of (2.6). An important consequence of the window functions is that each basis function has compact support, which leads to sparse matrices $\mathbf{M}$ and $\mathbf{B}$. This sparsity can be exploited using efficient storage schemes and linear solvers. Similarly to (2.3), the global solution can be constrained to satisfy specific boundary conditions *a priori*, further simplifying the linear system

by eliminating the boundary terms.

This formulation, referred to as an ELM-FBPINN, leverages the advantages of FBPINNs–namely their approximation capabilities and mitigation of spectral bias through domain localisation–while dramatically reducing convergence time compared to gradient-based optimisation. The resulting method combines random feature approximation with domain decomposition in a principled manner. The general form of the algorithm is similar to the Random Feature Method (RFM) [63] without a coarse correction, although details such as the partition of unity, window functions, and sampling strategies differ. Our aim is to show that ELM-FBPINNs maintain comparable solution accuracy while achieving training times orders of magnitude faster than FBPINNs, and that both the ELM formulation and localisation through domain decomposition lead to improved spectral properties of the matrices arising in the least-squares problem.

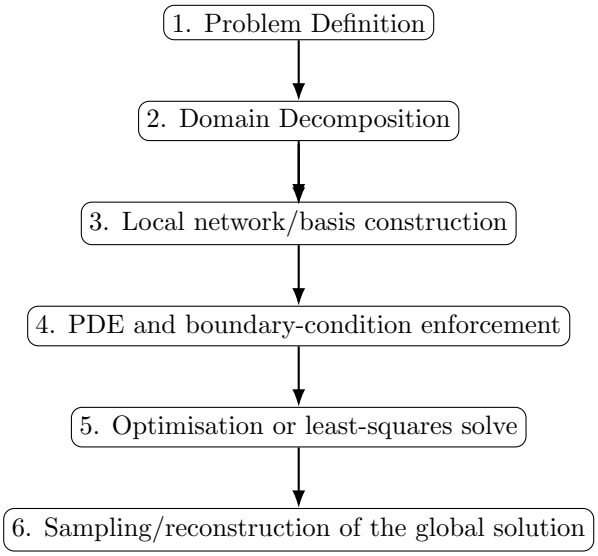


Figure 3.1: Generalised workflow of PINN like methods with domain decomposition. PINN, FBPINN, and ELM-FBPINN all follow this basic structure, with Step 2. only being relevant for FBPINN and ELM-FBPINN.

3.3 Problems studied

To investigate the potential of the ELM-FBPINN, the model is compared to a PINN and an FBPINN. This comparison allows us to separate the effects of the underlying

network architecture from those of the training strategy. In particular, it enables a direct comparison between gradient-descent-based solvers and linear least-squares solves within otherwise similar approximation frameworks. We make the decision to not compare ELM-FBPINN against similar architectures (e.g. DPIELM), as the focus of the study is the direct effect that linearisation of the optimisation problem has on the FBPINN formulation. Once a better understanding of the effects of the introducing ELM to FBPINN has been achieved, direct comparison to traditional methods, and similar neural network methods would be a useful direction. Two problems are studied: a one-dimensional damped harmonic oscillator and a two-dimensional multiscale Laplace problem. These test cases are deliberately chosen to probe different sources of difficulty commonly encountered by physics-informed models, such as high-frequency content and multiscale structure. Both problems are described in detail before a range of ablation studies are presented and discussed. These studies are designed to assess scalability, efficiency, stability, and solution accuracy of the models.

3.3.1 1D problem – Damped harmonic oscillator

The one-dimensional damped harmonic oscillator is chosen due to the multiscale, high-frequency features present in its solution. Such features are well known to challenge PINNs due to spectral bias and slow convergence of gradient-based optimisation. In contrast, with a sufficient number of subdomains, an FBPINN is expected to approximate the solution more efficiently through localisation. The displacement of the oscillator as a function of time, $u(t)$, is modelled by the following second-order ordinary differential equation:

$$\begin{cases} mu_{tt} + \mu u_t + ku(t) = 0, \quad t \in (0, 1], \\ u(0) = u_0, \quad u_t(0) = v_0, \end{cases} \tag{3.4}$$

where m is the mass of the oscillator, μ is the damping (friction) coefficient, and k is the spring constant. The initial conditions u_0 and v_0 represent the initial displacement and velocity, respectively.

The exact solution of the harmonic oscillator depends on the damping regime. In

the overdamped case, the oscillator slowly returns to equilibrium, while in the critically damped case it does so as rapidly as possible without oscillation. Here, we focus on the under-damped regime, which exhibits oscillatory behaviour combined with exponential decay and is therefore particularly challenging for neural network approximations. This regime occurs when

$$\delta < \omega_0, \quad \text{where} \quad \delta = \frac{\mu}{2m} \quad \text{and} \quad \omega_0 = \sqrt{\frac{k}{m}}. \tag{3.5}$$

We fix $m = 1$ and $\mu = 4$, which gives $\delta = 2$. By varying ω_0, the oscillation frequency can be controlled. Increasing ω_0 introduces increasingly high-frequency components into the solution while preserving the same decay rate. With these parameters, the exact solution in the under-damped regime is given by

$$u_{\text{exact}}(t) = e^{-\delta t}\, 2A\, \cos(\phi + \omega t), \tag{3.6}$$

where

$$\omega = \sqrt{\omega_0^2 - \delta^2}, \quad A = \frac{u_0}{2\cos(\phi)}, \quad \phi = \arctan\Big(- \frac{(v_0/u_0) + \delta}{\omega} \Big).$$

The parameters A and ϕ correspond to the amplitude and phase of the oscillation.

All models minimise the same loss function,

$$\mathcal{L} = \mathcal{L}_{\text{phys}} + \lambda_{u_0}\, \mathcal{L}_{\text{bc}}^{(u_0)} + \lambda_{v_0}\, \mathcal{L}_{\text{bc}}^{(v_0)}, \tag{3.7}$$

where

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_f} \sum_{i=1}^{N_f} \Big(\hat{u}_{tt}(\mathbf{t}) + 4\hat{u}_t(\mathbf{t}) + k\hat{u}(\mathbf{t}) \Big)^2, \tag{3.8}$$

$$\mathcal{L}_{\text{bc}}^{(u_0)} = \frac{1}{N_b} \sum_{j=1}^{N_b} \big(\hat{u}(0) - u_0 \big)^2, \tag{3.9}$$

$$\mathcal{L}_{\text{bc}}^{(v_0)} = \frac{1}{N_b} \sum_{j=1}^{N_b} \big(\hat{u}_t(0) - v_0 \big)^2. \tag{3.10}$$

The scaling parameters λ_{u_0} and λ_{v_0} are fixed at 10^6 and 10^2, respectively, to ensure strong enforcement of the initial conditions. Note, we have chosen to include the boundary conditions within the loss function for the harmonic oscillator, rather than introduce hard constraints. This is primarily to observe the effect, if any, additional loss terms have on the training dynamics of the models.

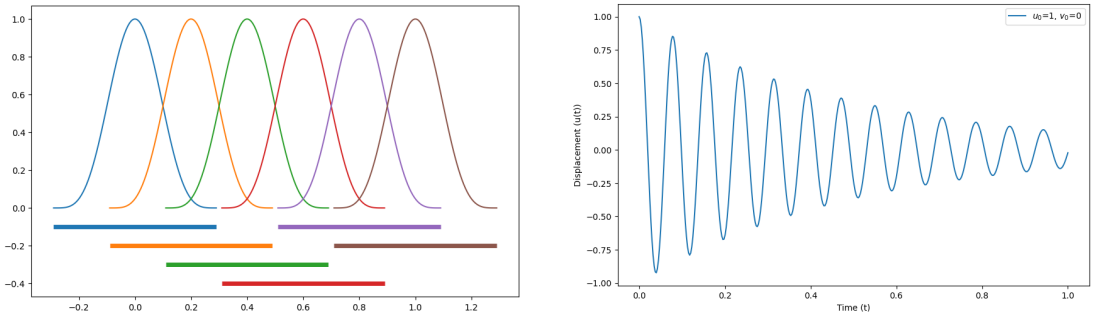


Figure 3.2: Left: Example domain decomposition with 6 subdomains and width factor, $\eta = 2.9$. Coloured bars indicate the support of each window function. Right: Exact solution of the damped harmonic oscillator for $\omega_0 = 80$, $u_0 = 1$, and $v_0 = 0$.

For the FBPINN and ELM-FBPINN, the global time interval is divided into J equally spaced subdomains, with widths chosen such that neighbouring subdomains overlap moderately.

This overlap is essential to guarantee rapid transmission of information between subdomains while preserving the locality of each subdomain approximation. An example domain decomposition and corresponding exact solution are shown in Figure 3.2.

3.3.2 Multiscale Laplace in 2D

The second problem studied is a variation of a multiscale Laplace problem in two dimensions, as introduced in [58]. This test case is designed to assess the ability of the models to capture increasingly complex multiscale structure in higher dimensions. The

system is governed by the second-order PDE with Dirichlet boundary conditions:

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{3.11}$$

where the source term $f(x_1, x_2)$ is given by

$$f(x_1, x_2) = \frac{2}{n_\omega} \sum_{i=1}^{n_\omega} (\omega_i \pi)^2 \sin(\omega_i \pi x_1) \sin(\omega_i \pi x_2). \tag{3.12}$$

The corresponding exact solution is

$$u(x_1, x_2) = \frac{1}{n_\omega} \sum_{i=1}^{n_\omega} \sin(\omega_i \pi x_1) \sin(\omega_i \pi x_2). \tag{3.13}$$

The problem is posed on the unit square $\Omega = [0, 1]^2$ which is divided into J^2 rectangular subdomains, with J representing the per axis number of subdomains. An example of the two-dimensional domain decomposition, along with a representative exact solution, is shown in Figure 3.3. The general neural network architecture, consisting of h hidden layers with C basis functions per layer, remains unchanged across models. However, due to the increased dimensionality and multiscale nature of the solution, a larger number of basis functions is required to achieve sufficient expressivity compared to the one-dimensional case. In this study, the frequency components are fixed as $\omega_i = 4i$, while the number of multiscale components, n_ω, is varied to control problem difficulty. This linear frequency scaling enables a gradual increase in complexity while introducing progressively higher-frequency modes into the solution.

Similarly to (2.3), a constraining function is applied to the neural network output to enforce the Dirichlet boundary conditions exactly:

$$\mathcal{C}[\hat{u}(x_1, x_2)] = \tanh\left(\frac{x_1}{\sigma}\right) \tanh\left(\frac{1 - x_1}{\sigma}\right) \tanh\left(\frac{x_2}{\sigma}\right) \tanh\left(\frac{1 - x_2}{\sigma}\right) \hat{u}(x_1, x_2), \tag{3.14}$$

where $\sigma = 0.2$. This multiplicative mask smoothly enforces the homogeneous Dirichlet boundary conditions and removes the need for explicit boundary loss terms. As a result,

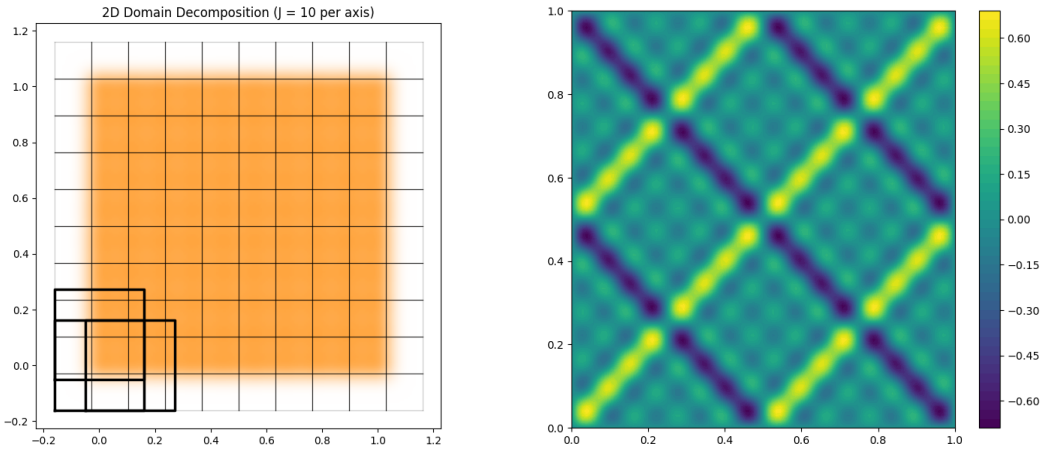


Figure 3.3: Left: Example two-dimensional domain decomposition with 10×10 subdomains and width factor of $\eta = 2.9$. Three window functions are highlighted in the lower-left region, while the orange colouring indicates the summed partition of unity. Right: Example exact solution of the multiscale Laplace problem with $n_\omega = 4$ and $\omega_4 = 16$.

the loss function reduces to a purely interior residual:

$$\mathcal{L} = \frac{1}{N} \sum_{k=1}^{N} \left[-(\mathcal{C}[\hat{u}])_{xx}(x_{1,k}, x_{2,k}) - (\mathcal{C}[\hat{u}])_{yy}(x_{1,k}, x_{2,k}) - f(x_{1,k}, x_{2,k}) \right]^2. \tag{3.15}$$

3.3.3 Common implementation details

Several aspects of the training and testing setup are shared across both the one- and two-dimensional problems. These include the window functions, domain decomposition strategy, and general neural network architectures. Unless explicitly stated otherwise, these settings remain fixed across all experiments to ensure a fair comparison between models.

Both the FBPINN and ELM-FBPINN utilise the same domain decomposition strategy. In one and two dimensions, we assume that the computational domain has a tensor product structure, i.e. that $\Omega = \bigotimes_{i=1}^{d} [x^i_{\min}, x^i_{\max}]$. Then, we can partition Ω into equally sized intervals (1D) or rectangular subdomains (2D) arranged on a uniform grid. Each subdomain overlaps with its immediate neighbours according to a prescribed width factor, which controls the extent of information exchange between local models.

The partition-of-unity (POU) and window functions used to localise each subdomain network are defined as

$$\omega_j(\mathbf{x}) = \frac{\hat{\omega}_j(\mathbf{x})}{\sum_k \hat{\omega}_k(\mathbf{x})}, \quad \hat{\omega}_j(\mathbf{x}) = \prod_{i=1}^{d} \left[1 + \cos\Big(\pi \frac{x_i - \mu_{j_i}}{\sigma_{j_i}} \Big) \right]^2, \tag{3.16}$$

where μ_{j_i} denotes the centre of the j^{th} subdomain in dimension i, and σ_{j_i} defines its half-width. The subdomain centres $\{\mu_{j_i}\}$ are prescribed a priori (e.g. on a uniform grid). The half-width of each subdomain is then defined in terms of a width factor, η, as

$$\sigma_{j_i} = \frac{\eta}{2} h_i, \quad h_i = \mu_{j_{i+1}} - \mu_{j_i}. \tag{3.17}$$

The corresponding subdomain bounds are given by

$$x^j_{\min} = \mu_{j_i} - \sigma_{j_i}, \quad x^j_{\max} = \mu_{j_i} + \sigma_{j_i}. \tag{3.18}$$

The width factor η controls the support size of each window function. When $\eta = 1$, subdomains meet but do not overlap, while $\eta > 1$ introduces increasing overlap between neighbouring subdomains. Figure 3.4 illustrates example 1D configurations: $\eta = 1$ yields no overlap, $\eta = 1.5$ produces mild overlap, and $\eta = 2.9$ results in substantial overlap. In this work, we adopt $\eta = 2.9$ for all experiments to ensure sufficient interaction between neighbouring subdomains.

All models use fully connected neural networks with tanh activation functions. For the ELM-FBPINN, the linear output weights, $\mathbf{a}$, are initialised independently for each subdomain, while the hidden-layer parameters are shared and fixed across all subdomains. This differs from the FBPINN, where a full network is initialised and trained independently on each subdomain. The intuition behind this design is to retain localisation through the output layer while reducing the total number of trainable parameters and enabling a global linear solve. Additionally, due to the random initialisation of the network weights, there should be an insignificant affect on model performance.

The PINN and FBPINN are trained using the Adam optimiser with a learning rate of 10^{-3} for 15,000 iterations. The ELM-FBPINN employs an iterative LSQR sparse

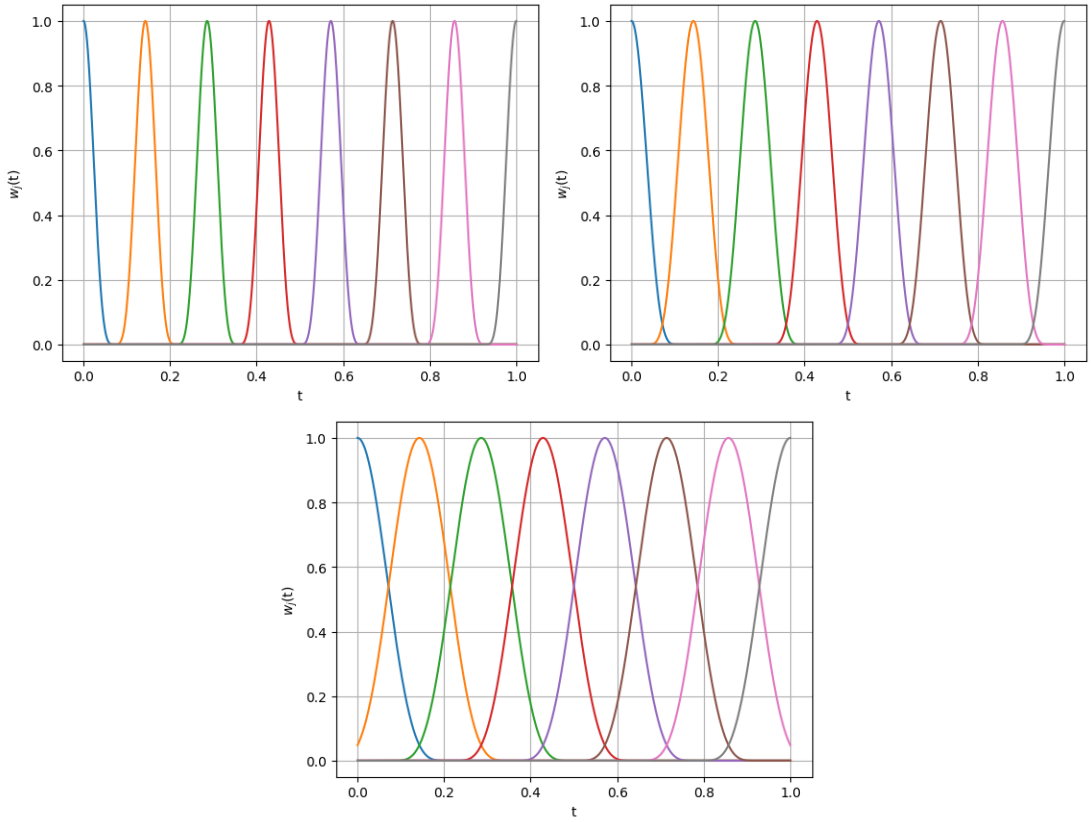


Figure 3.4: Example one-dimensional decompositions for varying width factors. Top Left: $\eta = 1$. Top Right: $\eta = 1.5$. Bottom: $\eta = 2.9$

solver from SciPy [64], with a maximum of 15,000 iterations. Unless stated otherwise, the solver is allowed to run for the full iteration budget to enable a direct comparison of convergence behaviour. All networks are initialised using uniform LeCun initialisation [65], with the exception that the hidden-layer weights in the ELM-FBPINN are frozen.

Model performance is evaluated using the L^1 error,

$$e_{L^1} = \sum_{i=1}^{N} |\hat{u}(\mathbf{x}_i) - u(\mathbf{x}_i)|, \tag{3.19}$$

where $N = N_I + N_B$ denotes the total number of collocation points. Training and test points are sampled uniformly, with sufficient resolution per subdomain to accurately assess local approximation quality. Each experiment is repeated using six different random seeds. Reported errors correspond to the mean L^1 error, denoted as $\hat{e}_{L^1}$, with

error bars defined by the difference between the best and worst observed values.

All models are implemented in Python using JAX [10]. The ELM-FBPINN implementation builds upon the open-source FBPINN codebase available at `github.com/benmoseley/FBPINNs`. Experiments are run on either an AMD Ryzen 5 7600X CPU with 32 GB of RAM or an NVIDIA RTX 5070 Ti GPU with 16 GB of VRAM. Due to current incompatibilities between SciPy and JAX, the linear systems are solved on the CPU; however, a JAX-native LSQR implementation would allow the ELM-FBPINN to run fully on the GPU.

3.4 Numerical Results

In this section, numerical results are presented to evaluate the performance of the ELM-FBPINN and to compare it against the PINN and FBPINN. The focus is on convergence behaviour, solution accuracy, and computational efficiency. Sections 3.4.1 and 3.4.2 present results obtained for the damped harmonic oscillator and the multi-scale Laplace problem, respectively. Each section begins with a baseline configuration, followed by a series of ablation studies. These ablations are designed to isolate the effects of frequency content, network capacity, and domain decomposition parameters. For the one-dimensional problem, additional analysis is performed on the shape of the hidden-layer basis functions used by the ELM-FBPINN. The section concludes with a brief discussion summarising the key findings.

3.4.1 1D Damped harmonic oscillator

Baseline

We begin with a baseline experiment to establish a direct comparison between the performance of the three models. For all experiments in this subsection, we set $\omega_0 = 80$, $u_0 = 1$, and $v_0 = 0$, corresponding to the exact solution shown in Figure 3.2 (right). This configuration produces a highly oscillatory solution with slowly decaying amplitude, which is known to be challenging for PINNs.

The PINN is initialised with $h = 2$ hidden layers and $C = 64$ basis functions

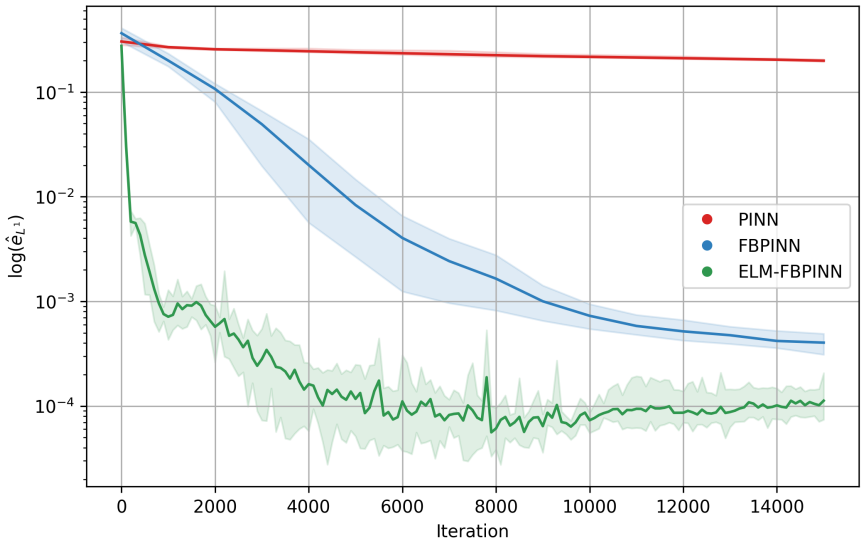


Figure 3.5: Logarithm of the $\hat{e}_{L^1}$ test error versus iteration count for the baseline 1D damped harmonic oscillator.

per layer. Both the FBPINN and ELM-FBPINN use a single hidden layer ($h = 1$) with $C = 12$ basis functions per subdomain. The domain is decomposed into $J = 20$ subdomains, with a fixed width factor of $\eta = 2.9$ for both FBPINN-based models.

All models are trained using $(C+1)J$ collocation points. The PINN is deliberately trained on the same number of points as the FBPINN and ELM-FBPINN to ensure a fair comparison in terms of data availability. The sampling strategy is chosen to guarantee sufficient coverage within each subdomain for the FBPINN and ELM-FBPINN, and is more than sufficient for the single-domain PINN. The number of training points also ensures that the ELM-FBPINN system matrix $\mathbf{M}$ is overdetermined, guaranteeing a unique least-squares solution.

Figure 3.5 shows the evolution of the L^1 test error (log scale) as a function of iteration count. As expected, the PINN fails to converge to an acceptable solution within the allotted training budget. This behaviour is consistent with spectral bias, which limits the ability of gradient-based training to resolve high-frequency components. Both the FBPINN and ELM-FBPINN converge successfully. Notably, the ELM-FBPINN achieves the lowest final error and converges significantly faster than both the PINN and the FBPINN.

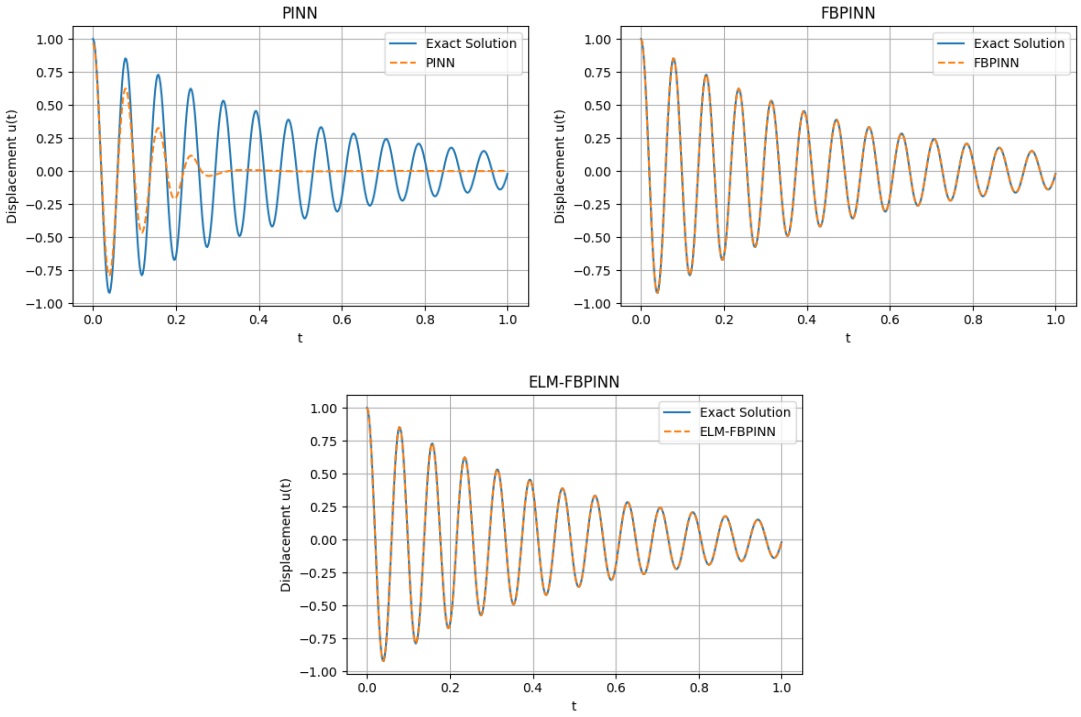


Figure 3.6: Model predictions versus the exact solution for the 1D damped harmonic oscillator. Top left: PINN; top right: FBPINN; bottom: ELM-FBPINN.

Predictions from each model evaluated on 400 uniformly spaced test points are shown in Figure 3.6. The PINN prediction fails to capture both the amplitude and frequency of the oscillations, and its slow convergence is attributed to spectral bias. Both the FBPINN and ELM-FBPINN accurately recover the solution and are visually indistinguishable. Quantitative differences between these two models are therefore best assessed through convergence speed and final error metrics.

These metrics are summarised in Table 3.1. Despite achieving comparable $\hat{e}_{L^1}$ errors, the ELM-FBPINN, averaged across the seeds, converges approximately 3.6 s faster than the FBPINN. It should be noted that the reported FBPINN timing includes an initial JIT compilation overhead of approximately 1–2 s. While this means the true training time is slightly overestimated, the relative performance difference remains of the same order of magnitude.

In practical settings, the ELM-FBPINN would also benefit from early stopping of the LSQR solver. For this baseline experiment, convergence is effectively reached after

approximately 8,000 iterations, suggesting that the elapsed time could be reduced by nearly a factor of two without degrading accuracy.

Model	$\hat{e}_{L^1}$	$\hat{e}_{L^2}$	$\hat{e}_{L^\infty}$	Time (s)
PINN	(1.99±0.090)e-01	(2.41±0.094)e-01	(2.48±0.097)e-01	12.4±0.228
FBPINN	(4.22±0.163)e-04	(4.93±0.190)e-04	(5.70±0.190)e-04	14.8±0.063
ELM-FBPINN	(1.13±0.132)e-04	(1.35±0.149)e-04	(2.38±0.150)e-04	12.2±0.0558

Table 3.1: Baseline results for the 1D damped harmonic oscillator. Reported $\hat{e}_{L^1}$, $\hat{e}_{L^2}$ and $\hat{e}_{L^\infty}$ values correspond to mean ± range over random seeds, with scientific notation factorized for readability. Time values are mean ± 1 standard deviation.

For the ELM-FBPINN, the (2-norm) condition number $\kappa(\mathbf{M})$ and numerical rank of the system matrix $\mathbf{M}$ are computed using the singular value decomposition (SVD). These quantities provide insight into the stability and numerical robustness of the linear system. The results are reported in Table 3.2 and serve as reference values for subsequent ablation studies.

Model	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)
ELM-FBPINN	1.30e+18	216

Table 3.2: Condition number $\kappa(\mathbf{M})$ and numerical rank of the ELM-FBPINN system matrix for the baseline configuration.

Varying Number of Basis Functions

With a baseline established, we now move on to a set of ablation studies. Specifically, we first examine the effect of varying the number of basis functions, C, in each model. This allows us to evaluate how each approach behaves under reduced model capacity at low values of C, and to assess how convergence properties evolve as the network size increases. In particular, this study highlights the extent to which each method can compensate for limited representational capacity through its architectural design or optimisation strategy.

For these experiments, the PINN retains $h = 2$ hidden layers, while the number of basis functions C is varied in line with the FBPINN and ELM-FBPINN models to ensure a fair comparison. The convergence curves for all models are shown in Figure 3.7.

Here, all PINN models fail to converge, which is not unexpected, as the largest number of basis functions considered here ($C = 64$)—which should in principle provide the greatest expressivity and hence the best achievable performance for this architecture—corresponds to the baseline model for which convergence was not achieved (cf. Figure 3.5). This further reinforces the difficulty of training standard PINNs for highly oscillatory problems, even when the nominal network capacity is increased.

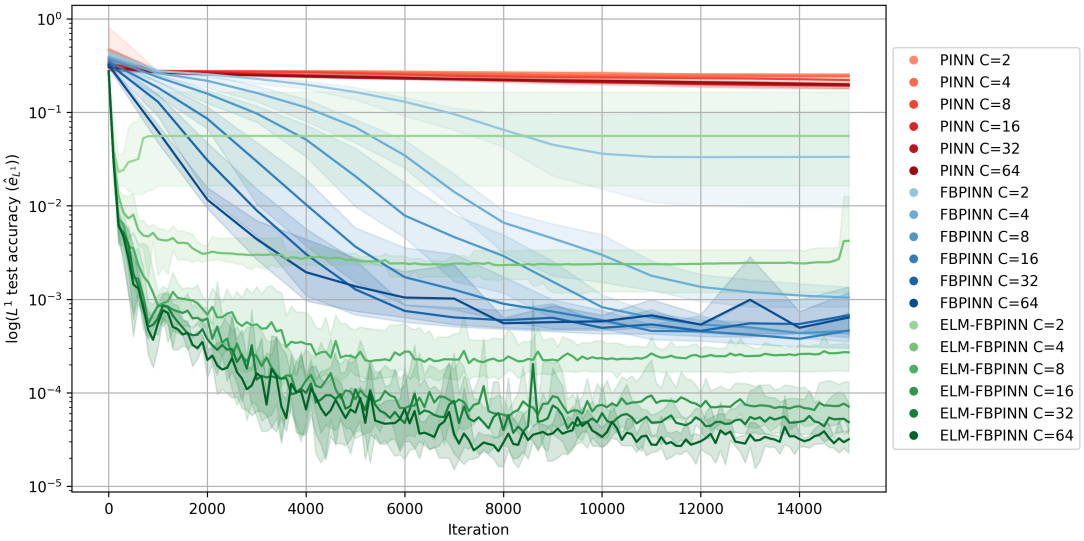


Figure 3.7: Logarithm of the $\hat{e}_{L^1}$ error vs iterations for the 1D damped harmonic oscillator for different numbers of basis functions, C.

Both the FBPINN and ELM-FBPINN exhibit significantly improved convergence behaviour. The ELM-FBPINN is the only model to show a clear and consistent improvement in the test error, $\hat{e}_{L^1}$, as C increases, while also converging at a markedly faster rate in all tested configurations. This suggests that the linear solve inherent to the ELM-FBPINN formulation enables more effective utilisation of additional basis functions, compared to gradient-based optimisation.

At the lowest number of basis functions, FBPINN performs best, with ELM-FBPINN failing to fully converge until the network capacity reaches at least $C > 4$. This behaviour is expected, as the columns of the ELM-FBPINN system matrix $\mathbf{M}$ are directly tied to the number of basis functions and subdomains. Without sufficient matrix capacity, the model struggles to generalise from the $(C+1)J$ training points to the $50 \times J$

test points. In this low-capacity regime, the expressivity bottleneck dominates over the advantages of the linear solver, limiting achievable accuracy.

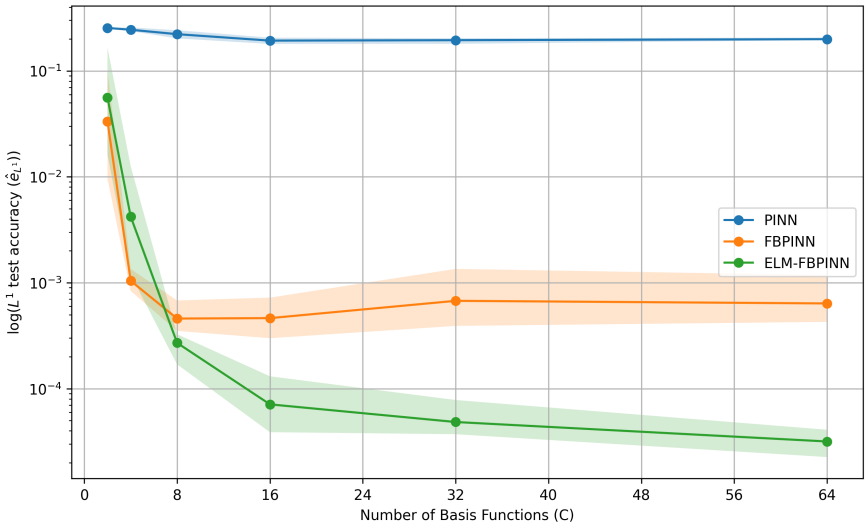


Figure 3.8: Final $\hat{e}_{L^1}$ values for each model vs the number of basis functions, C.

The final $\hat{e}_{L^1}$ values plotted against C are shown in Figure 3.8, with the corresponding numerical values and average run times reported in Table 3.3. While the performance of the PINN and FBPINN remains relatively flat as C increases, the ELM-FBPINN exhibits a sharp improvement in accuracy before plateauing at larger values of C. This saturation behaviour suggests that beyond a certain capacity, additional basis functions do not contribute meaningful new information for this problem.

C	PINN	FBPINN	ELM-FBPINN
2	(2.59±0.0396)e-01	(1.25±0.677)e-03	(5.6±14.9)e-02
4	(2.41±0.132)e-01	(5.89±2.47)e-04	(4.21±11.4)e-03
8	(2.23±0.291)e-01	(4.56±2.13)e-04	(2.7±1.57)e-04
16	(1.93±0.266)e-01	(4.14±2.99)e-04	(7.11±9.19)e-05
32	(1.94±0.204)e-01	(5.58±8.53)e-04	(4.85±4.1)e-05
64	(2±0.137)e-01	(5.6±6.51)e-04	(3.17±1.83)e-05

Table 3.3: Final $\hat{e}_{L^1}$ values for each model at different number of basis functions, C.

The condition number and rank of the ELM-FBPINN system matrices are shown in Figure 3.9. As C increases, both the condition number and the matrix rank increase before plateauing at larger values. Since the number of columns of $\mathbf{M}$ depends directly

on C, this behaviour is expected. In particular, the rapid growth of the condition number reflects the increasing presence of near-linear dependencies among the basis functions.

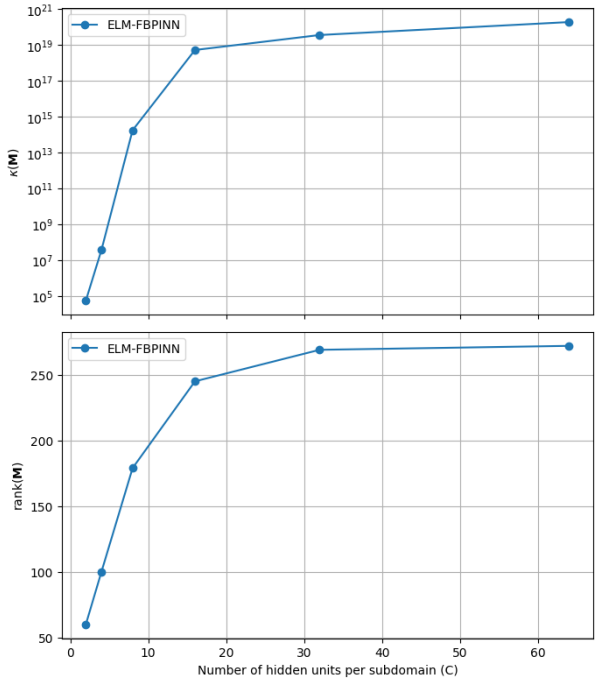


Figure 3.9: Top: Condition number, $\kappa(\mathbf{M})$ and Bottom: rank($\mathbf{M}$) vs number of basis functions, C.

C	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)
2	5.50e+04	60
4	3.61e+07	100
8	1.73e+14	179
12	**1.30e+18**	216
16	5.10e+18	245
32	3.49e+19	269
64	1.84e+20	272

Table 3.4: SVD-based condition number and matrix rank for ELM-FBPINN as a function of C.

The near-linear dependence of the basis functions, and hence columns of $\mathbf{M}$, also leads to a saturation in rank as more basis functions are added. This upper limit depends on several factors, including the number and width of the subdomains, the choice of initialisation of the hidden weights and biases, and the choice of activation functions. Figure 3.10 illustrates the basis functions within a single subdomain for $C = 8$ and $C = 32$. By considering possible linear combinations of these functions, it becomes evident that higher values of C lead to a greater likelihood of redundancy, thereby explaining the observed plateau in rank and accuracy improvements. These observations motivate the use of a pre-training stage, which is explored in Chapter 4, to introduce more non-linearity in to the hidden-layer basis, mitigating the redundant basis functions. However, we stress that, despite the ill-conditioning of $\mathbf{M}$, the least-

squares problems are evidently solved well enough to obtain accurate ELM-FBPINN solutions. This suggests that the full least-squares problems (3.3) involving $\mathbf{M}$, $\mathbf{B}$, $\mathbf{c}$ and $\mathbf{g}$, are not as badly ill-conditioned as $\kappa(\mathbf{M})$ indicates.

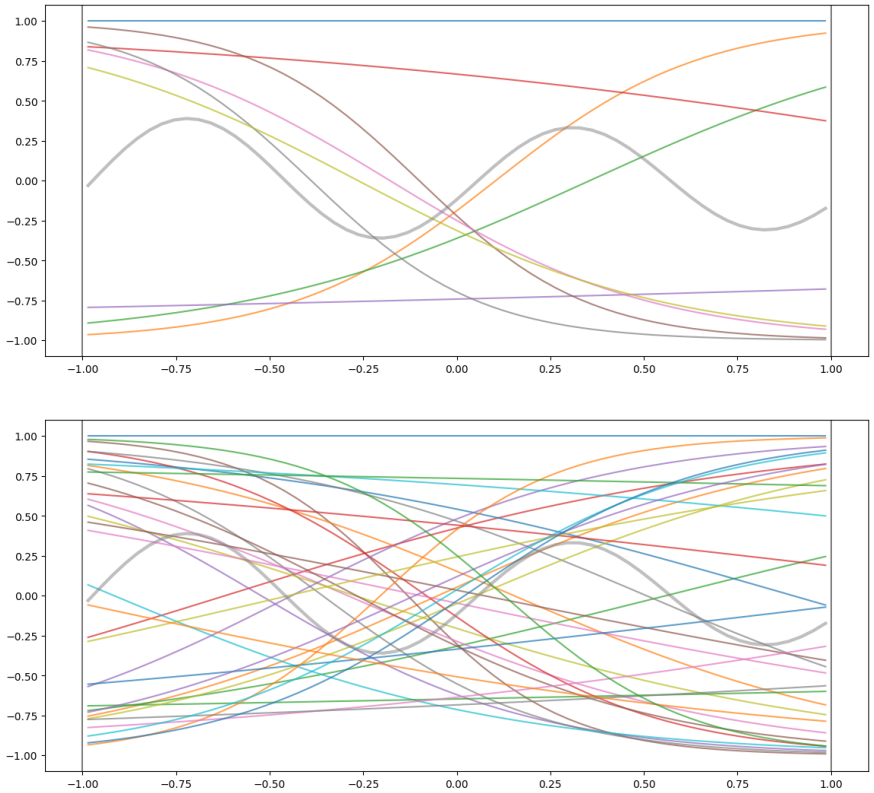


Figure 3.10: Basis functions of ELM-FBPINN, within a single subdomain. Top: $C = 8$ Bottom: $C = 32$.

Overall, we see that, except for very low numbers of basis functions ($C = 2, 4$), the ELM-FBPINN method outperforms both the PINN and FBPINN approaches in terms of accuracy, and that the accuracy obtained is insensitive to the number of basis functions used. Moreover, these solutions are obtained without the potentially large expense of gradient-based optimisation. Although the least-squares problems solved within ELM-FBPINN involve matrices whose condition number grows with C, this does not seem to negatively affect the obtained approximations to the harmonic oscillator solution.

Varying Number of Subdomains

We next investigate the effect of varying the number of subdomains, J. As the number of subdomains increases, it is expected that the solution accuracy will improve, since each subnetwork is required to approximate the solution over a smaller spatial region. This domain decomposition is particularly important for highly oscillatory problems, where global neural networks are known to suffer from spectral bias.

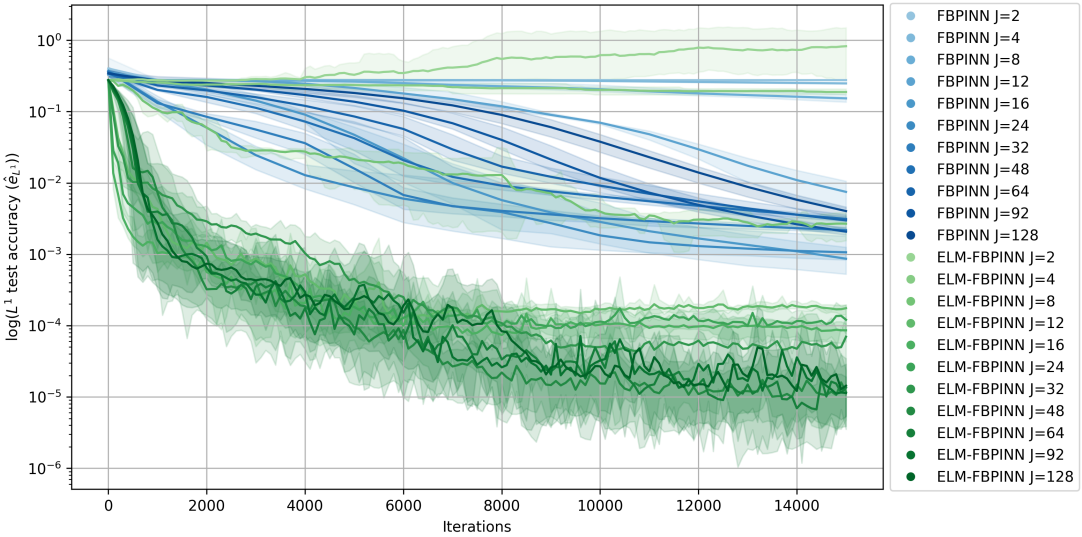


Figure 3.11: Convergence curves for varying number of subdomains, J, for FBPINN and ELM-FBPINN.

For these experiments, all model parameters are fixed at their baseline values, and only J is varied. As the PINN does not decompose the global domain into subdomains, only the FBPINN and ELM-FBPINN models are considered. The convergence curves for both models are shown in Figure 3.11. In general, the test error $\hat{e}_{L^1}$ improves as the number of subdomains increases for both methods, with the ELM-FBPINN outperforming the FBPINN in almost all cases. This improvement is accompanied by a clear acceleration in convergence, particularly for moderate to large values of J.

For $J > 8$, the ELM-FBPINN converges notably faster than the FBPINN. While the FBPINN $\hat{e}_{L^1}$ could potentially be reduced further with additional training iterations, the shallow slope of the convergence curves indicates slow progress. In contrast, within the same number of iterations, the ELM-FBPINN appears to have nearly con-

verged when a sufficient number of subdomains is used. This highlights the advantage of replacing gradient-based optimisation with a linear solve once a suitable domain decomposition has been established.

J	FBPINN	ELM-FBPINN
2	(2.76±0.00346)e-01	(8.25±12.3)e-01
4	(2.48±0.0577)e-01	(1.88±0.332)e-01
8	(1.53±0.384)e-01	(2.68±2.26)e-03
12	(7.49±6.24)e-03	(1.75±0.525)e-04
16	(8.66±7.3)e-04	(8.58±4.67)e-05
24	(1.07±0.96)e-03	(1.21±1.18)e-04
32	(2.18±0.321)e-03	(6.99±13.1)e-05
48	(3.13±0.521)e-03	(1.17±1.84)e-05
64	(2.98±1.23)e-03	(1.35±3.06)e-05
92	(2.07±0.401)e-03	(1.13±1.56)e-05
128	(4.03±1.52)e-03	(1.44±2.35)e-05

Table 3.5: Final $\hat{e}_{L^1}$ values for each model at different number of basis functions, J.

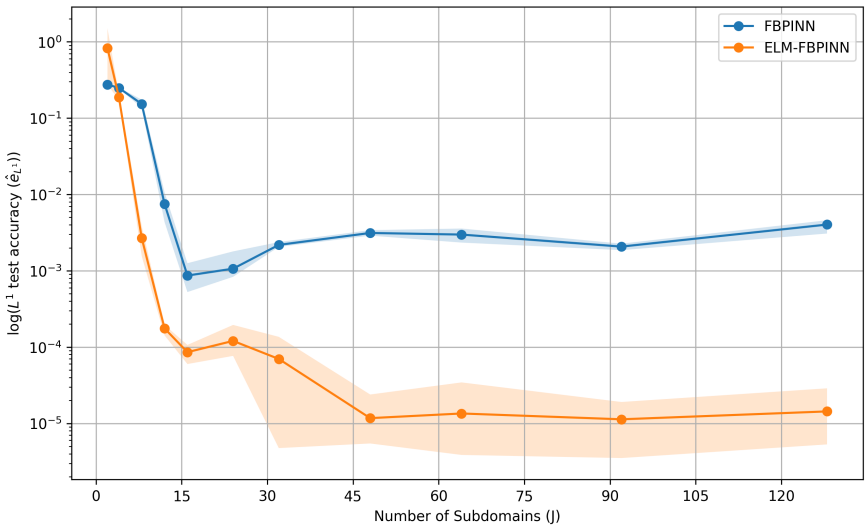


Figure 3.12: Logarithm of final $\hat{e}_{L^1}$ values for varying number of subdomains, J, for FBPINN and ELM-FBPINN.

The final $\hat{e}_{L^1}$ values for each choice of J, together with the corresponding run times, are shown in Table 3.5 and Figure 3.12. It is clear that a minimum number of subdomains is required for both models to overcome spectral bias, with both methods failing to converge when $J = 2$ and $J = 4$. In these cases, each subdomain remains too

large for the local networks to resolve the oscillatory structure of the solution.

At $J = 8$, the ELM-FBPINN is already able to produce an acceptable solution, whereas the FBPINN is not, as illustrated in Figure 3.13. Both models continue to improve in accuracy as J increases, before eventually stagnating. Notably, the ELM-FBPINN continues to improve up to approximately $J \approx 48$, while the FBPINN saturates earlier, around $J \approx 16$. This suggests that the ELM-FBPINN is able to exploit finer domain decompositions more effectively than the gradient-trained FBPINN.

A likely explanation for this stagnation at higher numbers of subdomains is the relative simplicity of the 1D damped harmonic oscillator problem. As J increases, each subdomain problem becomes progressively simpler, eventually reaching a point where further subdivision yields diminishing returns. Although the final $\hat{e}_{L^1}$ is marginally lower at $J = 128$, the result at $J = 16$ is within the same order of magnitude, while requiring a substantially smaller system matrix $\mathbf{M}$. As shown in Figure 3.13, even at $J = 8$, the ELM-FBPINN is able to capture the main oscillatory features of the solution. Beyond a certain threshold in J, further accuracy improvements are therefore more likely to require changes to other model parameters, such as the number of basis functions.

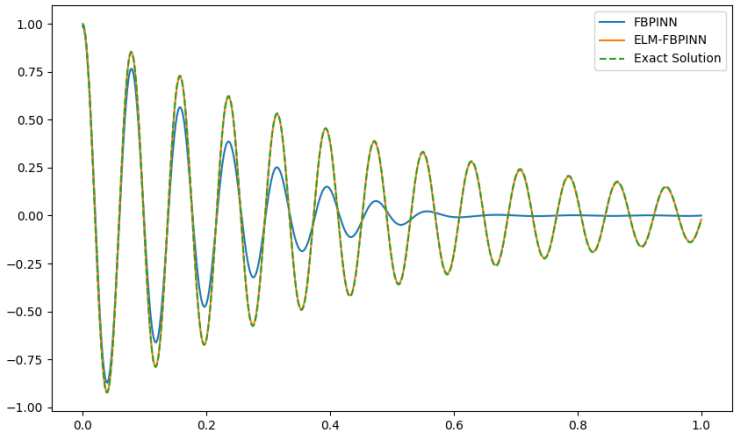


Figure 3.13: Exact solution and model predictions when $J = 8$.

The 2-norm condition number and rank of the ELM-FBPINN system matrix are

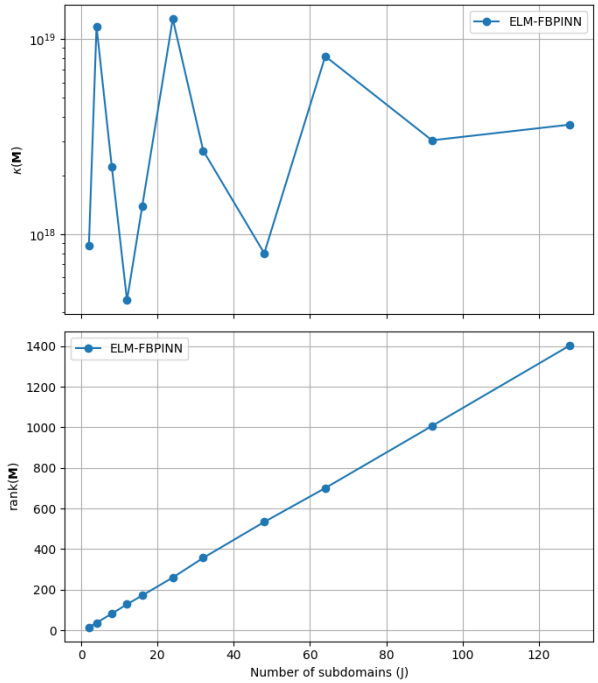


Figure 3.14: Condition number, $\kappa(\mathbf{M})$ and rank($\mathbf{M}$) vs number of subdomains, J.

J	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)
2	8.75e+17	15
4	1.16e+19	37
8	2.22e+18	82
12	4.59e+17	128
16	1.39e+18	172
24	1.27e+19	260
32	2.68e+18	357
48	7.99e+17	534
64	8.17e+18	701
92	3.03e+18	1007
128	3.64e+18	1402

Table 3.6: SVD-based condition number and matrix rank for ELM-FBPINN as a function of J.

shown in Figure 3.14 and detailed in Table 3.6. The condition number remains relatively stable across all tested values of J, lying within the range 10^{17} to 10^{19}. This indicates that increasing the number of subdomains does not significantly exacerbate the ill-conditioning of the linear system.

In contrast, the matrix rank exhibits a quasi-linear increase with J, demonstrating that, for a fixed number of basis functions per subdomain, the additional columns introduced by increasing J are largely linearly independent. This behaviour is desirable, as it implies that increasing the number of subdomains primarily increases the expressive power of the model, rather than introducing redundant degrees of freedom.

In summary, while both the FBPINN and ELM-FBPINN methods benefit from increasing the number of subdomains, J, which makes the problem on each subdomain more linear, the ELM-FBPINN method is better able to capitalise on this, converging more quickly to more accurate solutions. Moreover, the least-squares problems that must be solved in ELM-FBPINN seem largely unaffected by the number of subdomains,

indicating that the optimisation step is not significantly more challenging.

Scaling Solution Frequency With Subdomain Count

Next, we examine the scalability of the method with respect to problem difficulty. To do this, we investigate the effect of increasing the solution frequency, ω_0, while also increasing the number of subdomains J; all remaining parameters of the FBPINN and ELM-FBPINN models are fixed at their baseline values. By coupling the problem frequency with the number of subdomains, the effective resolution per subdomain is kept approximately constant, isolating the impact of the learning and optimisation strategy. Hence, in this setting, both the oscillatory complexity of the target solution and the effective model capacity are increased simultaneously. Ideally, the final values of $\hat{e}_{L^1}$ should remain approximately constant as ω_0 increases, which would indicate that the model is able to scale robustly to more challenging, higher-frequency problems.

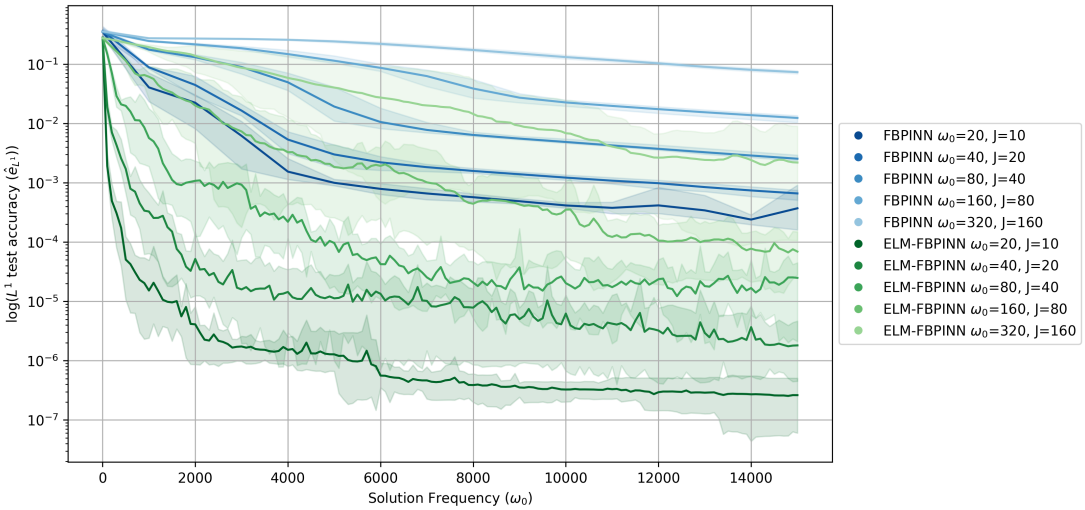


Figure 3.15: Convergence curves for varying number of subdomains, J, and solution frequency, ω_0 for FBPINN and ELM-FBPINN.

The convergence behaviour for both models is shown in Figure 3.15. As expected, both approaches exhibit progressively slower convergence as the solution frequency ω_0 increases. This is unsurprising, as higher values of ω_0 correspond to increasingly oscillatory solutions, which are inherently more difficult to approximate. Even with an increase in J, higher-frequency regimes place greater demands on the expressive power

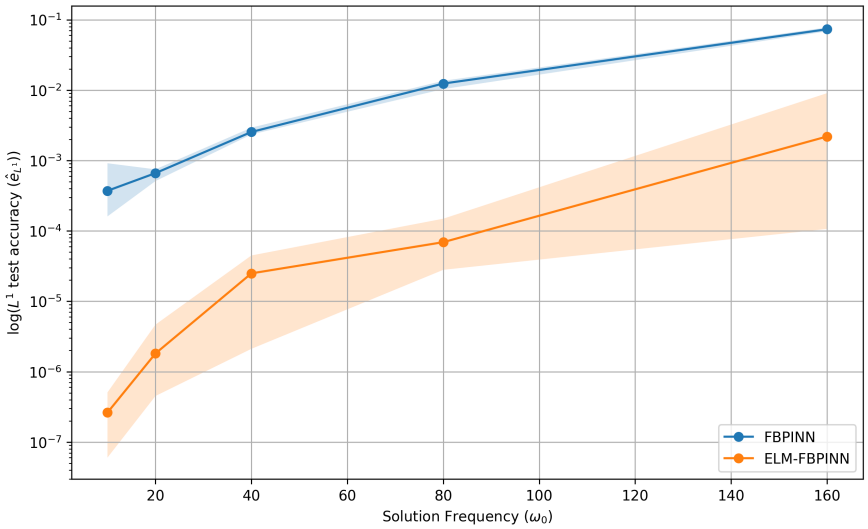


Figure 3.16: Convergence curves for varying number of subdomains, J, and solution frequency, ω_0 for FBPINN and ELM-FBPINN.

and numerical stability of the underlying networks.

Despite this increased difficulty, the ELM-FBPINN consistently outperforms the FBPINN across all tested values of ω_0, both in terms of convergence speed and final accuracy. The gap between the two models widens as ω_0 increases, suggesting that the advantages of the linear solve become more pronounced in harder regimes. This trend is further highlighted in Figure 3.16 and Table 3.7.

J	FBPINN	ELM-FBPINN
10	(3.71±7.53)e-04	(2.62±4.47)e-07
20	(6.62±2.38)e-04	(1.81±4.2)e-06
40	(2.54±0.584)e-03	(2.49±4.26)e-05
80	(1.24±0.322)e-02	(6.92±12.1)e-05
160	(7.33±0.919)e-02	(2.19±8.95)e-03

Table 3.7: Final $\hat{e}_{L^1}$ values for each model at different solution frequency, ω_0.

Network Hidden Layer Depth With Varying Subdomains

This ablation study examines the effect of increasing the hidden layer depth of the networks. In addition to assessing how both models handle deeper architectures, we also investigate the impact of weight saturation, which is commonly observed in deep neural

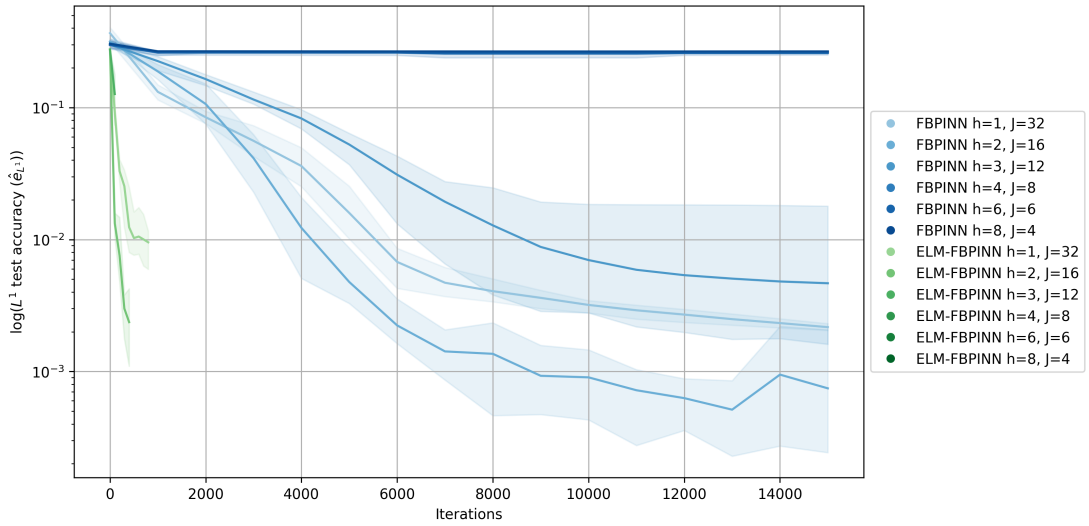


Figure 3.17: Convergence curves for increasing network layer depth, h and decreasing number of subdomains, J, for FBPINN and ELM-FBPINN.

networks with bounded activation functions. This experiment is motivated by the known trade-off between depth and trainability in physics-informed models, particularly when combined with domain decomposition.

Specifically, we increase the number of hidden layers h, while simultaneously decreasing the number of subdomains J or the number of basis functions C proportionally. This scaling preserves the overall parameter budget, allowing the effect of depth to be isolated from simple increases in model capacity. Due to numerical instability observed in the ELM-FBPINN linear solves for deeper networks, early stopping was imposed on the LSQR solver, which allowed the method to terminate stably while still reaching convergence when possible.

We first fix the number of basis functions, C, and investigate the effect of increasing the network depth h, while decreasing the number of subdomains, J. The convergence behaviour for both models is shown in Figure 3.17. For both the FBPINN and ELM-FBPINN, convergence is only observed for $h = 1, 2, 3$, while all deeper configurations fail to train satisfactorily. This suggests that increasing depth alone does not compensate for the reduced domain decomposition, and may in fact exacerbate training difficulties.

For the ELM-FBPINN, the imposition of early stopping on the LSQR solver makes

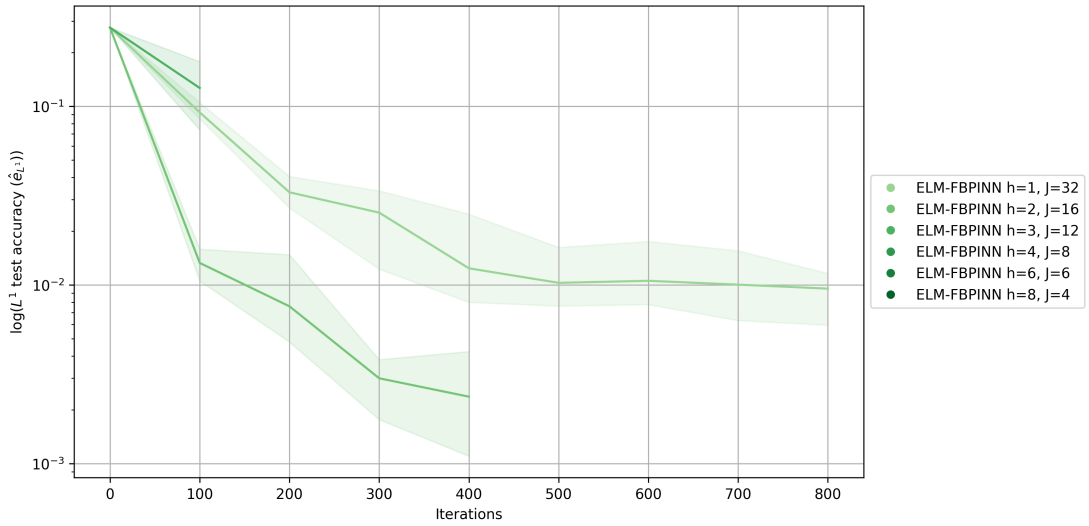


Figure 3.18: Convergence curves for increasing network layer depth, h and decreasing number of subdomains, J, for ELM-FBPINN.

it convenient to visualise the convergence behaviour separately, as shown in Figure 3.18. In these cases, early stopping prevents numerical divergence of the linear solver but cannot overcome the negative effect of excessive depth. We also remark that it is only when h increases that we observe any numerical instability in LSQR in our experiments. The behaviour of conjugate-gradient like Krylov subspace methods such as LSQR in finite precision is complicated, and depends on a number of factors including the orthogonality of the underlying basis [66]. However, it is known that convergence can be negatively affected by the condition number of the coefficient matrix.

It is clear that only models with $h < 3$ exhibit meaningful convergence. This observation is further supported by the iteration counts reported in Table 3.8, which show that LSQR terminates earlier as the depth increases, or fails to converge at all. A full summary of results, including the corresponding FBPINN configurations, is given in Table 3.9, confirming the lack of convergence for both models when $h > 3$. These results indicate that increasing depth does not provide a viable mechanism for improving accuracy in this setting.

At higher values of h, we observe pronounced saturation of the hidden-layer basis functions. Figure 3.19 shows representative basis functions for both FBPINN and

h	J	ELM-FBPINN	Iterations
1	32	(9.53±5.64)e-03	800
2	16	(2.37±3.14)e-03	400
3	12	(8.04±15.2)e-02	166
4	8	(2.76±0)e-01	0
6	6	(2.76±0)e-01	0
8	4	(2.77±0)e-01	0

Table 3.8: Final ELM-FBPINN $\hat{e}_{L^1}$ values and number of iterations for varying network depth, h and number of subdomains, J.

h	J	FBPINN	ELM-FBPINN
1	32	(2.17±0.259)e-03	(9.53±5.64)e-03
2	16	(7.45±18.3)e-04	(2.37±3.14)e-03
3	12	(4.67±16.3)e-03	(8.04±15.2)e-02
4	8	(2.65±0.0335)e-01	(2.76±0)e-01
6	6	(2.59±0.125)e-01	(2.76±0)e-01
8	4	(2.67±0.0153)e-01	(2.77±0)e-01

Table 3.9: Final $\hat{e}_{L^1}$ values for varying network depth, h and number of subdomains, J.

ELM-FBPINN at $h = 2$ and $h = 4$. In both cases, the deeper networks produce extremely flat basis functions, which is consistent with oversaturation of the tanh activation functions. Once saturated, these activations lose sensitivity to their inputs, effectively collapsing the expressive power of the hidden layers. As a consequence, the final output weights are unable to resolve the oscillatory structure of the solution, particularly when smaller values of J require each subdomain network to represent higher-frequency components. This highlights a key limitation of depth-based scaling for physics-informed models: without careful architectural design or activation choice, deeper networks can be strictly less effective than shallower ones, even when the total number of parameters is comparable.

The SVD condition number and rank of the ELM-FBPINN matrix, $\mathbf{M}$, can be seen in Figure 3.20 and Table 3.10. The condition number remains relatively stable, with all values being within the same order of magnitude. The rank progressively decreases as J is decreased, due to the number of columns in $\mathbf{M}$ decreasing. It appears, however, that in this instance the high condition numbers do affect the convergence of the LSQR

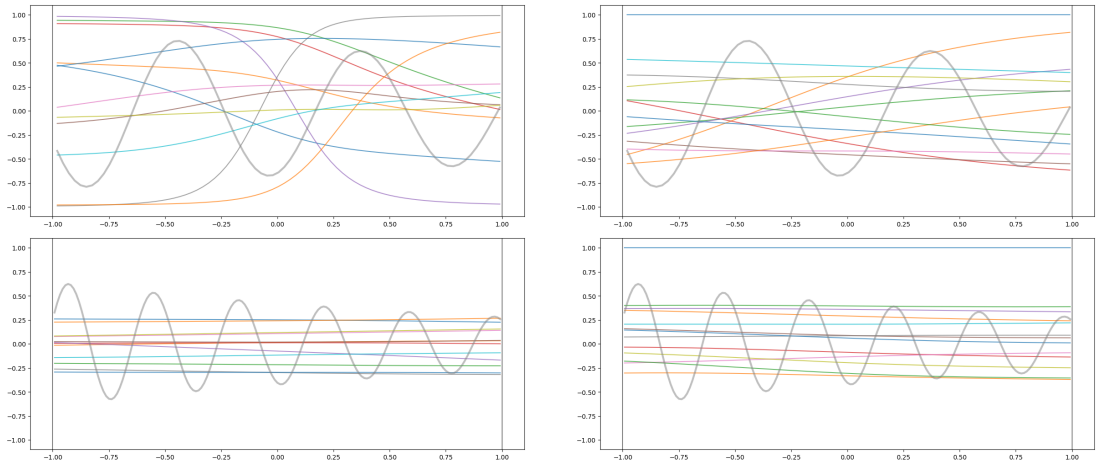


Figure 3.19: Hidden layer basis functions inside the 4th subdomain. Top: $h = 2$; Bottom: $h = 4$. Left: FBPINN: Right: ELM-FBPINN.

method which, as previously mentioned, suffers from numerical instabilities. Although it remains extremely challenging to pinpoint the cause of this instability, it would not be surprising if this were linked to the saturation of the basis functions shown in Figure 3.19.

Overall, we find that using a shallower network results in better performance for ELM-FBPINN (and FBPINN) when we vary depth with the number of subdomains, with $h = 2$ providing the best performance and the test loss for $h = 1$ being within the same order of magnitude. These issues appears to be related to the potential for basis functions to saturate when the depth is increased. Deeper networks result in worse approximations to the solution of the the damped harmonic oscillator problem and, for ELM-FBPINN, numerical instabilities within the LSQR least-squares solver. On the other hand, with early stopping of the least-squares solver, we find that ELM-FBPINN converges much more rapidly than FBPINN, provided that the network is not too deep. We stress that this issue with expressivity and depth is not an issue that is specific to ELM-FBPINN or FBPINN, but is a known drawback of deep neural networks [30].

Network Hidden Layer Depth With Varying Basis Functions

Finally, we investigate the effect of increasing the network depth, h whilst reducing the number of basis functions C, while keeping the number of subdomains fixed at

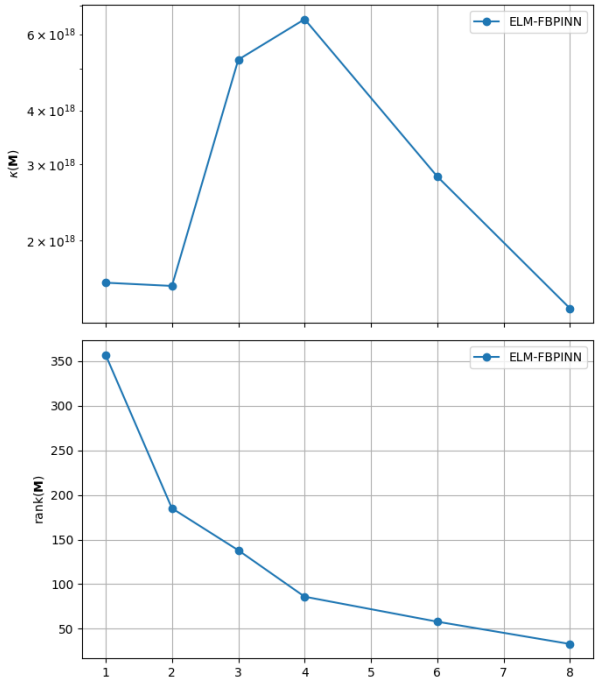


Figure 3.20: Condition number, $\kappa(\mathbf{M})$ and rank($\mathbf{M}$) vs depth of network layers, h. Note that while h is increased, number of subdomains, J, is decreased.

h	J	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)
1	32	1.60e+18	357
2	16	1.57e+18	185
3	12	5.24e+18	138
4	8	6.50e+18	86
6	6	2.81e+18	58
8	4	1.40e+18	33

Table 3.10: SVD-based condition number and matrix rank as h and J are varied for ELM-FBPINN.

$J = 20$. As the number of subdomains is fixed, a PINN model was also trained, with the same basis function and network depth parameters as FBPINN and ELM-FBPINN. Similarly to the previous tests, the LSQR solver used for ELM-FBPINN exhibits some numerical instability, and so early stopping is again applied. The convergence of each model is plotted in Figure 3.21, with the ELM-FBPINN convergence separately plotted in Figure 3.22 for visibility.

The PINN model does not converge at any value of h and C, which is expected behaviour, given the baseline results. Added complexity to the basis functions is not solely enough to address the spectral bias phenomenon. FBPINN converges to acceptably accurate solutions for almost all models except when $h = 8$ and $C = 8$. The best FBPINN ($h = 8$) model's $\hat{e}_{L^1}$ is within the same order of magnitude of the other test cases, but the range of the error indicates numerical instability, especially considering

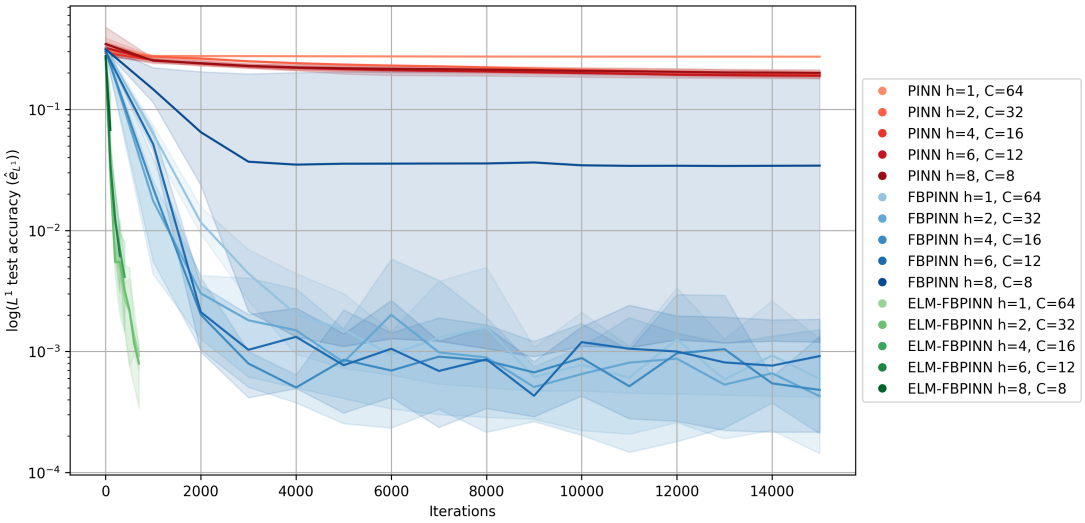


Figure 3.21: Convergence curves for increasing network layer depth, h and decreasing number of basis functions, C, for all models.

that $C = 8$ basis functions should be sufficient (see Table 3.3). Despite the numerical instability seen in ELM-FBPINN results, each model has converged, although the accuracy of the obtained solutions decreases as h increases (and C decreases). As h increases, the final $\hat{e}_{L^1}$ grows and LSQR terminates in fewer iterations. As before, ELM-FBPINN does converge more quickly than the other models, but FBPINN on average outperforms in terms of final $\hat{e}_{L^1}$ values. From Table 3.11, we can see that at the lowest values of h, ELM-FBPINN achieves a comparable performance to FBPINN, despite only running for < 900 iterations (see Table 3.12).

h	C	PINN	FBPINN	ELM-FBPINN
1	64	(2.73±0.00265)e-01	(5.94±8.49)e-04	(5.06±6.87)e-04
2	32	(1.95±0.236)e-01	(4.27±13.7)e-04	(6.29±8.75)e-04
4	16	(1.91±0.253)e-01	(4.81±11.3)e-04	(3.01±2.57)e-03
6	12	(1.9±0.22)e-01	(9.15±16.3)e-04	(7±13.5)e-03
8	8	(2±0.23)e-01	(3.43±19.5)e-02	(4.26±7.66)e-02

Table 3.11: Final $\hat{e}_{L^1}$ values for varying network depth, h and number of basis functions, C.

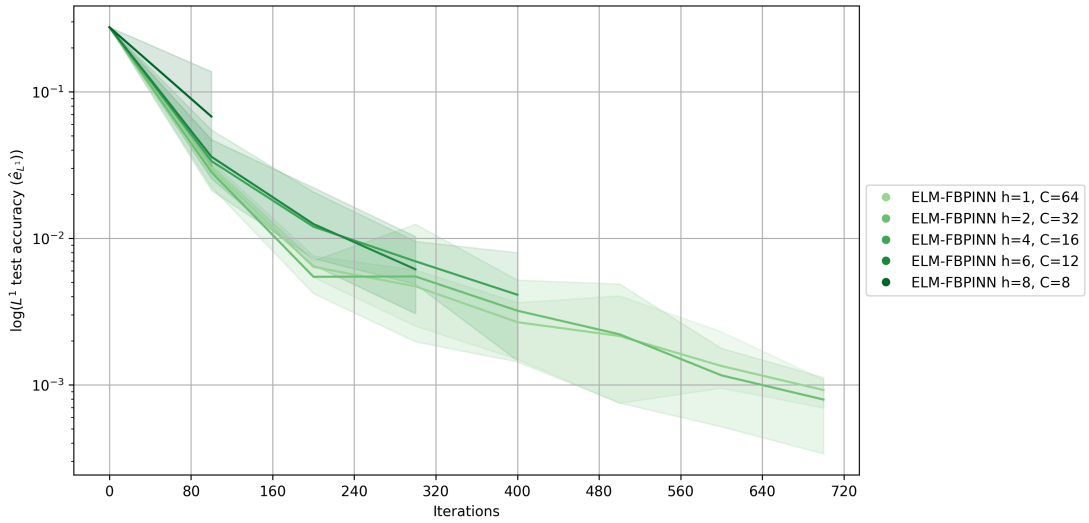


Figure 3.22: Convergence curves for increasing network layer depth, h and decreasing number of basis functions, C, for ELM-FBPINN.

h	C	$\hat{e}_{L^1}$	Iterations
1	64	(5.06±6.87)e-04	883
2	32	(6.29±8.75)e-04	883
4	16	(3.01±2.57)e-03	566
6	12	(7±13.5)e-03	400
8	8	(4.26±7.66)e-02	166

Table 3.12: Final ELM-FBPINN $\hat{e}_{L^1}$ values and number of iterations for varying network depth, h and number of basis functions, C.

The SVD condition number and rank of the ELM-FBPINN matrix, $\mathbf{M}$, can be seen in Figure 3.23 and Table 3.13. As h is increased, with decreasing C, the condition number also decreases. This is likely due to the number of columns also decreasing with C, rather than an indication of deeper layer networks being better conditioned. Additionally, while the rank appears to be stable across tests, the matrix $\mathbf{M}$ has many singular values that are numerically singular, indicating that many columns of this matrix are linearly dependent. This is again likely attributed to deep networks causing saturation of the tanh activation function (cf. Figure 3.19). Moreover, similarly to above, this seems to cause numerical instability in the LSQR solver.

In summary, for the 1D damped harmonic oscillator, we find that ELM-FBPINN is

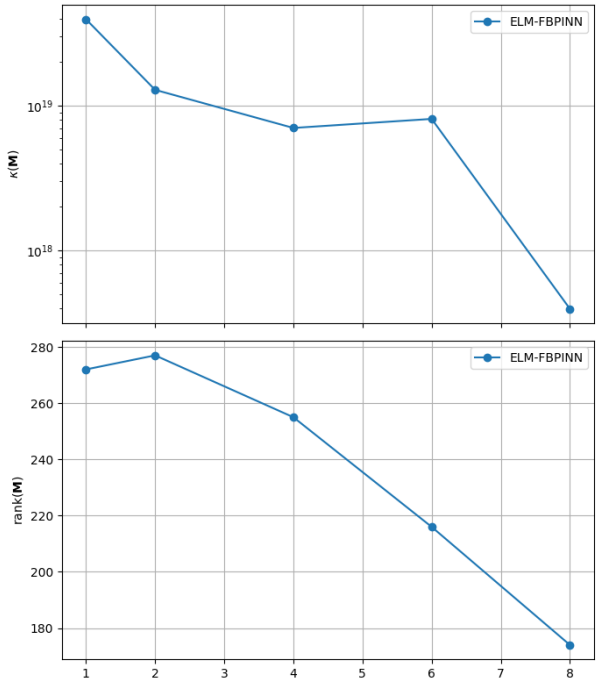


Figure 3.23: Condition number, $\kappa(\mathbf{M})$ and $\text{rank}(\mathbf{M})$ vs depth of network layers, h. Note that while h is increased, number of basis functions, C, is decreased.

h	C	$\kappa(\mathbf{M})$	$\text{rank}(\mathbf{M})$
1	64	3.96e+19	272
2	32	1.29e+19	277
4	16	7.02e+18	255
6	12	8.11e+18	216
8	8	3.95e+17	174

Table 3.13: SVD-based condition number and matrix rank as h and C are varied for ELM-FBPINN.

an attractive modification of the FBPINN approach, which retains attractive features of the original FBPINN method, specifically its ability to accurately solve highly oscillatory problems, whilst improving efficiency by replacing expensive gradient-based optimisation by a linear least-squares solve. Our results show ELM-FBPINN easily outperforms the original PINN model, and that it is possible to achieve more accurate solutions than FBPINN, at least as measured by the L^1 error $\hat{e}_{L^1}$. In addition, convergence of the least-squares solver in ELM-FBPINN is almost always achieved in fewer iterations than for the gradient-based solver in FBPINN and PINN.

We see that ELM-FBPINN is robust with respect to the choice of the number of basis functions and subdomains used, provided that these are not so few that the oscillatory solution cannot be captured by the basis functions. Interestingly, ELM-FBPINN is able to approximate even highly oscillatory solutions ($\omega_0 = 320$), provided that sufficient subdomains are used to overcome the effect of spectral bias. Moreover, the

obtained solutions are significantly more accurate than those obtained by FBPINN. However, we caution that both ELM-FBPINN and FBPINN are better suited to shallower networks ($h = 1, 2$), at least when used with tanh activation functions, for which oversaturation can cause problems.

3.4.2 2D - Multiscale Laplace

Baseline

We now evaluate all models on the 2D multiscale Laplace problem and establish a baseline set of results. This problem provides a significantly more demanding test than the 1D oscillator, as it combines higher dimensionality with multi-frequency spatial structure, thereby increasing both representational and optimisation difficulty. All models are trained on $\left(\lceil\sqrt{C+1}\rceil\, J\right)^2$ points and tested on $\left(\lceil\sqrt{30}\rceil\, J\right)^2$ points where $\lceil\cdot\rceil$ denotes the ceiling function. This sampling ensures approximately uniform resolution within each subdomain while maintaining an overdetermined least-squares system for the ELM-FBPINN formulation.

The PINN will have $h = 2$ layers and $C = 64$ basis functions by default as in the 1D tests. These parameters will vary in line with the other models for all tests that specifically alter them. This configuration provides the PINN with substantial expressive capacity, ensuring that any performance limitations primarily stem from optimisation challenges rather than insufficient network size.

We fix $n_\omega = 4$, which results in frequency components $\omega_1 = 4$, $\omega_2 = 8$, $\omega_3 = 12$, $\omega_4 = 16$. This provides a solution (see Figure 3.3) that has sufficient multiscale features, which should prove difficult for a PINN to solve. In particular, the superposition of multiple sine modes introduces increasingly fine spatial oscillations, which are known to exacerbate spectral bias in globally trained neural networks. The number of subdomains per axis is set to $J = 16$, and the number of basis functions is $C = 16$. All models are trained for 15,000 iterations unless explicitly stated otherwise.

A plot of convergence for all models is shown in Figure 3.24. PINN failed to converge fully, which is expected given the multiscale solution domain. The slow and incomplete convergence reflects the difficulty of resolving multiple high-frequency com-

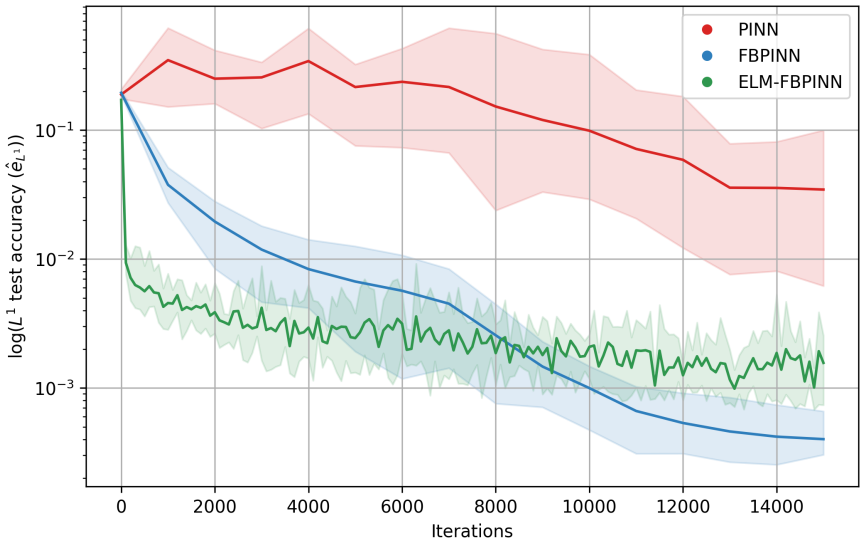


Figure 3.24: Convergence curves for baseline models for the 2D multiscale Laplace problem.

ponents simultaneously using gradient-based optimisation in a single global network. FBPINN and ELM-FBPINN have both converged to an acceptable $\hat{e}_{L^1}$, with FBPINN achieving the best final error. The improved performance of FBPINN relative to PINN highlights the benefit of domain localisation in mitigating spectral bias. ELM-FBPINN exhibits a rapid initial convergence rate, with the loss dropping sharply within the first $\approx$ 1,000 iterations. This rapid descent corresponds to the early iterations of the LSQR solver efficiently reducing the dominant residual components of the linear system. The rate then slows, with the loss gradually decreasing for the remaining iterations. The subsequent plateau suggests that the remaining error components correspond to finer-scale interactions that are more difficult to eliminate within the fixed random feature basis. Table 3.14 shows the final $\hat{e}_{L^1}$ values, as well as the average run time per model. The PINN is significantly slower to train on this problem, partially due to the additional hidden layer and number of basis functions per layer. More importantly, the extended runtime reflects the increased optimisation difficulty associated with fitting multiscale structure in a high-dimensional domain. It also indicates that the problem is difficult for the PINN to optimise for. While the FBPINN and ELM-FBPINN have an increased run time compared to the 1D baseline results (see Table 3.1), the PINN run

time is an order of magnitude longer. Between the localised methods, ELM-FBPINN achieves a slightly faster runtime than FBPINN, although FBPINN attains a lower final error. This illustrates a clear trade-off: the linear least-squares formulation yields improved computational efficiency, whereas gradient-based optimisation of trainable hidden weights provides additional flexibility that is beneficial for complex multiscale spatial problems.

Model	$\hat{e}_{L^1}$	Time (s)
PINN	(3.46±9.35)e-02	209±0.62
FBPINN	(4.02±3.62)e-04	25.3±0.0227
ELM-FBPINN	(1.55±1.84)e-03	21.4±0.111

Table 3.14: Baseline results for the 2D multiscale Laplace.

Despite the differences in $\hat{e}_{L^1}$ values, each model managed to accurately predict the test solution as seen in Figure 3.25. Qualitatively, all models recover the dominant spatial oscillations of the exact solution. The only visual difference between the PINN model's predictions is some minor refinement that both the FBPINN and ELM-FBPINN appear to have resolved. These refinements correspond to improved resolution of higher-frequency components, which are more accurately captured when localisation or linear least-squares fitting is employed.

Varying Number of Basis Functions

Next we fix all parameters and vary the number of basis functions, C in the hidden layer. This study isolates the effect of representational capacity while keeping the domain decomposition and sampling strategy fixed. For the PINN, we continue to use $h = 2$ layers, while FBPINN and ELM-FBPINN retain a single layer. This ensures that changes in performance can be attributed primarily to the number of basis functions rather than architectural depth.

A plot of model convergence can be seen in Figure 3.26. All PINN models have similar trends and have not converged. The near-identical behaviour across values of C suggests that increasing or decreasing the number of neurons alone does not overcome the optimisation difficulty associated with multiscale structure in a single

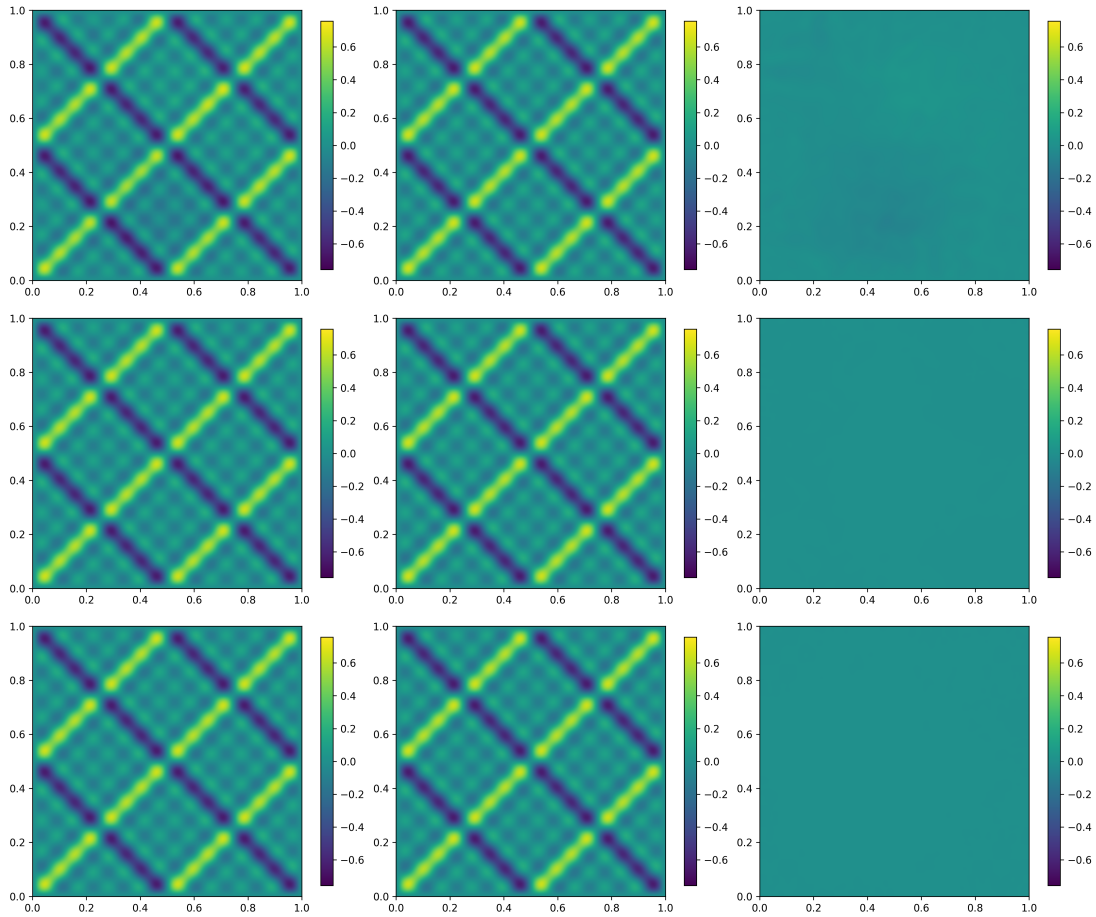


Figure 3.25: Baseline model predictions for the 2D multiscale Laplace problem. The rows correspond to the solutions obtained with PINN (top), FBPINN (middle) and ELM-FBPINN (bottom). For each method the predicted solution (left), exact solution (middle) and difference between the predicted and exact solutions (right) are shown.

global network. Unlike in the baseline test with $C = 64$ basis functions per layer, the relatively lower values of C are insufficient for PINN. This indicates that, in two dimensions, the PINN requires both substantial width and effective optimisation to approximate the composite frequency content.

FBPINN converges for all models, with an improvement in $\hat{e}_{L^1}$ as C increases. The near monotonic improvement demonstrates that additional local basis functions enhance expressivity within each subdomain, allowing finer spatial oscillations to be resolved. Unlike the other models, FBPINN has the greatest variation in $\hat{e}_{L^1}$ as C is varied, and ultimately achieves the best precision. This suggests that trainable

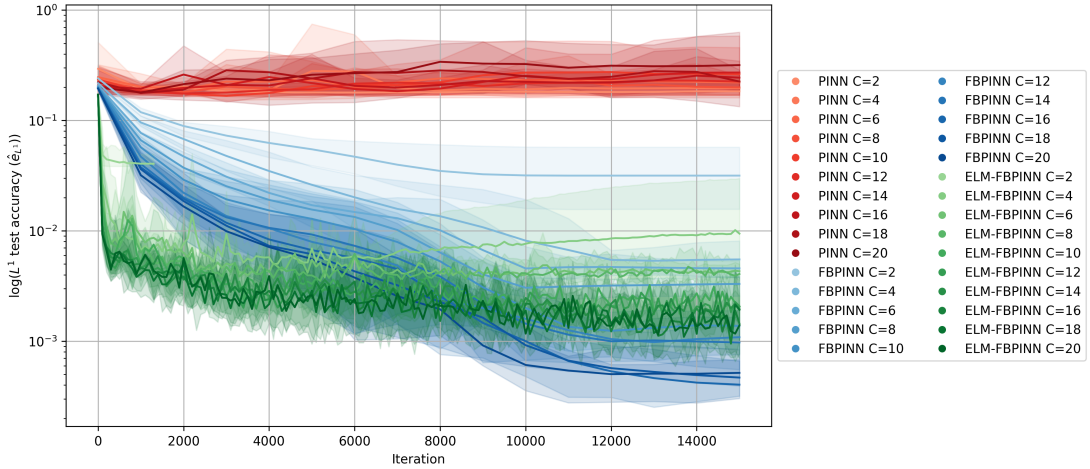


Figure 3.26: Convergence curves for all models for varying number of basis functions C.

hidden weights provide meaningful additional flexibility in the 2D multiscale setting, particularly when sufficient local capacity is available.

ELM-FBPINN also converges for most models, with the exception of $C = 2$ for which there is some numerical instability resulting in the solver being unable to complete all 15,000 iterations. In this extremely low-capacity regime, the linear system becomes poorly posed relative to the complexity of the solution, and the LSQR iterations amplify residual components instead of reducing them effectively. For some of the lower values of C the ELM-FBPINN models seem to be diverging, which shows the need for early stopping to ensure the best performance. This behaviour is characteristic of over-iterating an ill-conditioned least-squares system, where later iterations may increase the test error despite reducing the algebraic residual.

The speed of convergence for ELM-FBPINN is again the fastest between the models, with a sharp initial descent before a more gradual decline, excluding the models that exhibit some divergence. The rapid initial decrease reflects the efficiency of solving for the dominant linear coefficients in a single global least-squares formulation, while the slower tail phase corresponds to resolving finer-scale interactions between subdomains.

All final errors are reported in Table 3.15, along with a plot of $\hat{e}_{L^1}$ against C in Figure 3.27. It is clear that FBPINN has achieved the best precision, especially at

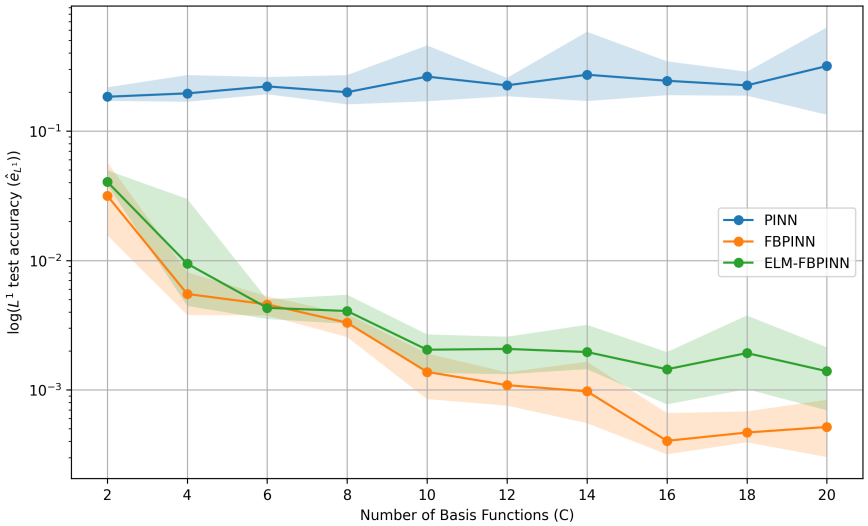


Figure 3.27: Final L^1 test loss ($\hat{e}_{L^1}$) vs number of basis functions (C) for all models.

higher values of C, but ELM-FBPINN presents comparable performance. In particular, for moderate to large C, the ELM-FBPINN achieves errors within approximately one order of magnitude of FBPINN, but in potentially half the required iterations. This highlights a precision–efficiency trade-off between trainable hidden representations and fixed random feature approximations.

There is a large error for ELM-FBPINN when $C = 4$, which similarly to $C = 2$ appears to have some numerical instability. The large variance observed in this regime further indicates sensitivity to the random initialisation of hidden weights, as small basis sets increase the probability of poorly aligned random features. This is also likely due to the size of the matrix, $\mathbf{M}$, being insufficient for the problem domain as C becomes too small.

C	PINN	FBPINN	ELM-FBPINN
2	(1.84±0.465)e-01	(3.16±4.16)e-02	(4.05±1.13)e-02
4	(1.95±1.02)e-01	(5.51±4.32)e-03	(9.47±25.2)e-03
6	(2.21±0.69)e-01	(4.57±1.53)e-03	(4.32±1.62)e-03
8	(1.99±1.09)e-01	(3.3±1.29)e-03	(4.05±2.01)e-03
10	(2.63±2.88)e-01	(1.38±1.06)e-03	(1.81±0.935)e-03
12	(2.25±0.712)e-01	(1.09±0.614)e-03	(1.75±0.855)e-03
14	(2.72±4.13)e-01	(9.73±11)e-04	(1.83±1.08)e-03
16	(2.44±1.56)e-01	(3.99±3.47)e-04	(1.54±1.79)e-03
18	(2.25±0.995)e-01	(4.7±2.91)e-04	(1.97±2.84)e-03
20	(3.17±4.97)e-01	(5.17±5.28)e-04	(1.61±1.07)e-03

Table 3.15: Final $\hat{e}_{L^1}$ values and average run time for varying number of basis functions, C.

Varying Number of Subdomains

We now vary the number of subdomains and fix all other parameters for FBPINN and ELM-FBPINN. In this case, J represents the per-axis number of subdomains. Since the domain is two-dimensional, the total number of subdomains, as well as the number of columns of the ELM-FBPINN matrix, $\mathbf{M}$, increase quadratically with J. Hence, localisation becomes significantly finer as J increases, which results in rapidly improved expressive capacity.

The convergence of both models can be seen in Figure 3.28. It is clear that FBPINN has reached stable solutions more consistently than ELM-FBPINN, with only the $J = 2$ and $J = 4$ models failing to converge fully. On the other hand, ELM-FBPINN has only fully converged for models with $J > 8$. For small values of J, each subdomain remains too large to isolate the higher-frequency components of the solution, leading to persistent approximation errors. Additionally, FBPINN has lower $\hat{e}_{L^1}$ for the models with the largest J. This suggests that, in two dimensions, the ability to adapt hidden-layer weights through gradient-based optimisation provides additional flexibility that becomes increasingly beneficial as localisation is refined.

For ELM-FBPINN models with $J > 10$, the speed of convergence is initially faster than FBPINN, and then slows to a gradual decline for the remaining iterations. The rapid initial decrease reflects the efficient global least-squares solve, whereas the slower tail phase likely corresponds to residual inter-subdomain interactions that are more

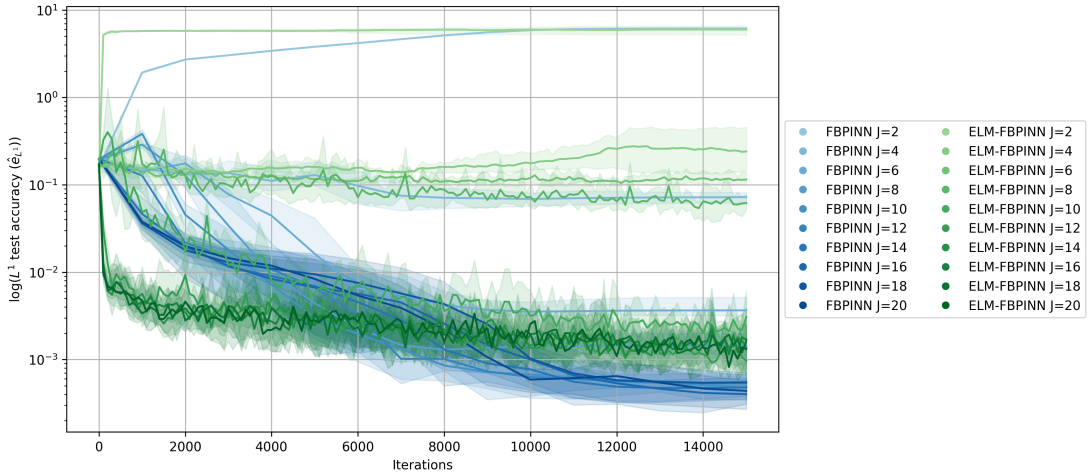


Figure 3.28: Convergence curves for FBPINN and ELM-FBPINN for varying number of subdomains per axis (J).

difficult to capture with fixed random features.

The final $\hat{e}_{L^1}$ against the number of subdomains per axis has been plotted in Figure 3.29, with the exact values reported in Table 3.16. As well as confirming FBPINN's precision advantage at higher values of J, it is now clearer to see that FBPINN outperforms ELM-FBPINN for almost every value of J. In contrast to the 1D experiments, increasing localisation alone is not sufficient for the ELM-FBPINN to match the accuracy of the fully trainable FBPINN.

The only outlier is $J = 2$, for which neither model found an acceptable solution, and for which both exhibited some numerical instability. In this regime, the domain decomposition is too coarse to effectively mitigate spectral bias, and both methods struggle to represent the composite multi-frequency solution.

At higher values of J, the models are closer in performance but FBPINN remains approximately one order of magnitude more accurate than ELM-FBPINN. This persistent accuracy gap indicates that, for the 2D multiscale Laplace problem, the flexibility of trainable hidden weights is more important than in the 1D oscillator case.

In this case, the gradient-descent-based optimisation of FBPINN is better suited to the problem domain, and the complexity of the multiscale solution may require more flexibility in the hidden network weights. The fixed random feature basis of the ELM-

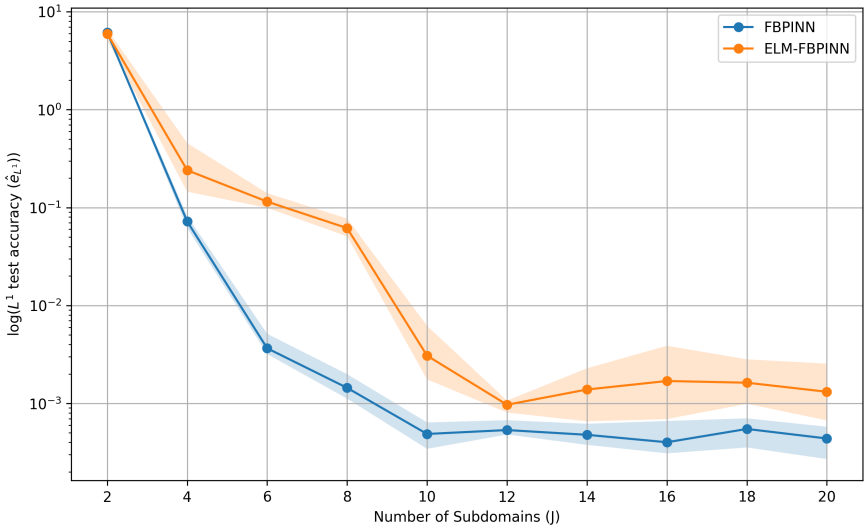


Figure 3.29: Final L^1 test loss ($\hat{e}_{L^1}$) vs number of subdomains per axis (J).

FBPINN limits its ability to adapt to anisotropic or interacting frequency components that arise in two spatial dimensions.

J	FBPINN	ELM-FBPINN
2	(6.15±0.264)e+00	(5.96±1.43)e+00
4	(7.19±1.99)e-02	(2.41±3.12)e-01
6	(3.65±1.93)e-03	(1.16±0.464)e-01
8	(1.44±0.864)e-03	(5.72±2.27)e-02
10	(5.01±2.82)e-04	(1.97±0.694)e-03
12	(5.31±1.95)e-04	(1.04±0.688)e-03
14	(4.79±2.4)e-04	(1.57±2.88)e-03
16	(4.02±3.52)e-04	(1.46±1.25)e-03
18	(5.48±3.49)e-04	(1.45±1.97)e-03
20	(4.38±3.03)e-04	(1.3±1.81)e-03

Table 3.16: Final $\hat{e}_{L^1}$ values for varying number of subdomains, J.

Scaling Solution Frequency with Subdomains

Now we investigate the ability of the models to find solutions for progressively higher frequency problems. Specifically, we vary n_ω, which determines the number of frequency components included in (3.12) and (3.13). Increasing n_ω increases the spectral richness of the exact solution, introducing additional interacting oscillatory modes across the domain. We also scale the network capacity by increasing J alongside n_ω. By coupling

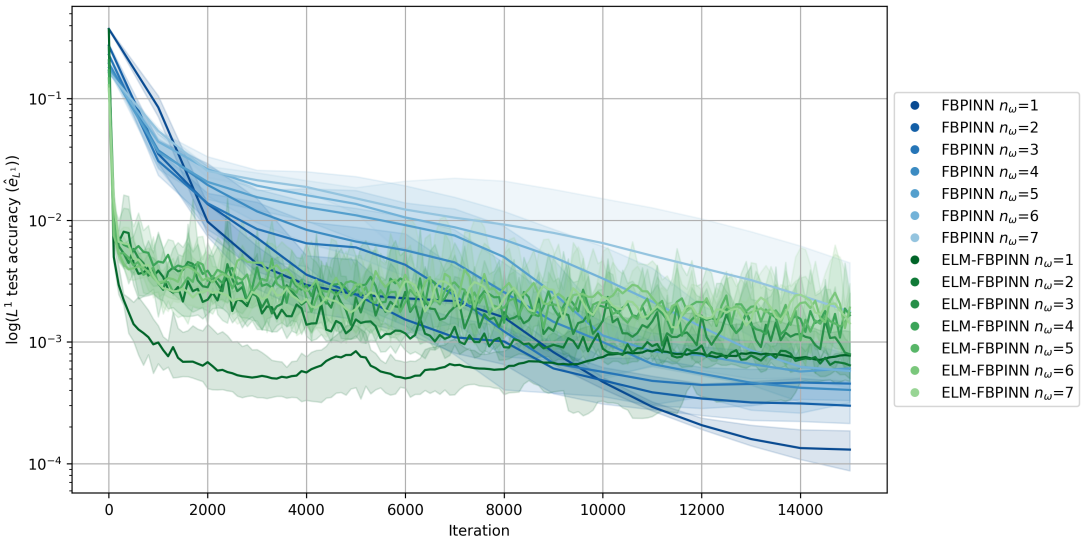


Figure 3.30: Final L^1 test loss ($\hat{e}_{L^1}$) vs number of subdomains per axis (J) as the number of frequency components n_ω, is varied.

J and n_ω, the effective spatial resolution per subdomain is kept approximately constant, allowing us to isolate how well each training strategy handles increasing global complexity. Again, the ideal result would be little variation between $\hat{e}_{L^1}$ values for each model. Such behaviour would indicate robustness of the method with respect to problem frequency and demonstrate scalability beyond a fixed baseline configuration.

Convergence plots for each model can be seen in Figure 3.30. ELM-FBPINN is the model with fastest convergence at each value of n_ω, and on average is marginally more consistent than FBPINN. The consistently steep initial descent reflects the efficiency of solving a single global linear least-squares system, even as the number of oscillatory components increases. Despite being more distinct on the plot, FBPINN $\hat{e}_{L^1}$ values are smaller, meaning the difference between each model's value is also smaller. Thus, although the visual separation between convergence curves may appear significant, the absolute difference in final error remains modest in most cases.

The final $\hat{e}_{L^1}$ against n_ω is plotted in Figure 3.31, and the numerical results can be seen in Table 3.17. The difference between the maximum and minimum values is also reported. This range provides a simple measure of stability across increasing problem difficulty.

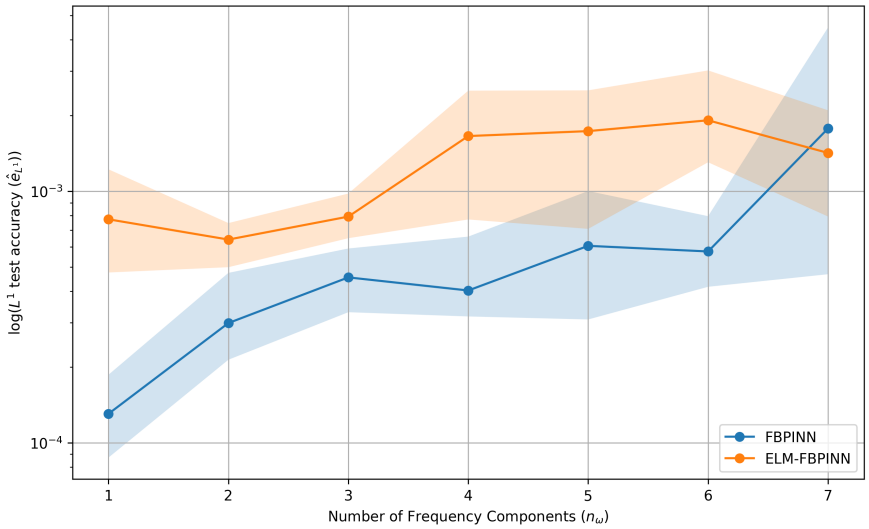


Figure 3.31: Final L^1 test loss ($\hat{e}_{L^1}$) vs number of subdomains per axis (J) as the number of frequency components n_ω, is varied.

It is now clearer to see that, despite performance generally degrading as n_ω increases, both models are relatively stable as the problem scales. In particular, the variation in $\hat{e}_{L^1}$ remains within the same order of magnitude across all tested values of n_ω, indicating that neither method exhibits catastrophic degradation as frequency increases.

Again the efficiency trade-off of ELM-FBPINN is noted, with FBPINN achieving better precision, but requiring the full iteration budget. Since the LSQR solver often reaches effective convergence well before the maximum iteration count, adaptive stopping criteria would likely reduce runtime without compromising accuracy.

The trend of ELM-FBPINN final $\hat{e}_{L^1}$ values as n_ω increases is generally flatter than that of FBPINN, which exhibits a positive correlation between error and solution frequency. The smaller max–min range for ELM-FBPINN (1.19×10^{-3} compared to 1.65×10^{-3} for FBPINN) suggests slightly more uniform performance across increasing spectral complexity. This shows that ELM-FBPINN has more potential for scalability than FBPINN for the 2D multiscale Laplace. While FBPINN achieves lower absolute errors overall, the comparatively flatter degradation profile of ELM-FBPINN indicates that fixed random features combined with increasing localisation can provide competitive robustness as global frequency content grows.

n_ω	FBPINN	ELM-FBPINN
0	(1.33±1)e-04	(7.8±7.79)e-04
1	(2.97±2.57)e-04	(6.75±2.98)e-04
2	(4.52±2.62)e-04	(8.38±8.29)e-04
3	(3.98±3.6)e-04	(1.86±2.17)e-03
4	(6.06±6.87)e-04	(1.7±1.53)e-03
5	(5.8±3.83)e-04	(1.86±1.79)e-03
6	(1.78±4.01)e-03	(1.37±1.42)e-03
Max-Min	1.65e-03	1.19e-03

Table 3.17: Final e_{L^1} values and average run time for varying n_ω.

Network Hidden Layer Depth With Varying Subdomains

We now vary the depth of the networks together with the number of subdomains per axis. As in the corresponding 1D experiment, increasing h is accompanied by a reduction in J in order to maintain a comparable overall parameter budget. This allows us to isolate the effect of depth from simple increases in model size.

Plots of convergence for FBPINN and ELM-FBPINN can be seen in Figure 3.32. The final $\hat{e}_{L^1}$ for both models are plotted in Figure 3.33, with exact values reported in Table 3.18. Similarly to the 1D tests, we observe a drop in performance as h increases, likely due to the saturation of hidden basis functions. For $h \leq 2$, FBPINN converges to the lower $\hat{e}_{L^1}$ values. This indicates that, despite saturation effects, trainable hidden weights retain some adaptive flexibility, particularly when at least moderate localisation ($J \geq 8$) is preserved.

ELM-FBPINN exhibits steeper convergence up to iteration $\approx$ 1,000, before a more gradual decline for the remaining runtime. The rapid early decrease reflects efficient resolution of the dominant linear coefficients, while the later stagnation suggests that fixed random features lack sufficient expressivity to compensate for the reduced domain decomposition at higher depths.

At $h > 2$, both models produce $\hat{e}_{L^1}$ values above 10^{-1}, exhibiting behaviour similar to the corresponding 1D test. This confirms that increasing depth does not provide a viable alternative to localisation for mitigating spectral bias in multiscale problems.

Unlike the 1D test, however, the LSQR solver of ELM-FBPINN continues to run

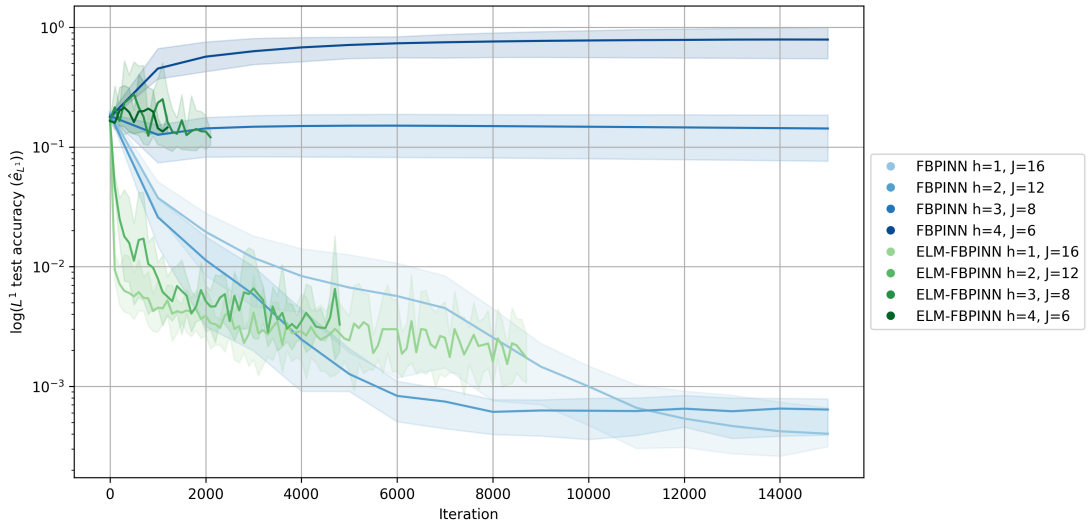


Figure 3.32: Convergence curves for FBPINN and ELM-FBPINN for varying network depth (h) and number of subdomains per axis (J).

h	J	FBPINN	ELM-FBPINN
1	16	(4.01±3.52)e-04	(1.48±0.968)e-03
2	12	(6.34±4.48)e-04	(5.9±14.4)e-03
3	8	(1.43±1.09)e-01	(1.27±0.931)e-01
4	6	(8.05±3.88)e-01	(1.43±0.494)e-01

Table 3.18: Final $\hat{e}_{L^1}$ values and average run time for varying number of hidden layers (h) and number of subdomains per axis (J).

for $h = 4$, with the number of iterations for all values of h reported in Table 3.19. The decreasing number of LSQR iterations as h increases reflects the reduced size of the linear system resulting from smaller values of J, rather than improved conditioning or accuracy. Despite higher $\hat{e}_{L^1}$ values for ELM-FBPINN at $h = 1$ and $h = 2$, the model reaches stable solutions in a significantly lower number of iterations. This demonstrates that computational efficiency alone does not guarantee solution quality when architectural choices limit expressivity.

Network Hidden Layer Depth With Varying Basis Functions

Finally, we vary the number of hidden layers together with the number of basis functions (C), while keeping the number of subdomains fixed. This experiment isolates the

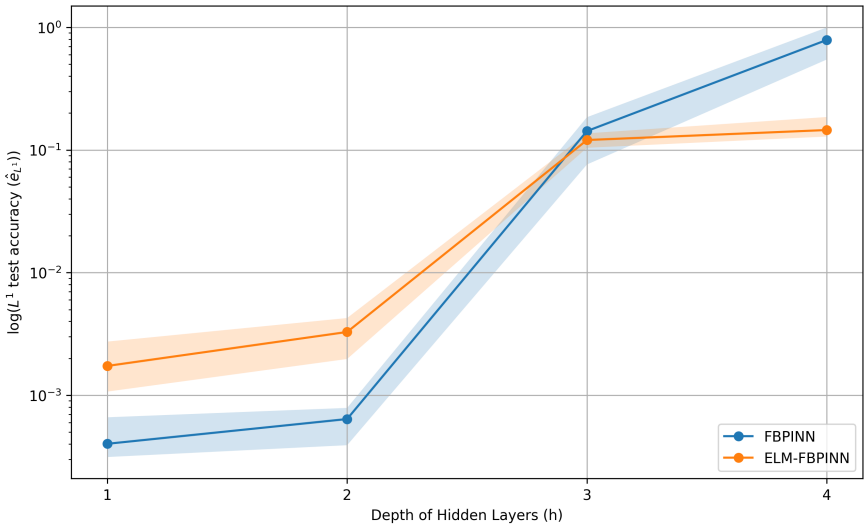


Figure 3.33: Final L^1 test loss ($\hat{e}_{L^1}$) for FBPINN and ELM-FBPINN for varying network depth (h) and number of subdomains per axis (J).

h	J	$\hat{e}_{L^1}$	Iterations
1	16	(1.48±0.968)e-03	8700
2	12	(5.9±14.4)e-03	4800
3	8	(1.27±0.931)e-01	2100
4	6	(1.43±0.494)e-01	1200

Table 3.19: Final L^1 test loss ($\hat{e}_{L^1}$) values and number of iterations for varying network depth, h and number of subdomains, J.

interaction between depth and width within a fixed localisation framework. Since the number of subdomains is fixed, we reintroduce the PINN model, varying h and C identically for all networks.

The convergence of all networks can be seen in Figure 3.34. Again, the PINN fails to find an adequate solution for all values of h, with convergence curves remaining flat and near their initial values throughout training. This confirms that increasing architectural depth alone does not alleviate optimisation challenges in the absence of domain decomposition.

FBPINN converges for all models, with performance degrading as h increases. The degradation is consistent with activation saturation and increased optimisation difficulty in deeper networks. Similarly, ELM-FBPINN converges for all models, exhibiting

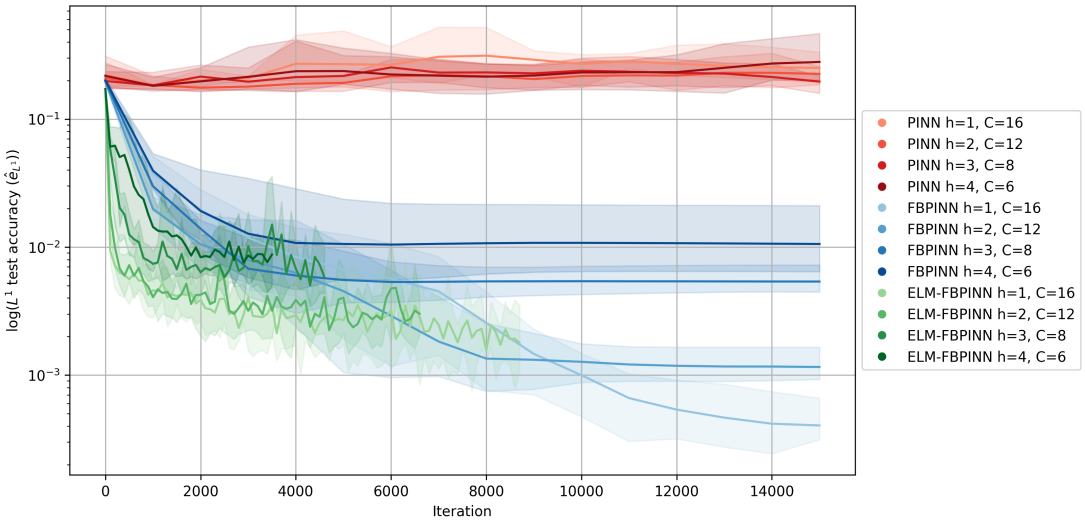


Figure 3.34: Convergence curves for FBPINN and ELM-FBPINN for varying network depth (h) and number of basis functions (C).

sharper initial convergence than FBPINN, but noticeably higher $\hat{e}_{L^1}$ values for $h = 1$ and $h = 2$. This indicates that, when sufficient width is available, adaptive hidden weights provide measurable benefits in representing multiscale spatial interactions.

The final $\hat{e}_{L^1}$ is plotted against h in Figure 3.35. Table 3.20 reports the final $\hat{e}_{L^1}$ values for each model together with average runtime. For $h < 3$, FBPINN achieves superior precision compared to both PINN and ELM-FBPINN, while performance becomes nearly identical to ELM-FBPINN for $h \geq 3$. This convergence of performance at higher depths reflects a shared limitation imposed by saturation effects, which restrict effective representational capacity regardless of training strategy.

The number of LSQR iterations used by ELM-FBPINN is shown in Table 3.21. As h increases and C decreases, the required number of iterations steadily declines, reflecting the reduced dimensionality of the least-squares system rather than improved approximation quality.

3.5 Discussion and conclusions

In this chapter, we compared PINN, FBPINN and ELM-FBPINN across a comprehensive set of ablation studies on two representative problems: a 1D damped harmonic

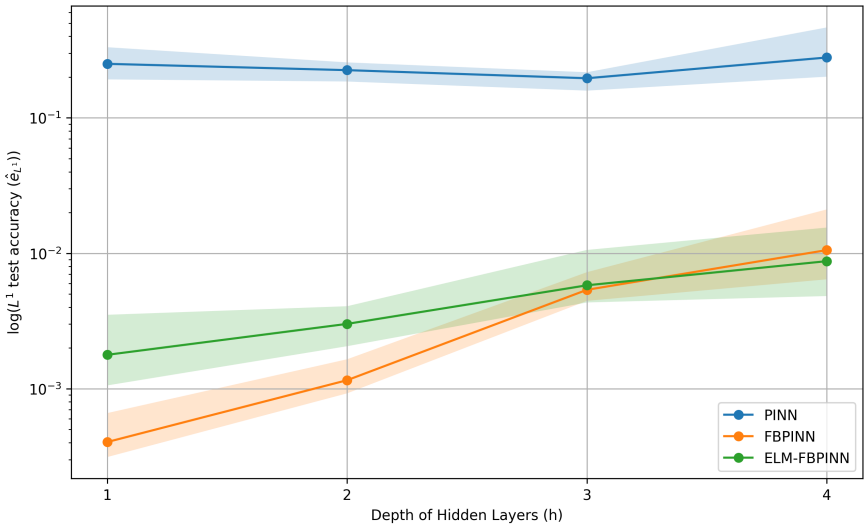


Figure 3.35: Final $\hat{e}_{L^1}$ for FBPINN and ELM-FBPINN for varying network depth (h) and number of basis functions (J).

h	C	PINN	FBPINN	ELM-FBPINN
1	16	(2.51±1.4)e-01	(3.99±3.5)e-04	(1.92±0.948)e-03
2	12	(2.25±0.712)e-01	(1.16±0.72)e-03	(2.86±2.14)e-03
3	8	(1.96±0.583)e-01	(5.37±2.79)e-03	(5.57±5.16)e-03
4	6	(2.79±2.64)e-01	(1.06±1.46)e-02	(9.19±10.8)e-03

Table 3.20: Final e_{L^1} values for varying number of hidden layers (h) and basis functions (C).

h	C	$\hat{e}_{L^1}$	Iterations
1	16	(1.92±0.948)e-03	8700
2	12	(2.86±2.14)e-03	6600
3	8	(5.57±5.16)e-03	4600
4	6	(9.19±10.8)e-03	3500

Table 3.21: Final $\hat{e}_{L^1}$ values and number of iterations for varying network depth, h and number of basis functions, C.

oscillator and a 2D multiscale Laplace equation. These experiments were designed to isolate the effects of domain decomposition, network capacity, depth, and solution frequency, while assessing convergence behaviour, stability, and computational efficiency.

Across all studies, ELM-FBPINN consistently demonstrates a significantly faster initial convergence rate than both PINN and FBPINN. For both the 1D and 2D prob-

lems, the characteristic behaviour of ELM-FBPINN consists of a sharp early reduction in error, followed by a more gradual convergence phase. This behaviour reflects the efficiency of solving a global linear least-squares problem, where dominant error components are rapidly eliminated before finer-scale residuals are resolved.

As expected, PINNs fail to converge to acceptable solutions for the multiscale configurations considered. Without localisation or architectural strategies to mitigate spectral bias, the global network struggles to represent high-frequency components, resulting in consistently large $\hat{e}_{L^1}$ values. These results reaffirm the well-documented difficulty of training standard PINNs on oscillatory or multiscale problems.

For the 1D damped harmonic oscillator, ELM-FBPINN achieves the best precision across nearly all ablation studies, except in tests involving increased network depth. In these cases, deeper architectures lead to activation saturation, causing hidden-layer basis functions to become nearly linear. ELM-FBPINN appears more sensitive to this effect, likely due to the frozen hidden weights, which limit the model's ability to adapt internal representations.

In the 2D multiscale Laplace problem, ELM-FBPINN typically achieves errors within the same order of magnitude as FBPINN, although FBPINN generally stabilises at a lower final $\hat{e}_{L^1}$. This suggests that, in higher dimensions with interacting frequency components, the flexibility of trainable hidden weights provides a measurable advantage.

The scaling experiments provide additional insight into robustness. When solution frequency and network capacity are increased simultaneously, ELM-FBPINN maintains relatively stable error levels, exhibiting only moderate degradation as problem difficulty grows. FBPINN displays similarly stable behaviour, although certain trends suggest mild deterioration in precision as both frequency and capacity increase. Overall, these results indicate that both localisation and linear least-squares training scale favourably, with ELM-FBPINN offering competitive robustness at reduced computational cost if an early stopping regime is considered.

From a numerical linear algebra perspective, the conditioning of the ELM-FBPINN system matrix $\mathbf{M}$ plays a central role in performance. The SVD-based condition num-

ber is typically large (on the order of 10^{18}) and grows rapidly as the number of basis functions C increases. This growth is primarily attributable to increasing linear dependence among randomly initialised basis functions. In contrast, scaling the number of subdomains J does not produce the same rapid growth in conditioning, since hidden-layer basis functions are shared across subdomains. However, increasing either C or J eventually leads to effective rank deficiency.

When C is increased, additional basis functions become progressively redundant, introducing near-linear dependencies in $\mathbf{M}$. When J is increased, each subdomain becomes simpler, and the local solutions approach quasi-linear behaviour. In this regime, many basis functions are unnecessary for accurate approximation, again producing effective linear dependence. Thus, both width and localisation can induce rank saturation if capacity exceeds problem complexity.

One important limitation observed in this study is hidden-layer saturation in deeper architectures. As h increases, repeated application of bounded activation functions drives basis functions toward flat, near-linear responses. While FBPINN partially compensates for this effect through adaptive hidden weights, ELM-FBPINN is more sensitive due to its frozen representation. Addressing this limitation may require alternative activation functions, normalisation strategies, or architectural modifications.

Improving the conditioning of $\mathbf{M}$ remains another promising direction for future work. Reducing redundancy among basis functions, encouraging more informative random features, or incorporating regularisation strategies could mitigate ill-conditioning and enhance numerical robustness. If a greater proportion of basis functions contribute meaningfully to the approximation space, it may be possible to extend ELM-FBPINN to more complex and highly multiscale problems while preserving computational efficiency.

Additionally, our study is limited to the 1D and 2D problems presented, primarily due to hardware constraints. While there is no theoretical blocker to applying the method to higher dimensional problems, it was outside the scope of feasibility in this work. This means that all of the numerical results gathered are with the caveat that we do not have certainty in more difficult regimes. For higher dimensional problems,

ELM-FBPINN is already capable of being applied assuming the linearity of differential operators, and as with (FB)PINNs can easily be adapted for inverse problems. For non-linear PDEs, additional work is required in order to adapt ELM-FBPINN for non-linear operators. This would involve replacing the linear solver with an iterative linearization technique, such as Newton's method, which in turn would increase the computational cost, requiring some additional investigation into possible optimisation efficiency. Applying ELM-FBPINN to more complicated problems, in higher dimensions is an obvious and essential future work. Furthermore, it would be useful to evaluate the model more extensively in the L^2 or L^∞ norms, which aligns closely with the standard neural network literature. The L^1 norm was chosen following the prior work on FBPINN, and is a useful metric for evaluating the mean absolute error across the whole, reconstructed, domain.

In summary, the results of this chapter demonstrate that combining domain decomposition with an ELM-based linear least-squares formulation yields a highly efficient and competitive alternative to gradient-based FBPINNs. While adaptive hidden representations remain advantageous for complex high-dimensional problems, ELM-FBPINN offers substantial gains in convergence speed and competitive scalability. These findings suggest that random-feature-based domain decomposition provides a principled and promising direction for scalable scientific machine learning methods.

Chapter 4

Basis pre-training

4.1 Introduction

In Chapter 3, we found that the ELM-FBPINN method was a promising alternative to the original FBPINN approach, that replaced a costly gradient-based optimisation by a least-squares solve. Whilst the ELM-FBPINN method performed remarkably well on the oscillatory and multiscale test problems, the random basis functions used in ELM-FBPINN were not sufficient to accurately capture solutions of complex high-dimensional problems.

To remedy this shortcoming, in this chapter we replace the random weights and biases in ELM-FBPINN with ones that are tailored to the type of problem to be solved. Specifically, we perform a supervised learning stage on randomly generated exact solutions, referred to as the pre-trainer (PT), to produce a generic, expressive, and well-suited basis for a given problem class. The key idea is to improve the quality of the hidden-layer feature map before solving the physics-constrained least-squares system. The trained hidden weights are then transferred to ELM-FBPINN with the aim of improving the computational efficiency and precision, while alleviating the oversaturation of the hidden layer basis functions. This pre-trained ELM-FBPINN (PT-ELM-FBPINN) is then compared to FBPINN and standard ELM-FBPINN, which allows us to examine the effect of pre-training the normally random, frozen hidden layer weights of ELM-FBPINN. Additional studies focus exclusively on the pre-training stage and

the resulting PT-ELM-FBPINN. We evaluate each model on the same problems as described in Chapter 3, with minor updates to accommodate the random sampling schedule employed during pre-training.

This chapter is structured as follows. In Section 4.2, we formally define the supervised pre-training procedure and the associated loss function. Subsequent sections investigate how the pre-trained basis affects conditioning, saturation behaviour, and scalability with respect to frequency and dimensionality. Section 4.3 redefines the problems studies, focusing on the relevant pre-training sampling strategies. Section 4.4.1 presents the results for the 1D damped harmonic oscillator, and Section 4.4.2 presents the 2D multiscale Laplace results. Finally, a discussion of the results and conclusions are shown in Section 4.5.

We look to address the following research questions: can we improve the training efficiency and precision of ELM-FBPINN by replacing the random, frozen weights with a pre-trained basis? What effect does the pre-trained basis have on the conditioning of the matrix $\mathbf{M}$? Does a pre-trained basis still exhibit the same saturation seen in standard ELM-FBPINN when increasing the depth of the network? Can we train on a low frequency problem and improve the ELM-FBPINN solve of a higher frequency problem?

4.2 Pre-Training Definition

In this work, we replace the random, frozen hidden-layer weights of ELM-FBPINN with pre-trained weights. Specifically, by training on a number of randomly sampled exact solutions, we aim to learn a general function representation for the hidden-layer basis functions of the ELM-FBPINN. Training across multiple solutions yields a basis that is generic and representative of the problem class. In contrast to standard ELM initialisation, where hidden weights are drawn once from a fixed distribution and never updated, the pre-training stage optimises these weights in a supervised manner. The key idea is to enhance ELM-FBPINN, retaining the computational efficiency of the linear least-squares solve, while improving the expressivity of the hidden-layer basis.

We first describe the supervised network architecture used during pre-training, before explaining how the learned parameters are transferred to the ELM-FBPINN framework.

Let $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$ be the input to the network. Then the hidden layer-wise activations, up to layer L, are defined as:

$$\boldsymbol{\phi}(\mathbf{x}, \boldsymbol{\theta}^*) = \sigma\Big(\mathbf{W}^{(L)}\,\sigma\Big(\mathbf{W}^{(L-1)}\,\cdots\,\sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(L-1)}\Big) + \mathbf{b}^{(L)}\Big), \tag{4.1}$$

where $\boldsymbol{\theta}^* = \{\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}\}_{l=1}^{L}$ are the network weights for each hidden layer, ℓ. Here, $\mathbf{W}^{(\ell)} \in \mathbb{R}^{C_\ell \times C_{\ell-1}}$ and $\mathbf{b}^{(\ell)} \in \mathbb{R}^{C_\ell}$ are the weight matrix and bias vector for hidden layer ℓ, where C_ℓ denotes the number of neurons in layer ℓ, for $\ell = 1, \ldots, L$. Note that we have introduced $\boldsymbol{\theta}^*$ to delineate it from the hidden-layer weights of ELM-FBPINN, $\boldsymbol{\theta}$. Throughout this chapter, $\boldsymbol{\theta}^*$ therefore denotes pre-trained parameters, while $\boldsymbol{\theta}$ refers to randomly initialised ELM parameters. The final output layer is a linear transformation, which results in the approximated solution, $\mathbf{u}^*(\mathbf{x}, \boldsymbol{\theta}^*)$, given by:

$$\mathbf{u}^*(\mathbf{x}, \boldsymbol{\theta^*}) = \sum_{c=1}^{C_L} \mathbf{a}_c^*\, \boldsymbol{\phi}_c(\mathbf{x}, \boldsymbol{\theta^*}) + \mathbf{b}_{j,c}^*, \quad \mathbf{u}^*(\mathbf{x}, \boldsymbol{\theta^*}) \in \mathbb{R}^{d_{\text{out}}}, \tag{4.2}$$

with $\mathbf{a}_c^* = [a_1^*, \ldots, a_{C_L}^*] \in \mathbb{R}^{d_{\text{out}} \times C_L}$ and $\mathbf{b}_{(j,c)}^* \in \mathbb{R}^{d_{\text{out}}}$.

In this supervised stage, both the hidden parameters $\boldsymbol{\theta}^*$ and the output weights $\mathbf{a}^*$ are optimised jointly using gradient-based training.

Let $\mathbf{u}_{\text{exact}}(\mathbf{x})$ be an exact solution to some PDE. Then, the network can be trained by optimising the following supervised loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \big(\mathbf{u}^*(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{u}_{\text{exact}}(\mathbf{x}_i)\big)^2, \tag{4.3}$$

where N is the total number of input points. This corresponds to a standard mean-squared error loss over collocation points sampled from the solution domain. Note that this supervised loss construction does not prevent the use of FBPINN domain decomposition, with the same partition of unity and local subdomain network structure as described in Section 2.5. As such, the supervised learning approximation becomes:

$$\mathbf{u}^*(\mathbf{x}, \boldsymbol{\theta}^*) = \sum_{j=1}^{J} \omega_j(\mathbf{x}) \sum_{c=1}^{C_L} \mathbf{a}_{j,c}^* \, \boldsymbol{\phi}_c(\mathbf{x}, \boldsymbol{\theta}^*) + \mathbf{b}_{j,c}^*, \tag{4.4}$$

where $\boldsymbol{\theta}^*$ are the pre-train network hidden layer weights.

To create a more general basis, we extend this supervised learning regime to training on multiple solutions to related PDEs, which involve the same differential operator but have different initial conditions, boundary conditions and/or forcing terms. The solution to each PDE has its own output weights, but shares the same set of hidden parameters $\boldsymbol{\theta}^*$.

For each exact solution $p \in \{1, \ldots, P\}$, we can define unique output linear weights:

$$\mathbf{a}_{j,c}^{(p)} \in \mathbb{R}^{d_{\text{out}} \times C}, \quad \mathbf{b}_{i,j}^{(p)} \in \mathbb{R}^{d_{\text{out}}}. \tag{4.5}$$

Thus, the hidden representation is shared across all problems in the class, while the final linear layer adapts to each specific instance.

For each initial condition p, the solution is defined as:

$$\tilde{\mathbf{u}}(\mathbf{x}, \mathbf{a}^{(p)}, \boldsymbol{\theta}^*) = \omega_j(\mathbf{x}) \sum_{c=1}^{C} (\mathbf{a}_{j,c}^{(p)})^T \cdot \boldsymbol{\phi}(\mathbf{x}, \boldsymbol{\theta}^*) + \mathbf{b}_{j,c}^{(p)}. \tag{4.6}$$

Summing (4.6) over a set of N collocation points, the supervised loss for each solution, p is then:

$$\mathcal{L}^{(p)} = \frac{1}{N} \sum_{i=1}^{N} \left(\tilde{\mathbf{u}}^{(p)}(\mathbf{x}_i) - \mathbf{u}_{\text{exact}}^{(p)}(\mathbf{x}_i) \right)^2. \tag{4.7}$$

We then take the average error across all exact solutions such that:

$$\mathcal{L}_{PT} = \frac{1}{P} \sum_{p=1}^{P} \mathcal{L}^{(p)}. \tag{4.8}$$

This ensures that the hidden-layer parameters are optimised to minimise the average approximation error across the entire solution family. The full PT network is trained by optimising $\boldsymbol{\theta}^*$ representing a generic set of weights for the problem class.

After training, the hidden-layer parameters $\boldsymbol{\theta}^*$ are frozen and transferred to the

ELM-FBPINN framework. The subsequent physics-informed solve then proceeds exactly as in Chapter 3, except that the hidden basis functions are no longer random but pre-trained.

Two-Stage Workflow Overview. The full PT-ELM-FBPINN procedure consists of two distinct stages.

- *Stage 1 (Pre-Training):* a supervised neural network is trained using gradient-based optimisation on a family of exact solutions belonging to the same PDE class. During this stage, both hidden-layer parameters, $\boldsymbol{\theta}^*$, and output weights, $\mathbf{a}$, are optimised to minimise the mean-squared approximation error across multiple problem instances. The outcome is a set of expressive, shared hidden-layer basis functions tailored to the operator and solution structure of the problem class.
- *Stage 2 (Physics Solve):* the learned hidden parameters, $\boldsymbol{\theta}^*$, are frozen and transferred to the ELM-FBPINN framework. For a new PDE instance, only the output-layer coefficients are determined by solving the linear least-squares system derived from the physics-informed loss. No further gradient-based optimisation is required.

This separation cleanly decouples representation learning (Stage 1) from physics enforcement (Stage 2), preserving the computational efficiency of the ELM formulation while improving the quality of the underlying basis.

4.3 Problem Definition

To assess the effects the pre-training of hidden-layer basis functions has, we compare PT-ELM-FBPINN to a standard ELM-FBPINN. This allows for a direct assessment of how the pre-trained weights affect the performance of the model. We also compare against FBPINN to better understand how the performance of ELM-FBPINN changes in comparison to the gradient-descent-based solver. The two problems studied are the same as in Chapter 3: a 1D damped harmonic oscillator and a 2D multi-scale Laplace problem. These test cases are repeated to maintain consistency with prior

behaviours observed in ELM-FBPINN, thereby isolating the impact of pre-training. The description of each problem remains unchanged from Chapter 3, however, the sampling strategy for generating exact solutions will be detailed for both the 1D and 2D problems. These sampling strategies have been designed to create solutions with similar profiles to the test solution, which is excluded from the pre-training set.

By keeping the governing equations and numerical configurations identical to Chapter 3, any observed performance differences can be attributed solely to the introduction of the pre-trained basis. The only modification introduced in this chapter is therefore the construction of the supervised training dataset used in Stage 1 of the PT workflow.

Note, as with ELM-FBPINN, PT-ELM-FBPINN is able to handle higher dimensional problems with no additional alterations necessary. However, this is outside the scope of this research and is left as a future direction for evaluating performance of the model.

4.3.1 1D Damped Harmonic Oscillator

We again consider the under-damped regime of the harmonic oscillator ODE described in (3.4). Given this consideration, the same general exact solution as described in (3.6) is used, but now the initial conditions u_0 and v_0 are sampled. These represent the displacement and velocity respectively and control the initial position and amplitude of the oscillations. We fix the test solution to be the same as in Chapter 3, with $u_0 = 1$ and $v_0 = 0$.

The goal of the sampling procedure is to expose the pre-training network to a diverse yet structured family of oscillatory trajectories, while ensuring that the held-out test configuration remains representative of the training distribution.

The displacement, u_0 is sampled from a bimodal normal distribution:

$$u_0 \sim p\mathcal{N}(\mu_1, \sigma_1^2) + (1-p)\mathcal{N}(\mu_2, \sigma_2^2), \tag{4.9}$$

where p is the probability of sampling the first mode, while $\mu_{1,2}$ and $\sigma_{1,2}^2$ denote the means and variances of each mode, respectively. For all of our tests, we set the minimum

value of u_0 to $u_{\min} = -1.5$ and the maximum value $u_{\max} = 1.5$. The mean values are chosen as $\mu_1 = -1$ and $\mu_2 = 1$, which ensures the PT sees training solutions with both negative and positive initial displacements. This bimodal construction prevents the network from biasing toward a single displacement regime and encourages symmetry-aware feature learning in the hidden layer.

The standard deviations are set to

$$\sigma_1 = \frac{u_{\min} - \mu_1}{3},\ \sigma_2 = \frac{u_{\max} - \mu_2}{3},$$

which ensures three standard deviations fall within the interval $[u_{\min}, u_{\max}]$. This truncation ensures that extreme outliers are avoided, keeping the training distribution well-controlled while still covering a broad amplitude range.

The initial velocity, v_0, is sampled from a normal distribution:

$$v_0 \sim \mathcal{N}(\mu_v, \sigma_v^2), \tag{4.10}$$

where μ_v and σ_v^2 denote the mean and variance, respectively. The minimum and maximum values are set to $v_{\min} = -5$ and $v_{\max} = 5$, with a mean of $\mu_v = 0$, which results in a minor phase shift of the oscillations. Sampling v_0 primarily affects the phase of the oscillation while leaving the damping rate unchanged, thereby enriching the training set without altering the qualitative structure of the solution manifold. This phase shift is subtle, ensuring that the hidden-layer weights are representative of our test solution when $v_0 = 0$, while still introducing sufficient variability in the solutions, allowing the PT to learn a flexible basis, well fit for the problem class. Negative values of v_0 correspond to initial motion of the oscillator in the opposing direction. The standard deviation is:

$$\sigma_v = \frac{v_{\max} - v_{\min}}{6},$$

again ensuring that three standard deviations lie within the interval $[v_{\min}, v_{\max}]$.

Together, the joint sampling of (u_0, v_0) defines a two-parameter family of trajectories spanning variations in amplitude and phase, while preserving the same damping and

frequency regime as the test case.

Figure 4.1 shows the distributions for 250 random samples. Figure 4.4 shows representative solutions along with the pre-trainer approximations.

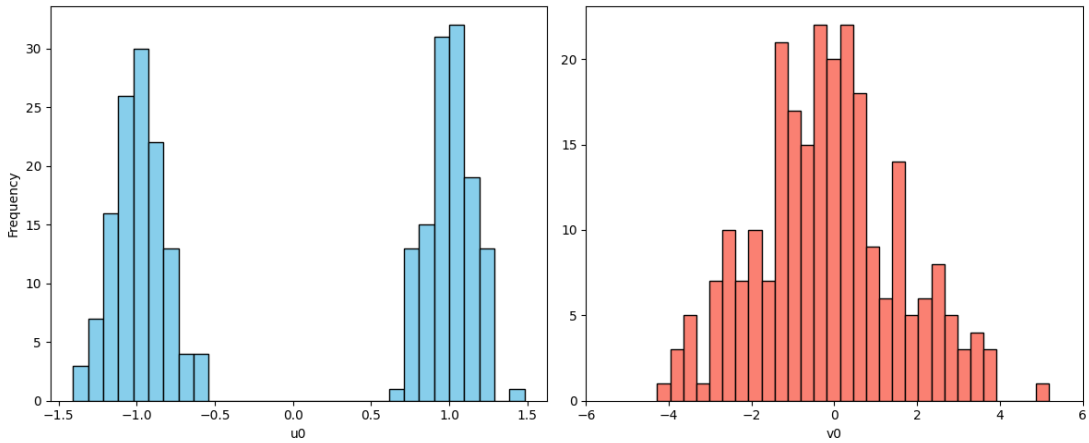


Figure 4.1: Histograms showing the distribution of 250 randomly sampled initial conditions for the 1D damped harmonic oscillator. Left: Bimodal normal distribution for displacement, u_0. Right: Normal distribution for velocity, v_0.

4.3.2 2D Multi-scale Laplace

For our 2D tests, we introduce scaling coefficients to the exact solution and source terms described in (3.13) and (3.12), respectively. These scaling coefficients are assigned to each frequency component, ω_i, altering the generated solutions by modulating the amplitude of frequency contributions. Unlike the 1D case, where variability arises from initial conditions, here the diversity of the solution family is generated directly in spectral space through amplitude modulation of each Fourier mode.

We sample the coefficients from a uniform distribution:

$$c_i \sim \mathcal{U}(c_{\min}, c_{\max}), \qquad i = 1, \ldots, n_\omega, \tag{4.11}$$

where $c_{\min} = -2$ and $c_{\max} = 2$. This choice of limits for the distribution allows for a broad range of frequency contributions, including cases where $c_i \approx 0$, which can strongly amplify or suppress frequencies, and cases where c_i is negative, which corresponds to a change of sign for a corresponding frequency component. This results in a diverse set

of solutions, and forces the PT to learn a general, expressive basis. Uniform sampling also ensures that no single frequency component dominates the pre-training process and encourages the hidden-layer basis to represent interactions between multiple scales. Importantly, the test solution corresponds to a fixed set of coefficients and is excluded from the sampling pool, ensuring that evaluation measures true generalisation rather than memorisation.

We can now redefine the source term as:

$$f(x_1, x_2) = \frac{2}{n_\omega} \sum_{i=1}^{n_\omega} c_i (\omega_i \pi)^2 w \sin(\omega_i \pi x_1) \sin(\omega_i \pi x_2). \tag{4.12}$$

The corresponding exact solution is

$$u(x_1, x_2) = \frac{1}{n_\omega} \sum_{i=1}^{n_\omega} c_i (\sin(\omega_i \pi x_1) \sin(\omega_i \pi x_2)). \tag{4.13}$$

Each sampled coefficient vector $\{c_i\}_{i=1}^{n_\omega}$ therefore defines a new PDE instance sharing the same operator but differing in spectral composition.

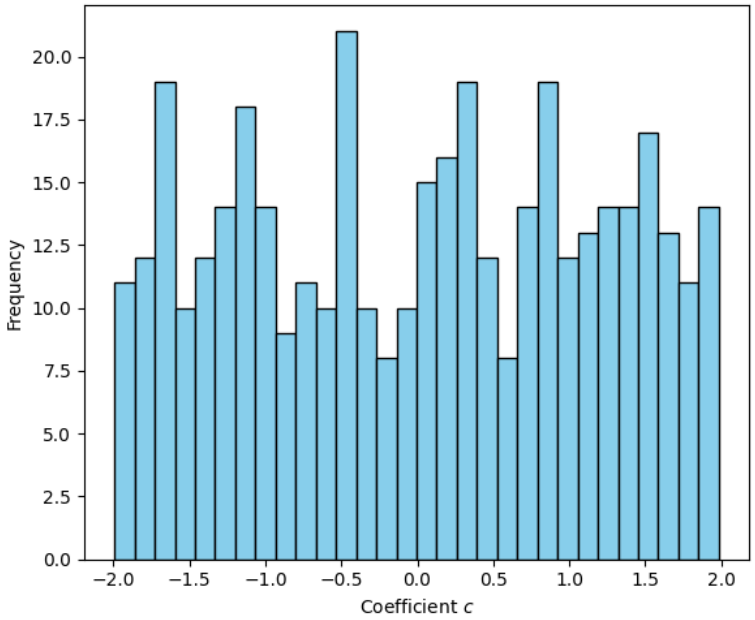


Figure 4.2: Histogram of the uniform distribution used to generate the scaling coefficients, c, for the 2D multi-scale Laplace problem. Here the number of solutions has been set to 100.

We can now generate an arbitrary number of scaling coefficient lists, each with n_ω terms. Figure 4.2 shows a histogram of the uniform distribution of c, with the number of solutions set to 100. Figure 4.39 shows representative solutions along with the pre-trainer approximations. In practice, this allows the size of the pre-training dataset to be scaled independently of the final physics-informed solve, providing flexibility in controlling representation learning without increasing Stage 2 computational cost.

4.3.3 Common Implementation

To ensure a fair comparison between the models, we again fix a number of settings across all experiments, which also allow us to better understand the effect of chosen variables. These settings remain unchanged, unless explicitly stated, and are consistent across both problems. In particular, we isolate the impact of pre-training by keeping the domain decomposition, activation functions, optimisation budgets, and evaluation metrics identical to those used in Chapter 3.

The POU and window functions remain unchanged from (3.16) for all models, including the PT. As default, we ensure the solution ansatz of PT and PT-ELM-FBPINN are aligned, so that the basis functions are well adapted for the subdomain frequencies. This alignment ensures that the representation learned during pre-training is structurally compatible with the physics-informed least-squares formulation used in Stage 2. In other words, the PT does not learn a global representation that is later reshaped, but directly learns basis functions in the same localised partition-of-unity framework.

We employ fully connected neural networks with tanh activation functions for all models. The PT utilises a shared hidden-layer basis across the subdomains, with C number of basis functions. We initialise the linear output weights independently for each subdomain, for each randomly generated pre-training exact solution. Similarly, ELM-FBPINN and PT-ELM-FBPINN utilise the same structure of a shared hidden-layer basis, and independent linear output weights per subdomain. As the ELM models are evaluated on a single test solution, we only require a single set of linear output weights. The FBPINN differs by using full networks per subdomain, each trained in-

dependently. All models are initialised using uniform LeCun initialisation, with the ELM-FBPINN hidden-layer weights being frozen, and the PT-ELM-FBPINN initialising the linear output weights only. The key distinction between methods is therefore as follows: FBPINN learns both hidden and output weights independently per subdomain; ELM-FBPINN freezes randomly initialised hidden weights and solves only for output coefficients; PT-ELM-FBPINN freezes pre-trained hidden weights and solves only for output coefficients. This allows us to directly assess whether learning a better basis compensates for removing gradient-based optimisation in Stage 2.

Both PT and FBPINN use the Adam optimiser with a learning rate of 10^{-3}. Both ELM models employ the same iterative LSQR sparse solver from Scipy. The default number of iterations for all models is set to 15,000. Unless otherwise indicated we allow the solvers to run the full number of iterations to directly compare convergence behaviour between the models. Using an identical iteration budget ensures that improvements in accuracy or speed are attributable to architectural differences rather than training duration. Where early stopping is introduced in later experiments, this will be explicitly stated.

For the 1D damped harmonic oscillator, we evaluate FBPINN, ELM-FBPINN, and PT-ELM-FBPINN on a test solution when $u_0 = 1$ and $v_0 = 0$. The solution frequency is fixed at $w_0 = 80$, which results in a solution with high frequency oscillations and a slowly decaying amplitude. This configuration mirrors the challenging baseline scenario from Chapter 3, ensuring that any gains from pre-training are tested in a high-frequency regime where spectral bias is known to be significant. For the 2D multi-scale Laplace problem, we fix the number of frequency components of the test solution to $n_\omega = 4$, with $\omega_i = 4i$. The scaling coefficients are also set to 1 for all models other than the PT. For the PT, we will define the training parameters in Section 4.4. Thus, while the PT is exposed to a diverse family of coefficient combinations, the evaluation is performed on a canonical instance with equal spectral weighting, allowing generalisation performance to be clearly quantified.

We again evaluate all models using the same e_{L^1} error as defined in Equation (3.19), with the same uniform sampling strategy that ensures sufficient resolutions per subdo-

main. Each experiment is repeated using six different random seeds. Reported errors correspond to the mean error, $\hat{e}_{L^1}$, with error bars defined by the difference between the best and worst observed values. Repeating experiments across multiple random seeds allows us to assess robustness with respect to weight initialisation and sampling variability, which is particularly important for ELM-based methods where randomness enters explicitly through hidden-layer initialisation.

All models are implemented in Python using JAX. The ELM-FBPINN implementation builds upon the open-source FBPINN codebase available at `github.com/benmoseley/FBPINNs`. Experiments are run on either an AMD Ryzen 5 7600X CPU with 32 GB of RAM or an NVIDIA RTX 5070 Ti GPU with 16 GB of VRAM. Due to current incompatibilities between SciPy and JAX, the linear systems are solved on the CPU; however, a JAX-native LSQR implementation would allow the ELM-FBPINN to run fully on the GPU.

4.4 Numerical Results

In this section, we present numerical results which demonstrate the performance of PT-ELM-FBPINN and compare it against FBPINN and ELM-FBPINN. We focus on convergence behaviour, solution accuracy, model stability, and computational efficiency. Sections 4.4.1 and 4.4.2 present results obtained for the 1D damped harmonic oscillator and the 2D multi-scale Laplace problem, respectively. Each section begins with a baseline configuration, followed by a series of ablation studies. These ablations are designed to isolate the effects of frequency content, network capacity, and domain decomposition parameters. Additional analysis is performed on the shape of the PT basis functions, and their effect on the linear least-squares solve of PT-ELM-FBPINN. The section concludes with a brief discussion summarising the key findings.

4.4.1 1D Damped Harmonic Oscillator

Baseline

We first evaluate a baseline set of results for each model. For the PT, we fix the number of solutions at $P = 250$, which is chosen to ensure sufficient training examples to create a general representation of the solution space. The remaining models are trained and tested on the same test problem as described in Section 4.3.3. All models, including PT, are initialised with a single hidden layer of $C = 12$ basis functions, with a domain decomposition comprising of $J = 20$ subdomains, each with a fixed width factor of $\eta = 2.9$. Each model is trained on $(C + 1)J$ collocation points, which is chosen to provide sufficient data to each subdomain, improving the models' ability to learn the global solution. This collocation sampling also ensures that the ELM models' matrices in the linear least-squares formulation are over-determined, guaranteeing a unique least-squares solution. All models are tested on $50 \times J$ points, which ensures adequate coverage of the domain, producing smooth approximate solutions.

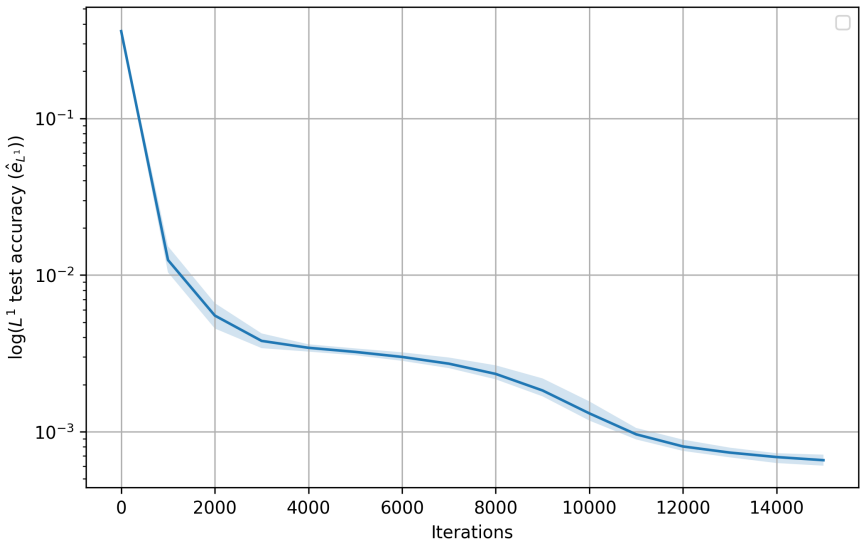


Figure 4.3: Mean error, $\hat{e}_{L^1}$, vs Iterations for PT for the 1D harmonic oscillator and the baseline configuration.

Figure 4.3 shows the convergence for the PT. Despite training on $P = 250$ exact

solutions, the model has achieved an acceptable error and has fit the solutions well (see Figure 4.4 for example predictions). The speed of convergence is initially rapid during the first $\approx$ 2000 iterations, before a period of relatively little change and a second descent during the final $\approx$ 7000 iterations. This trend is smooth and indicates a relative stability in the training evolution of the PT model on the 1D problem.

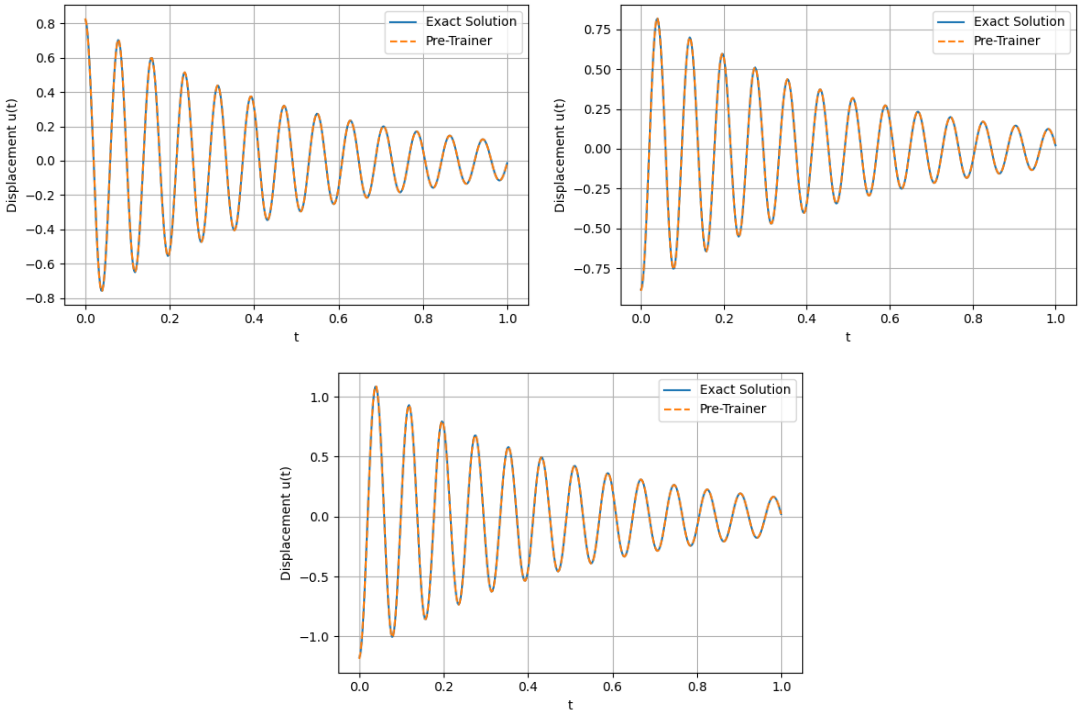


Figure 4.4: PT predictions versus the exact solution for the 1D damped harmonic oscillator. Top left: $u_0 = 0.82$, $v_0 = -2.01$; Top right: $u_0 = -0.89$, $v_0 = -1.14$; Bottom: $u_0 = -1.18$, $v_0 = 2.62$.

Figure 4.5 shows the convergence for FBPINN, ELM-FBPINN and PT-ELM-FBPINN. Both ELM models converge faster than FBPINN, which has a relatively steady, slower convergence overall. There is a possible indication that FBPINN is not fully converged within the 15,000 iterations, however, the convergence curve appears to be approaching a plateau by the end of the runs. The best performing model in terms of both convergence speed and final precision is PT-ELM-FBPINN, showing the advantages that can be gained from a more expressive basis.

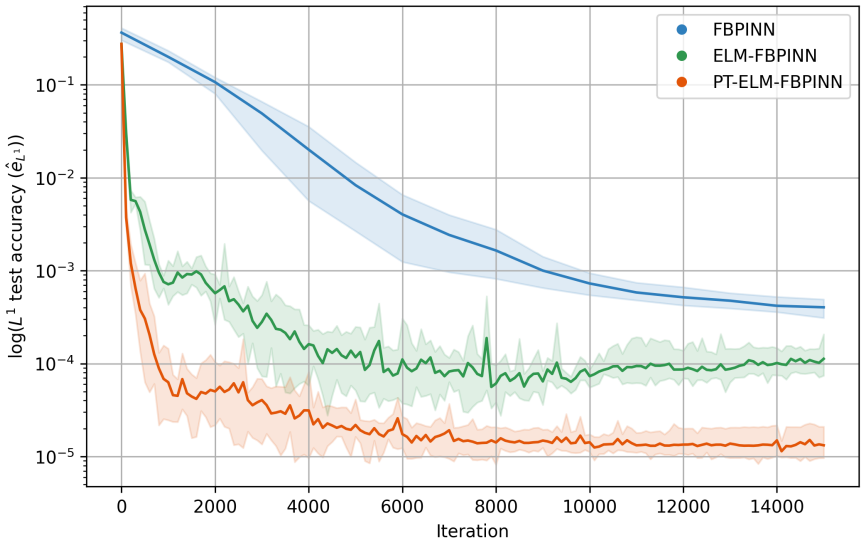


Figure 4.5: Convergence curves for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for the 1D harmonic oscillator with the baseline configuration.

Table 4.1 shows the final $\hat{e}_{L^1}$ and average times taken to complete the runs for each model. Note that the PT results are based on the pre-training of the $P = 250$ training solutions, and so it is not meaningful to compare them directly to the other models. The PT results are included in this table (and all others) for convenience. The PT row therefore reflects supervised approximation accuracy over the sampled training distribution, whereas the remaining rows correspond to physics-informed solves of a single held-out test instance.

FBPINN and ELM-FBPINN have achieved errors within the same order of magnitude, demonstrating that both are well suited to the problem. However, PT-ELM-FBPINN has achieved an error one order of magnitude lower than the others. This improvement indicates that learning a tailored hidden-layer representation prior to the least-squares solve significantly enhances the expressive efficiency of the ELM formulation.

While the PT-ELM-FBPINN appears to have resolved the runs on average faster than any other model, it should be noted that the timing of ELM-FBPINN includes the initialisation of weights. This initialisation does not occur when pre-trained weights are transferred, and so in practice the ELM model's time to solve is assumed to be

negligibly different. Additionally, the FBPINN time includes the $\approx 1-2$ seconds of start up compilation.

Model	$\hat{e}_{L^1}$	$\hat{e}_{L^2}$	$\hat{e}_{L^\infty}$	Time (s)
PT	(6.58±1.07)e-04	(1.32±0.11)e-03	(1.39±0.11)e-03	22.3±0.069
FBPINN	(4.03±1.81)e-04	(4.93±1.90)e-04	(5.70±1.90)e-04	14.8±0.0634
ELM-FBPINN	(1.13±1.33)e-04	(1.35±1.49)e-04	(2.38±1.50)e-04	12.2±0.0558
PT-ELM-FBPINN	(1.32±1.13)e-05	(1.64±1.31)e-05	(2.51±1.31)e-05	11.4±0.185

Table 4.1: Baseline results for the 1D damped harmonic oscillator. Reported $\hat{e}_{L^1}$, $\hat{e}_{L^2}$ and $\hat{e}_{L^\infty}$ values are mean ± range over random seeds, while time values are mean ± 1 standard deviation.

We compare the SVD condition number and rank of the ELM models' matrices, $\mathbf{M}$. The exact values are reported in Table 4.2. The baseline results show that while the condition number is minimally affected by transferring the pre-trained weights, the rank is notably improved. This observation suggests that the pre-training stage primarily enhances the diversity of the column space of $\mathbf{M}$, rather than significantly altering its spectral spread.

The increase in the rank shows that the pre-trained weights result in basis functions that are further from being linearly dependent and have more relevant information contributions than the randomly frozen weights of ELM-FBPINN. In practical terms, a higher numerical rank implies that fewer columns of $\mathbf{M}$ lie in near-null directions, reducing redundancy among basis functions and improving the effective conditioning of the least-squares solve.

The hidden-layer basis functions for ELM-FBPINN and PT-ELM-FBPINN are plotted in Figure 4.6. Although the difference is somewhat subtle for this configuration, there is a visible decrease in the number of quasi-linear basis functions in the PT-ELM-FBPINN basis functions. There is also an increase in the number of basis functions with sharper gradients, which visibly fit the subdomain solution better than the flatter basis functions of ELM-FBPINN. This qualitative difference aligns with the quantitative rank improvement: sharper and more localised basis functions contribute additional independent directions in function space, enabling the linear solver to capture oscillatory structure more efficiently.

Model	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)
ELM-FBPINN	1.30e+18	216
PT-ELM-FBPINN	1.69e+18	254

Table 4.2: Condition number $\kappa(\mathbf{M})$ and numerical rank of the ELM-FBPINN and PT-ELM-FBPINN system matrices for the baseline configuration.

Overall, these results indicate that pre-training does not merely accelerate convergence, but fundamentally alters the structure of the hidden representation in a way that improves both expressivity and numerical robustness.

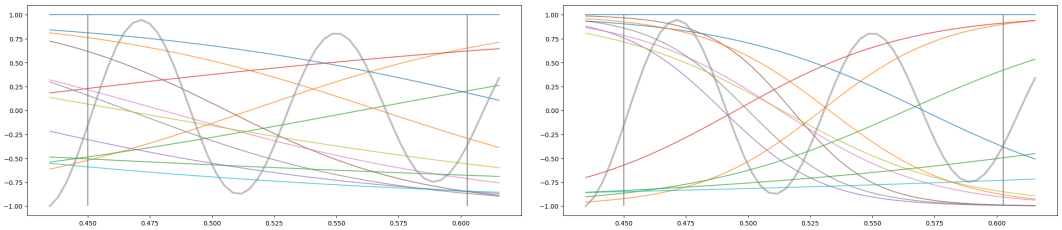


Figure 4.6: Hidden-layer basis functions for baseline configurations. Left: ELM-FBPINN; Right: PT-ELM-FBPINN.

Pre-training Iterations

We now fix all parameters, and investigate the effect of PT iterations on the performance of PT-ELM-FBPINN. Specifically, we run the baseline PT model for 15,000 iterations, saving parameters every 1,000 epochs. This effectively creates 15 separate PT models from which we can load hidden-layer weights in PT-ELM-FBPINN. By doing so, we obtain a controlled sequence of progressively refined hidden-layer representations, ranging from minimally trained to fully converged pre-training states.

This study allows us to observe what, if any, effect pre-training has on the linear least-squares solve as the basis functions are better fit to the pre-training data set of solutions. For smaller PT iterations, the weights are not well fit to the $P = 250$ solutions, but have still been trained partially into the problem space. The expectation is that these basis functions are more generic, but less tailored basis for the test problem. For larger PT iterations, the weights are well fit to the solutions, providing a slightly

less generic basis, but one that is well tailored to the test problem. This experiment therefore probes the trade-off between generic representation learning and problem-specific adaptation, and assesses how sensitive the downstream least-squares solve is to the maturity of the learned basis.

As this study pertains only to the PT iterations, we do not include FBPINN in this test, and only compare to an ELM-FBPINN model with baseline configuration. The baseline ELM-FBPINN therefore serves as a reference corresponding to zero pre-training iterations.

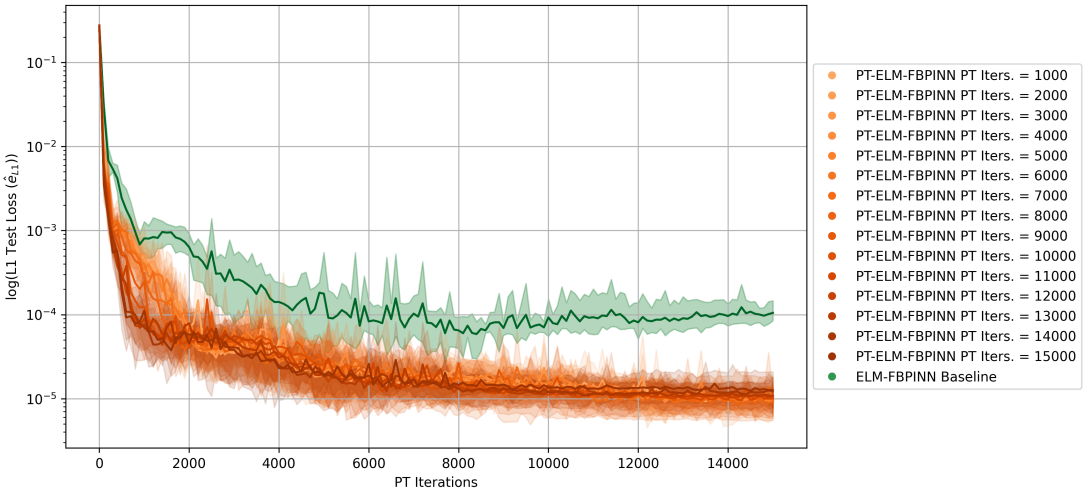


Figure 4.7: Convergence curves for PT-ELM-FBPINN for varying PT iterations. A baseline ELM-FBPINN convergence curve is also plotted.

The convergence of each PT-ELM-FBPINN model and the baseline ELM-FBPINN is plotted in Figure 4.7. There is a general trend of increasing convergence speed with increasing PT iterations. In particular, the initial slope of the convergence curves becomes progressively steeper as the number of PT iterations increases, indicating that a better-conditioned and more expressive basis accelerates the reduction of the residual. However, every PT-ELM-FBPINN model outperforms the ELM-FBPINN baseline in terms of both convergence speed and final $\hat{e}_{L^1}$ error. This demonstrates that just a small amount of pre-training can have a substantial increase in performance over the randomly frozen weights of ELM-FBPINN.

Remarkably, even the earliest saved checkpoints—corresponding to only 1,000 PT it-

erations—already produce a measurable improvement, suggesting that the hidden-layer representation quickly moves into a more favourable region of function space. This is further illustrated in Figure 4.8 and Table 4.3, where every PT-ELM-FBPINN model has a lower error, with the best models being two order of magnitudes smaller. The monotonic improvement in convergence behaviour reinforces the interpretation that pre-training enhances the linear independence and task-relevance of the basis functions. This demonstrates excellent robustness of the pre-trained weights and biases with respect to the number of iterations used in pre-training. We note that there is a very slight increase in the error as the number of PT iterations increases, but the impact of this is negligible and all solutions have an error on the order of 1e-05.

The slight non-monotonic variation in the final error for larger PT iteration counts may reflect mild over-specialisation of the basis to the sampled training set; however, the magnitude of this effect is negligible in practice. Overall, the results indicate that the PT-ELM-FBPINN framework is not highly sensitive to the exact stopping point of pre-training, provided that a modest level of representation learning has occurred.

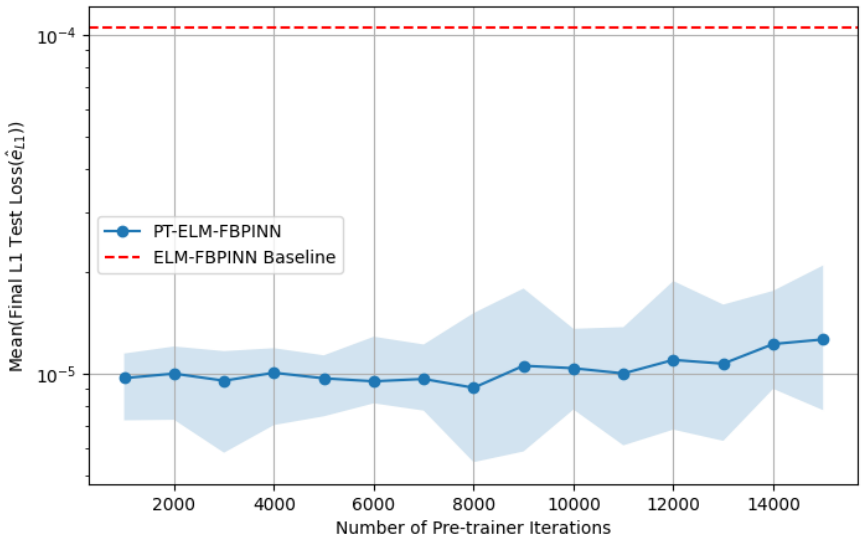


Figure 4.8: Final $\hat{e}_{L^1}$ values for PT-ELM-FBPINN as the number of PT iterations are varied. A baseline ELM-FBPINN result has been included for comparison.

The SVD-based condition number and matrix rank are shown in Figure 4.13, with the corresponding numerical values reported in Table 4.4. The condition number re-

PT Iterations	Final $\hat{e}_{L^1}$
1000	(9.69±4.18)e-06
2000	(9.99±4.71)e-06
3000	(9.52±5.8)e-06
4000	(1.01±0.483)e-05
5000	(9.68±3.84)e-06
6000	(9.48±4.67)e-06
7000	(9.63±4.41)e-06
8000	(9.07±9.6)e-06
9000	(1.05±1.19)e-05
10000	(1.04±0.572)e-05
11000	(1±0.756)e-05
12000	(1.1±1.19)e-05
13000	(1.07±0.964)e-05
14000	(1.22±0.853)e-05
15000	(1.26±1.3)e-05
ELM-FBPINN	(1.05±0.617)e-04

Table 4.3: Final $\hat{e}_{L^1}$ values for varying PT iterations for PT-ELM-FBPINN. A baseline ELM-FBPINN result has also been included for comparison.

mains relatively stable and comparable to the ELM-FBPINN baseline, indicating that pre-training the hidden-layer weights does not adversely affect the resulting numerical stability of the linear least-squares system. The PT-ELM-FBPINN matrices consistently exhibit a higher rank, suggesting increased linear independence among the basis functions. Moreover, the matrix has 260 columns, demonstrating that pre-trained weights yield matrices that are close to full rank. This implies that even with a relatively small number of pre-training iterations, the learned basis becomes more expressive and effectively mitigates redundant basis functions in the matrix $\mathbf{M}$.

Effect Of Pre-Training Solutions

Next, we will investigate the effect of the number of solutions, P, that the PT model is trained on. The intuition is training on more solutions achieves a more general representation of the problem class. It should also be expected that the supervised learning schedule is susceptible to overfitting if P is very small, which would result in performance similar to randomly initialised weights. This experiment therefore examines how much diversity is required in the pre-training dataset before the learned

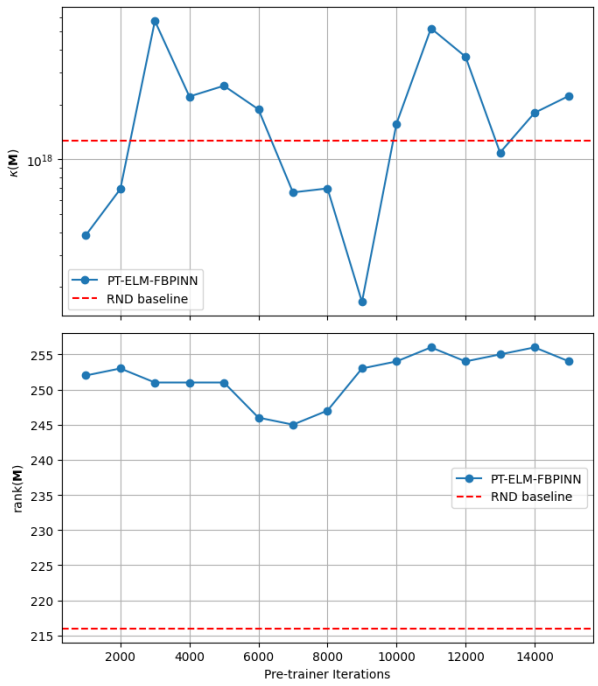


Figure 4.9: Condition number, $\kappa(\mathbf{M})$ and rank($\mathbf{M}$) vs number of PT iterations.

PT Iteration	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)
1000	3.85e+17	252
2000	6.89e+17	253
3000	5.73e+18	251
4000	2.21e+18	251
5000	2.53e+18	251
6000	1.88e+18	246
7000	6.60e+17	245
8000	6.94e+17	247
9000	1.66e+17	253
10000	1.57e+18	254
11000	5.22e+18	256
12000	3.65e+18	254
13000	1.09e+18	255
14000	1.80e+18	256
15000	2.23e+18	254
ELM-FBPINN	1.26e+18	216

Table 4.4: SVD-based condition number and matrix rank for PT-ELM-FBPINN as a function of PT iterations. A baseline ELM-FBPINN result has also been included for comparison.

hidden-layer basis meaningfully departs from random initialisation. In particular, it tests whether the gains observed in the baseline configuration are driven by broad coverage of the solution manifold or whether even limited exposure is sufficient to shape a useful representation.

The convergence of each PT model is plotted in Figure 4.10. We see very little variation in the convergence profiles as P varies. In other words, the supervised loss during pre-training decreases at a similar rate regardless of the number of sampled solutions, suggesting that the optimisation landscape is not significantly more difficult when P increases within the tested range.

This consistency in convergence with different values of P is carried into the PT-ELM-FBPINN results, as seen in Figure 4.11. This indicates that PT-ELM-FBPINN is very robust with respect to the number of pre-training solutions. Notably, even for relatively small values of P, the PT-ELM-FBPINN maintains a clear performance advantage over the baseline ELM-FBPINN. The absence of significant degradation at

lower P also indicates that severe overfitting does not occur within the tested regime.

Overall, these results reinforce the interpretation that pre-training acts primarily as a mechanism for shaping the functional basis toward the operator structure, and that only moderate dataset sizes are required to obtain substantial downstream improvements.

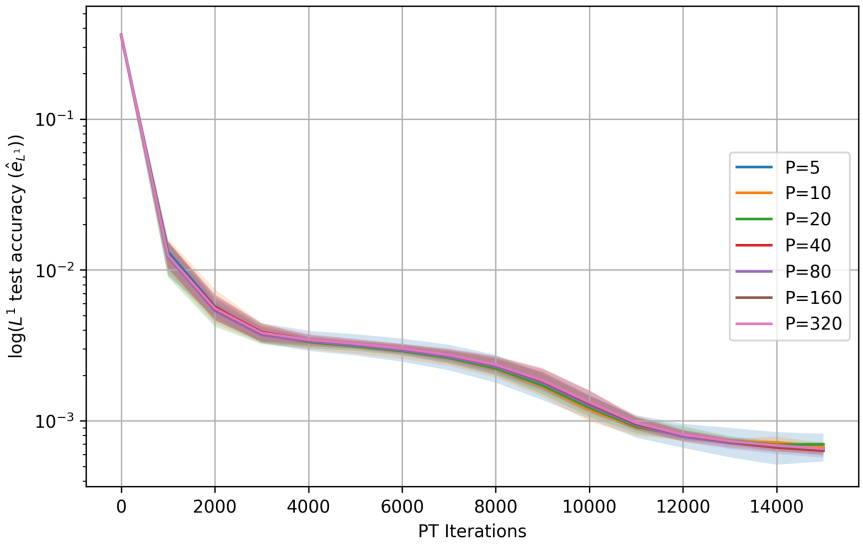


Figure 4.10: Convergence curves for PT with varying number of pre-training solutions (P).

The final $\hat{e}_{L^1}$ against P for both models are plotted in Figure 4.12. Note that these are not directly comparable as the PT-ELM-FBPINN models are only solving the test solution, while PT is optimising the supervised learning pre-training loss. This provides further evidence that the improved performance of PT-ELM-FBPINN over ELM-FBPINN (and FBPINN) is insensitive to the number of pre-training solutions used, at least for the 1D harmonic oscillator, with the exact values (see Table 4.5) showing the negligible effect that varying P has on performance. This means that, even for small P, the PT model is able to converge on a solution that results in an expressive basis for the 1D problem. This may be due to the relative simplicity for the domain-decomposition-based solution ansatz, but when comparing to the ELM-FBPINN baseline results, it is clear to see that pre-training does result in a performance gain, regardless of the number of solutions trained on.

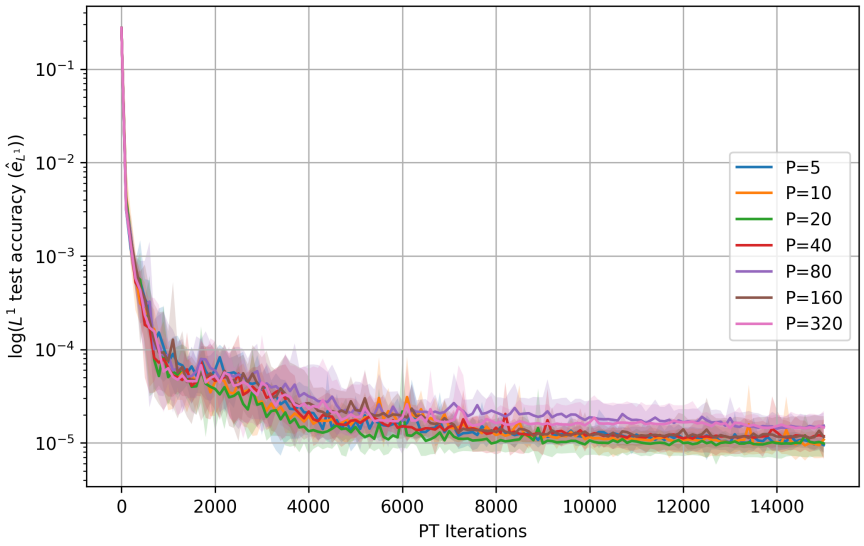


Figure 4.11: Convergence curves for PT-ELM-FBPINN with varying number of pre-training solutions (P).

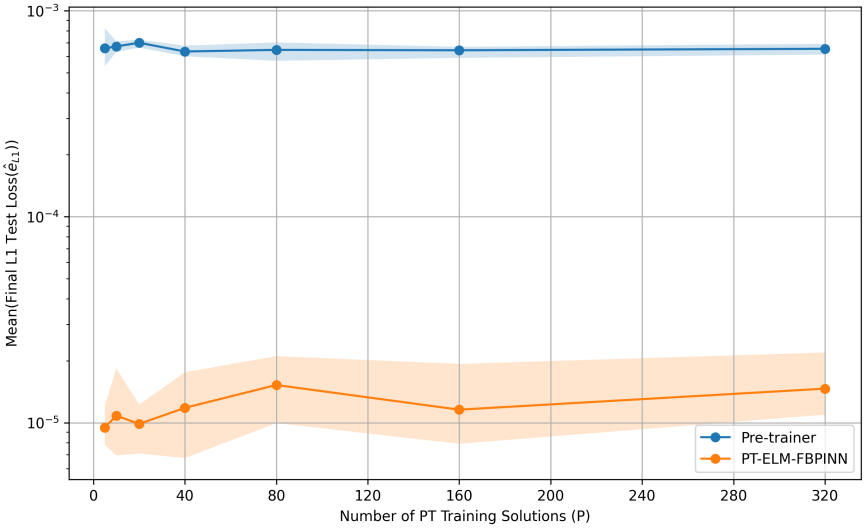


Figure 4.12: Final $\hat{e}_{L^1}$ values for PT and PT-ELM-FBPINN with varying number of pre-training solutions (P).

P	PT	PT-ELM-FBPINN
5	(6.57±2.82)e-04	(9.47±4.24)e-06
10	(6.71±0.8)e-04	(1.08±1.14)e-05
20	(6.99±0.603)e-04	(9.87±5.14)e-06
40	(6.34±0.759)e-04	(1.18±1.08)e-05
80	(6.46±1.3)e-04	(1.52±1.11)e-05
160	(6.43±0.777)e-04	(1.16±1.14)e-05
320	(6.54±0.786)e-04	(1.46±1.1)e-05

Table 4.5: Final $\hat{e}_{L^1}$ values and average run time for PT and PT-ELM-FBPINN for varying number of PT training solutions, P.

The condition number remain relatively stable as P is increased (see Figure 4.13 and Table 4.6). We observe that the condition number and rank of the matrix $\mathbf{M}$ do not vary in magnitude as the number of pre-training solutions varies, indicating that the number of PT training solutions has no adverse affect on the conditioning of the linear least-squares solve.

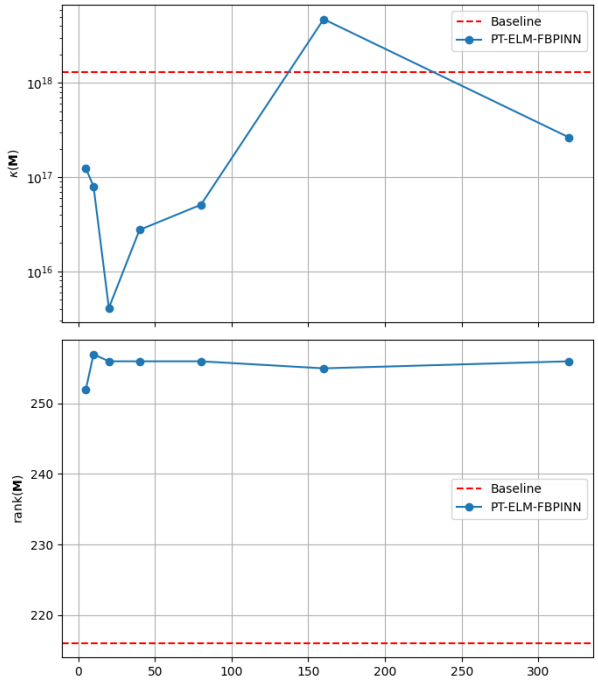


Figure 4.13: Condition number, $\kappa(\mathbf{M})$ and rank($\mathbf{M}$) vs number of PT training solutions, P.

P	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)
5	1.24e+17	252
10	8.00e+16	257
20	4.09e+15	256
40	2.76e+16	256
80	5.09e+16	256
160	4.74e+18	255
320	2.64e+17	256
ELM-FBPINN	1.26e+18	216

Table 4.6: SVD-based condition number and matrix rank for PT-ELM-FBPINN as a function of the number of training solutions.

Varying Number of Subdomains

Next, we investigate the effect of the number of subdomains, J, and fix all other parameters to the baseline setup. In Section 3.4.1, we observed that the ELM-FBPINN model outperformed the FBPINN model in terms of both precision and convergence, for all but the smallest values of J. This suggested an ability to take advantage of finer domain decompositions compared to the gradient-descent based method. Furthermore, for both models, there was a saturation of performance near $J \approx 40$, indicating a need for other parameters to be fine-tuned.

Convergence for the PT model is plotted in Figure 4.14, and for all other models in Figure 4.15. We see the convergence behaviour of FBPINN and ELM-FBPINN mirrored in the PT models' convergence. At values of $J < 8$, no models converge, displaying PINN-like behaviour and suggesting that the subdomains are too large for the local network output weights to resolve solutions with sufficient resolution of the oscillatory features. For $J > 8$, FBPINN exhibits an increase in convergence speed until $J = 16$, at which point the models progressively converge more slowly and to a worse final error, indicating diminishing returns from further domain subdivision. ELM-FBPINN generally improves in both convergence speed and final error precision as J increases. PT-ELM-FBPINN shows the sharpest convergence, with many of the models appearing to be fully converged in less than 4,000 iterations, significantly reducing the effective optimisation cost. This is significant as many of the ELM-FBPINN models appear to converge in the full 15,000 iterations, with a much more gradual reduction in loss. It should be noted that ELM-FBPINN on average typically achieves a lower final error value, with PT-ELM-FBPINN precision degrading for values of J greater than 24, suggesting a mild loss of robustness at very fine decompositions. Nevertheless, the best model is the PT-ELM-FBPINN model when $J = 24$, indicating that this domain decomposition has resulted in ideal conditions for the PT basis to represent the local solutions with both rapid convergence and high accuracy.

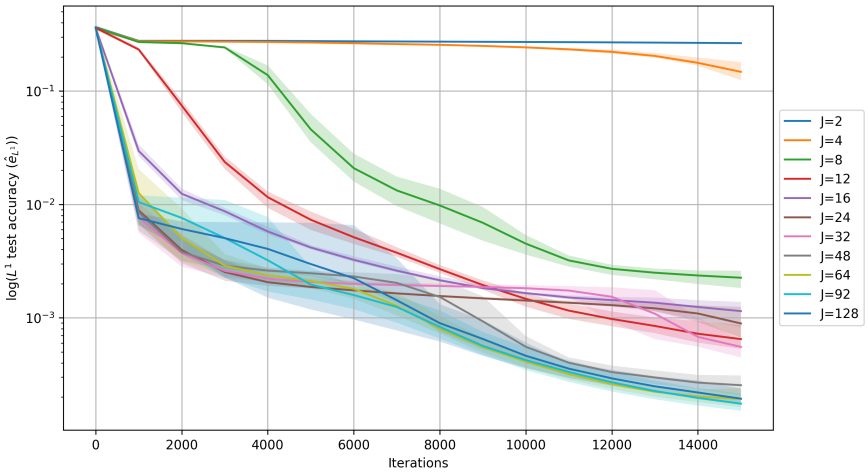


Figure 4.14: Convergence for each PT model with varying number of subdomains (J).

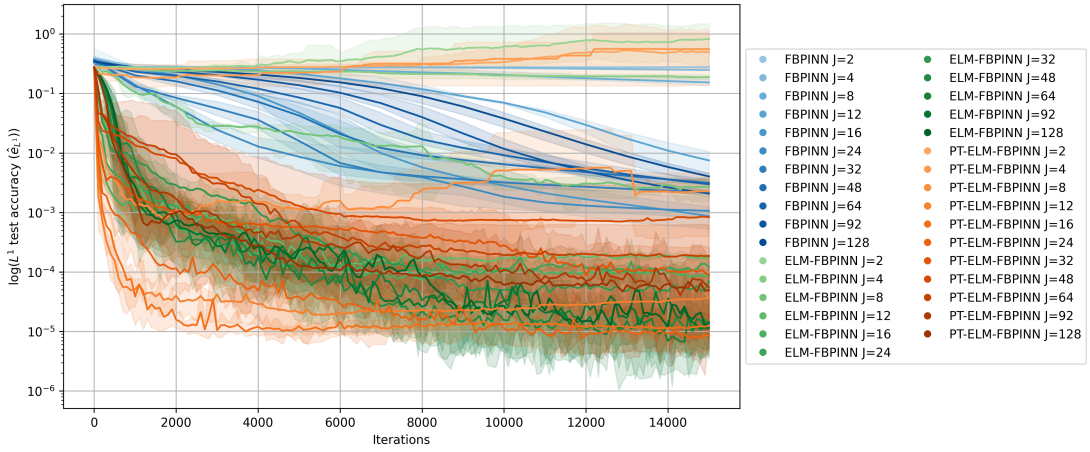


Figure 4.15: Convergence for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of subdomains (J).

The final error values for each model (except PT) are plotted in Figure 4.16 and reported in Table 4.7. Here, the PT-ELM-FBPINN's performance degradation is clearer, with final $\hat{e}_{L^1}$ improving up to $J = 24$, before a period of worsening performance which then improves and tends towards ELM-FBPINN performance. In spite of this fluctuating behaviour, both ELM models are more effective than FBPINN for $J > 4$. Additionally, if the best case scenarios are considered for both ELM models, the performance at higher values of J is much more comparable, but PT-ELM-FBPINN is still notably better with domain decompositions with $J = 12$ to $J = 24$.

J	FBPINN	ELM-FBPINN	PT-ELM-FBPINN	PT
2	(2.76±0.00346)e-01	(8.25±12.3)e-01	(4.96±8.9)e-01	(2.65±0.0608)e-01
4	(2.48±0.0577)e-01	(1.88±0.332)e-01	(5.56±9.15)e-01	(1.48±0.553)e-01
8	(1.53±0.384)e-01	(2.68±2.26)e-03	(2.26±4.21)e-03	(2.26±0.771)e-03
12	(7.49±6.24)e-03	(1.75±0.525)e-04	(3.76±5.27)e-05	(6.49±3.27)e-04
16	(8.66±7.3)e-04	(8.58±4.67)e-05	(1.25±1.38)e-05	(1.15±0.546)e-03
24	(1.07±0.96)e-03	(1.21±1.18)e-04	(8.61±8.42)e-06	(8.92±5.78)e-04
32	(2.18±0.321)e-03	(6.99±13.1)e-05	(8.78±24.6)e-05	(5.54±1.84)e-04
48	(3.13±0.521)e-03	(1.17±1.84)e-05	(8.32±37.1)e-04	(2.54±1.02)e-04
64	(2.98±1.23)e-03	(1.35±3.06)e-05	(1.85±6.01)e-04	(1.92±0.758)e-04
92	(2.07±0.401)e-03	(1.13±1.56)e-05	(5.45±13.4)e-05	(1.75±0.623)e-04
128	(4.03±1.52)e-03	(1.44±2.35)e-05	(4.95±9.24)e-05	(1.94±0.667)e-04

Table 4.7: Final $\hat{e}_{L^1}$ values for each model at different numbers of subdomains, J. PT results are included for discussion purposes and are not a direct comparison to other models' performance.

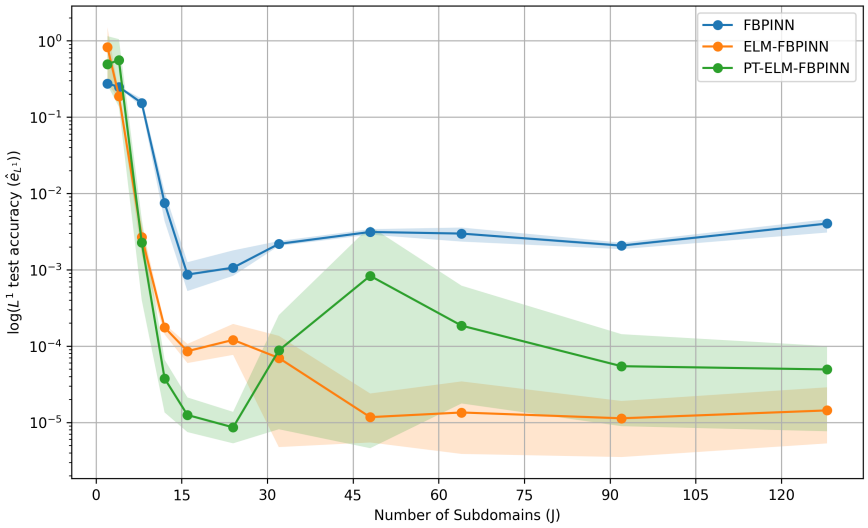


Figure 4.16: Final error, $\hat{e}_{L^1}$ vs number of subdomains (J) for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN.

To examine why PT-ELM-FBPINN performance is worse at higher values of J, we plot the basis functions for both ELM models to analyse how their expressivity changes under different domain decompositions. Figure 4.17 shows the centre subdomain ($j = \frac{J}{2}$) for ELM-FBPINN and PT-ELM-FBPINN when $J = 12$ and $J = 128$. The local solution inside the subdomain when $J = 12$ is significantly more oscillatory than when $J = 128$. When J is large, the subdomain widths are very small, resulting in significantly less of the global solution captured locally within a single subdomain and therefore a locally simpler target function. In contrast, when J is small, the subdomains are wide, which results in more of the oscillations being contained within each subdomain and hence a more challenging local approximation problem.

For ELM-FBPINN generally, hidden-layer basis functions appear quasi-linear, which is a poor fit for the local solution when $J = 12$ due to the strong oscillatory behaviour present in the wider subdomains. Conversely, they appear to be an appropriate choice for local solutions when $J = 128$, with some functions appearing to fit the local exact solution closely as the reduced subdomain width yields nearly linear local behaviour.

For PT-ELM-FBPINN, hidden-layer basis functions have more non-linearity in wider subdomains, which is a result of the supervised training pushing the weights

to learn the oscillatory features prior to freezing them for the least-squares stage. This creates a basis that is more appropriate than the randomly initialised basis of ELM-FBPINN for moderately sized subdomains. Even if they are not a perfect fit, there is an efficiency gain in the linear least-squares solve for the output weights, which is seen in the sharper convergence curves (see Figure 4.15) particularly for intermediate values of J. However, when $J = 128$ the supervised training causes the PT weights to learn fewer oscillations, creating flatter, less expressive basis functions that no longer provide a clear advantage over random features. In some cases, these pre-trained basis functions are a worse fit than the ELM-FBPINN basis functions because they retain residual non-linear structure misaligned with the nearly linear local solution. This is due to the PT weights learning some non-linear features, particularly from the early subdomains where oscillations are the highest and transferring this bias to all subdomains. As J increases and the local solutions become more linear, the PT produces quasi-linear basis functions that resemble the basis functions of ELM-FBPINN both visually and in approximation behaviour. This explains PT-ELM-FBPINN's convergence of performance towards ELM-FBPINN at large J, as the advantage of supervised pre-training diminishes when the local approximation task becomes nearly linear.

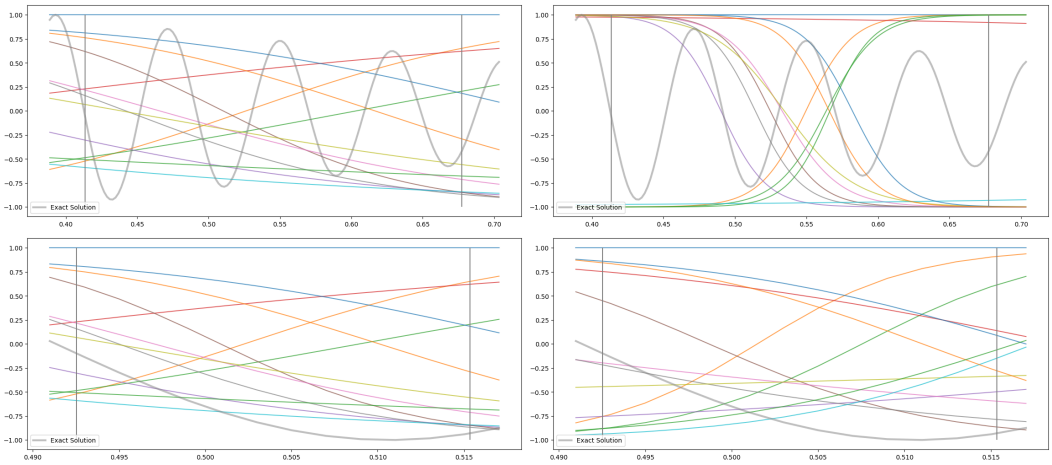


Figure 4.17: Basis functions for ELM-FBPINN and PT-ELM-FBPINN within the subdomain $\frac{J}{2}$. Top row: $J = 12$; Bottom row: $J = 128$. Left column: ELM-FBPINN; Right column: PT-ELM-FBPINN.

The condition number and rank of the matrix $\mathbf{M}$ for the ELM models are reported in

Figure 4.18 and Table 4.8 to further assess the numerical stability of the corresponding least-squares systems. We find that the condition number of PT-ELM-FBPINN is mostly stable and similar to ELM-FBPINN, except for values of $J < 24$ where a noticeable improvement is observed. As discussed above, the smaller values of J result in more oscillatory features present in the local subdomain solutions due to the increased spatial support of each window function. Pre-training produces more non-linear, better fitting basis functions with wider subdomains, which explains the improved conditioning at smaller values of J as the columns of $\mathbf{M}$ become less redundant and better aligned with the local solution structure. As J increases, the condition numbers of the two models converge, consistent with our expectations since the local solutions become nearly linear and the advantage of supervised feature shaping diminishes.

The rank of the PT-ELM-FBPINN matrices is consistently higher than ELM-FBPINN for $J < 32$, which indicates that the pre-training in the wider subdomains produces more linearly independent basis functions, eliminating redundancies and enhancing the effective dimensionality of the approximation space. For $J = 32$ and $J = 48$, PT-ELM-FBPINN has the lower rank matrices, with $J = 48$ being most rank deficient matrix relative to ELM-FBPINN among the configurations tested. This also corresponds to the worst performing PT-ELM-FBPINN model, and likely represents a situation in which pre-training results in some numerical instability in the matrix possibly due to over-adaptation of the hidden weights to specific oscillatory patterns. For $J > 48$, PT-ELM-FBPINN rank behaviour converges toward ELM-FBPINN rank behaviour, consistent with the trends observed previously and reinforcing the conclusion that the structural differences between the bases diminish as subdomains become sufficiently small.

	ELM-FBPINN		PT-ELM-FBPINN	
J	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)
2	8.75e+17	15	1.92e+17	18
4	1.16e+19	37	4.83e+16	48
8	2.22e+18	82	5.40e+15	103
12	4.59e+17	128	5.73e+13	155
16	1.39e+18	172	5.11e+15	206
24	1.27e+19	260	2.08e+18	304
32	2.68e+18	357	1.19e+18	342
48	7.99e+17	534	2.74e+18	478
64	8.17e+18	701	1.47e+18	710
92	3.03e+18	1007	4.75e+18	1079
128	3.64e+18	1402	2.81e+18	1404

Table 4.8: SVD-based condition number and matrix rank for ELM-FBPINN and pre-trained ELM-FBPINN as a function of the number of subdomains.

Varying Number of Basis Functions

We now examine the effect of varying the number of basis functions, C, which demonstrates how each model can handle different model capacities, when the subdomain solutions remain unchanged and the difficulty of the local approximation problem is fixed. At small values of C, each basis function must compensate for the reduced capacity of the network by representing a larger portion of the local solution, while at large values of C, the likelihood of redundant basis functions increases due to potential linear dependencies within the feature space.

The convergence for the PT models is plotted in Figure 4.19 and for all other models in Figure 4.20. All PT models converge to a similar $\hat{e}_{L^1}$, with a general trend of improved performance as C increases indicating that additional basis functions enhance representational flexibility. The performance gains indicate the improved expressivity of the basis functions, with the speed of initial convergence also increasing with C as the least-squares system is better able to capture the dominant solution modes. Regardless, even at small C, the PT achieves an acceptable fit to the $P = 250$ solutions,

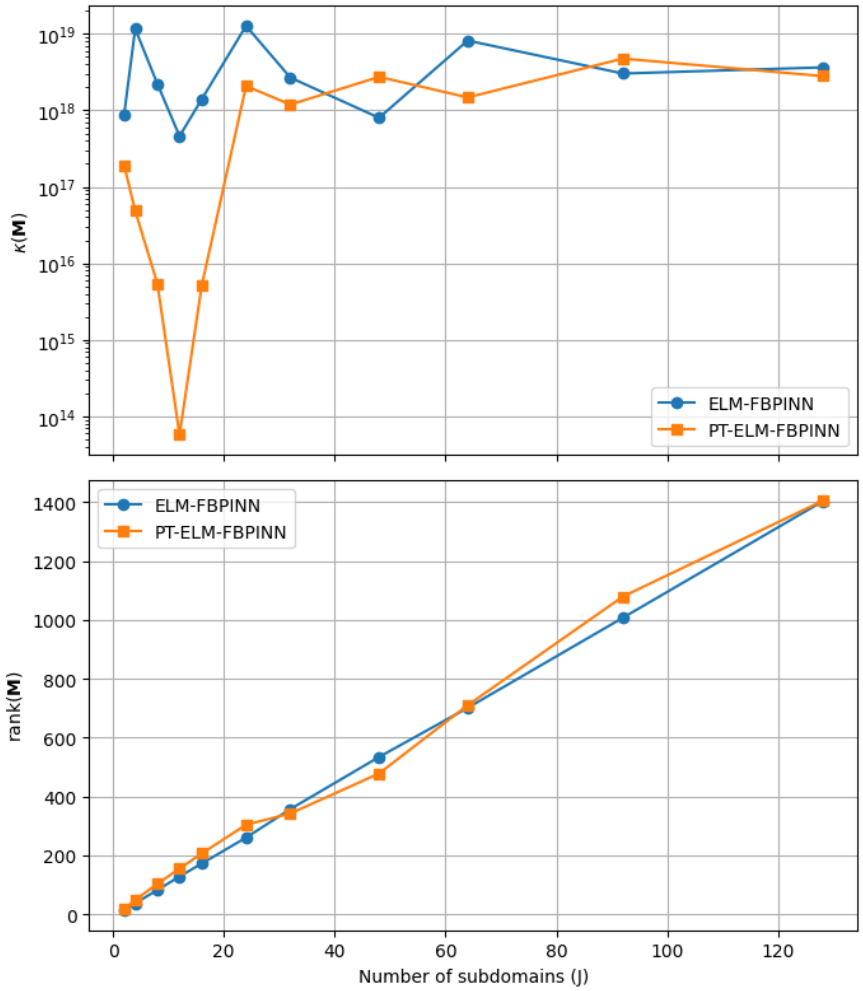


Figure 4.18: 2-norm condition number and rank of the matrix $\mathbf{M}$ for both ELM-FBPINN and PT-ELM-FBPINN for varing number of subdomains, J.

demonstrating robustness of the pre-trained basis even under constrained capacity. The FBPINN similarly exhibits an improved convergence speed with increasing C, with all models, except when $C = 2$, achieving a similar final error. This suggests a saturation in achievable accuracy beyond moderate capacity. The ELM-FBPINN has some numerical instability for the lowest value of C, but outperforms the FBPINN for $C \geq 8$, where sufficient expressivity is available. This shows that the gradient-descent based optimisation is more robust at lower values of C as the basis is more flexible and can adapt its hidden-layer weights during training. The linear least-squares method requires a sufficient network size to provide the expressivity needed to converge, particularly when the matrix $\mathbf{M}$ is static and the feature space cannot adapt dynamically.

PT-ELM-FBPINN also presents some numerical instability for low values of C. However, if the iterations were stopped early, before the loss started to increase again, PT-ELM-FBPINN would achieve a lower error, and have converged faster, than both FBPINN and ELM-FBPINN. This behaviour suggests that the instability arises primarily from over-iteration of the linear solver rather than from a fundamental limitation of the pre-trained basis. For $C \geq 8$, PT-ELM-FBPINN is the fastest to converge, and obtains the lowest $\hat{e}_{L^1}$ by an order of magnitude relative to the other architectures. Despite the numerical instability of the lowest values of C, this result indicates that pre-training the hidden-layer basis of ELM-FBPINN substantially enhances the performance of the fixed architecture by providing a more informative and better aligned feature space for the subsequent linear solve.

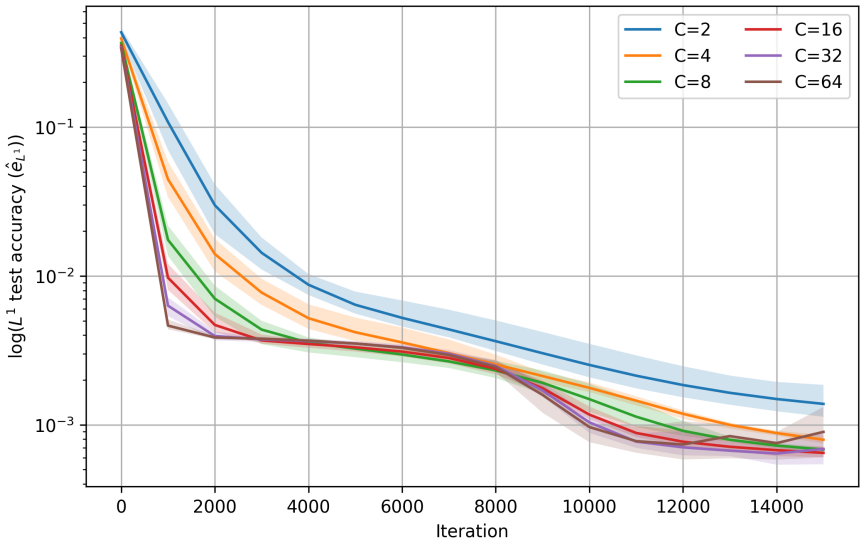


Figure 4.19: Convergence for PT with varying number of basis functions (C).

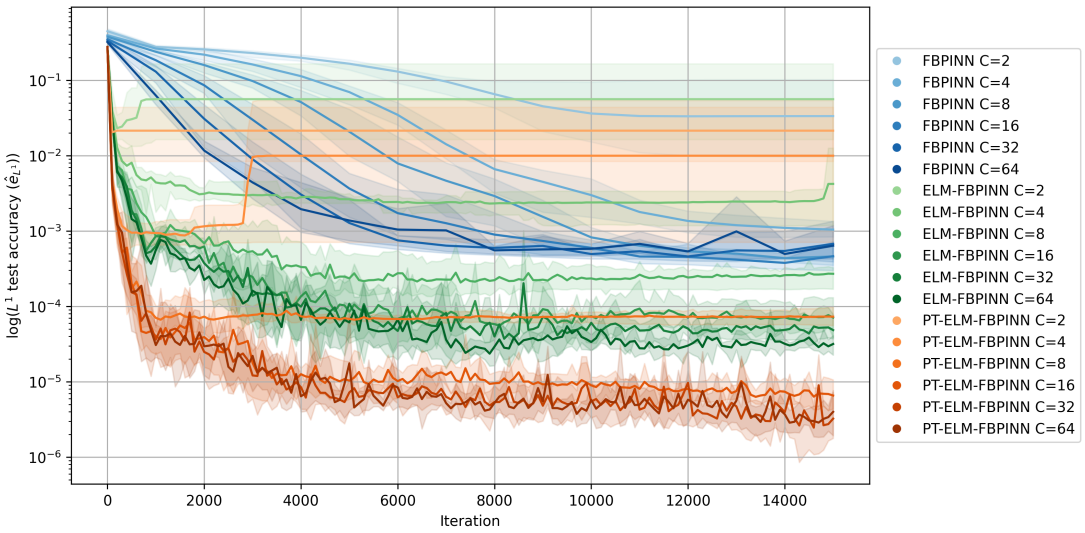


Figure 4.20: Convergence for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN with varying number of basis functions (C).

The final $\hat{e}_{L^1}$ values against C are plotted in Figure 4.21 and reported in Table 4.9. For all values of C, except the numerically unstable model with $C = 4$, PT-ELM-FBPINN is the model with the best final precision. With early stopping, PT-ELM-FBPINN would likely also outperform both models at $C = 4$, with ELM-FBPINN also benefitting for the $C = 2$ result. All models appear to have saturated in performance

at $C = 16$, likely due to the relative simplicity of the 1D problem with a domain decomposition training regime. However, it is clear that the addition of pre-trained weights has enabled ELM-FBPINN to better represent the solutions.

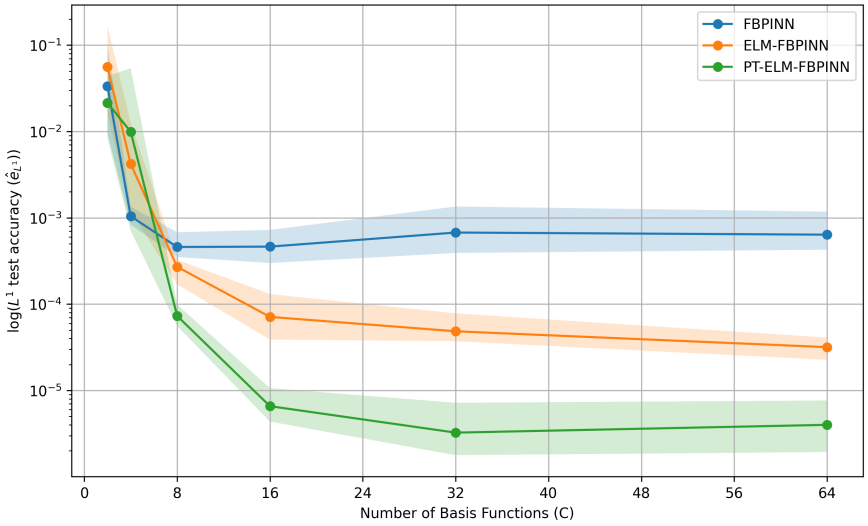


Figure 4.21: Final $\hat{e}_{L^1}$ values for PT and PT-ELM-FBPINN with varying number of basis functions, C.

C	FBPINN	ELM-FBPINN	PT-ELM-FBPINN	PT
2	(3.34±9.23)e-02	(5.6±14.9)e-02	(2.14±3.54)e-02	(1.38±0.722)e-03
4	(1.04±0.51)e-03	(4.21±11.4)e-03	(9.96±53.3)e-03	(7.94±1.05)e-04
8	(4.59±3.29)e-04	(2.7±1.57)e-04	(7.31±4.3)e-05	(6.81±0.669)e-04
16	(4.64±4.24)e-04	(7.11±9.19)e-05	(6.57±6.26)e-06	(6.47±1.08)e-04
32	(6.75±9.6)e-04	(4.85±4.1)e-05	(3.25±5.42)e-06	(6.88±3.42)e-04
64	(6.38±7.52)e-04	(3.17±1.83)e-05	(4±5.72)e-06	(8.96±7.14)e-04

Table 4.9: Final $\hat{e}_{L^1}$ values for each model at different number of basis functions, C; PT is separated for clarity.

The condition number and rank of the ELM models' matrices for each value of C are plotted and reported in Figure 4.22 and Table 4.10 respectively. Overall, the conditioning of the matrix, $\mathbf{M}$, is similar for both PT-ELM-FBPINN and ELM-FBPINN. There is a minor improvement at lower values of C, indicating improved linear independence of the basis functions after pre-training. However, the lower condition numbers observed for both models at small C are more likely attributable to the reduced number of columns, and do not necessarily indicate improved numerical stability overall,

particularly in light of the convergence behaviour observed at small C. At larger C, the condition number appears roughly equal for the two models, with pre-training of hidden-layer weights having no adverse effects.

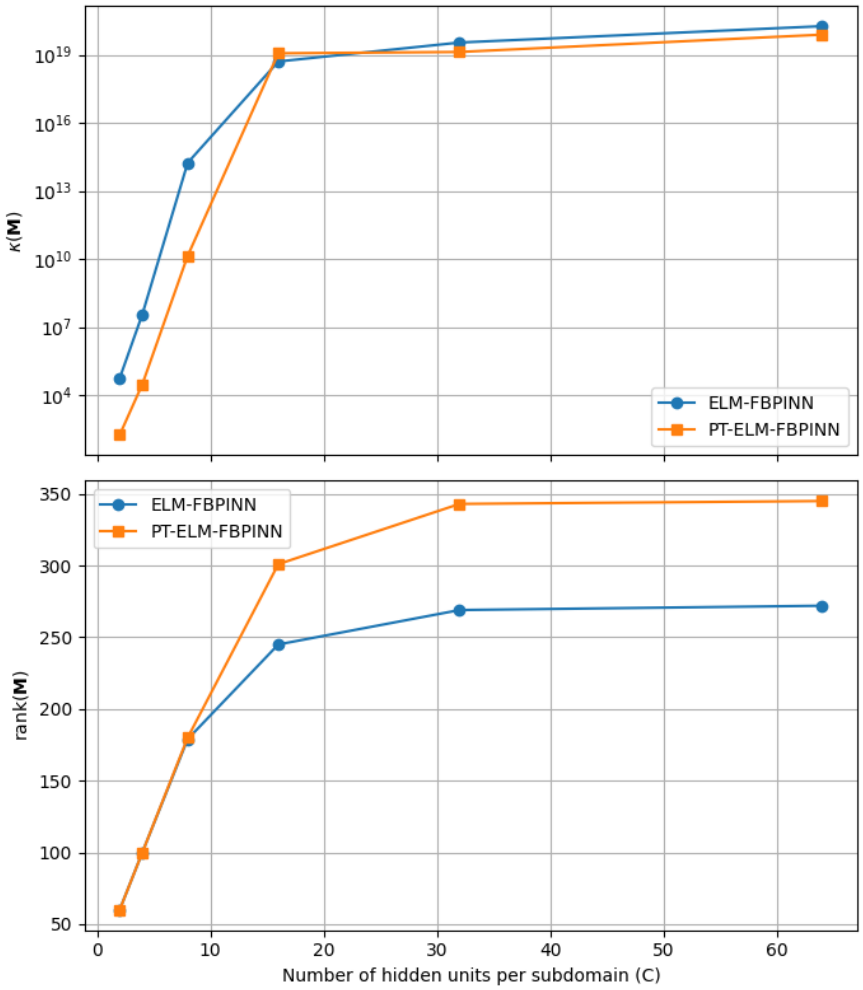


Figure 4.22: SVD-based condition number and rank of the matrix $\mathbf{M}$ for ELM-FBPINN and PT-ELM-FBPINN for varying number of basis functions, C.

For $C \leq 8$, the matrices of PT-ELM-FBPINN are all full rank, with ELM-FBPINN also having full rank matrices for $C = 2$ and $C = 4$. At $C = 8$, ELM-FBPINN is almost full rank, with only a single linearly dependent column. There is then a notable improvement in the rank of PT-ELM-FBPINN matrices for $C = 16$ compared to ELM-FBPINN. This improvement over ELM-FBPINN is maintained at higher values of C, but both models' ranks plateau in growth for $C > 16$. Given the relative simplicity of

the subdomain solutions when $J = 20$, this behaviour is in line with our expectations. While increasing C should equate to a more expressive basis, there are diminishing returns at large C, with significantly more redundant functions introduced.

	ELM-FBPINN		PT-ELM-FBPINN	
C	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)
2	5.50e+4	60	1.73e+2	60
4	3.61e+7	100	2.97e+4	100
8	1.73e+14	179	1.36e+10	180
16	5.10e+18	245	1.17e+19	301
32	3.49e+19	269	1.34e+19	343
64	1.84e+20	272	7.78e+19	345

Table 4.10: Condition numbers and ranks of ELM-FBPINN and PT-ELM-FBPINN matrices for varying number of hidden units, C.

Varying Depth of Hidden Layers With Number Of Subdomains

We now investigate the effect of varying number of hidden layers for all models in order to assess how architectural depth influences approximation quality and optimisation behaviour. Specifically, we choose to vary hidden-layer depth, h, and inversely scale the number of subdomains, J so that increases in depth are compensated by reductions in domain decomposition. The purpose of this inverse scaling of J, is to isolate the effect of network depth by preserving the overall number of parameters and avoiding confounding improvements due solely to increased capacity. Previously, we observed weight saturation when using deep-layer architectures, with both FBPINN and ELM-FBPINN exhibiting this behaviour through increasingly flat hidden-layer basis functions. ELM-FBPINN generally has quasi-linear basis functions regardless of h or J, while FBPINN does show some non-linearity in the trained basis, particularly as the subdomain width increases as J decreases, allowing greater adaptation to local oscillatory structure. This non-linearity correlates with performance, with FBPINN outperforming ELM-FBPINN in final $\hat{e}_{L^1}$ for lower values of h, where the adaptive hidden weights provide a clear advantage. With pre-training, we expect to see more expressive, non-linear hidden-layer basis functions, which could mitigate the weight saturation issue of ELM-FBPINN by shaping the feature space prior to freezing the

weights. To address the numerical instability seen in the LSQR solver for deeper layer networks, we employ early stopping for the ELM models, thereby preventing divergence of the linear solver while retaining the dominant convergence behaviour.

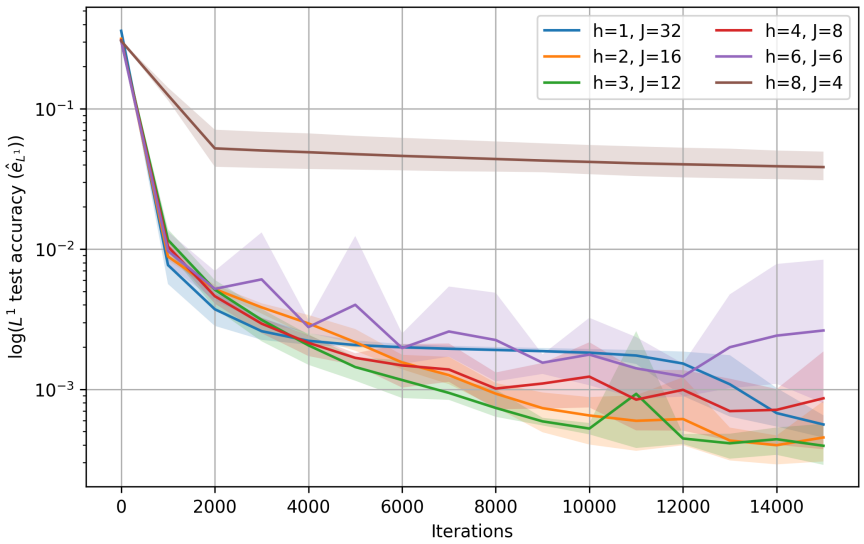


Figure 4.23: Convergence of PT for varying number of hidden layers, h, and number of subdomains, J.

Figure 4.23 shows the convergence behaviour of the PT models. For $h \leq 4$, each model converges to a similar final $\hat{e}_{L^1}$. The convergence of the $h = 6$ model is less smooth that the other models, indicating a level of instability in the weights, however, the final $\hat{e}_{L^1}$ is still relatively reasonable. The only model that appears to have not fully converged is when $h = 8$. At this depth, $J = 4$, which results in wide subdomains, with high frequency local solutions. In this case, the added capacity gained in additional hidden-layers is offset by the reduced domain decomposition and the basis is not able to accurately capture the solutions to the pre-trained problems.

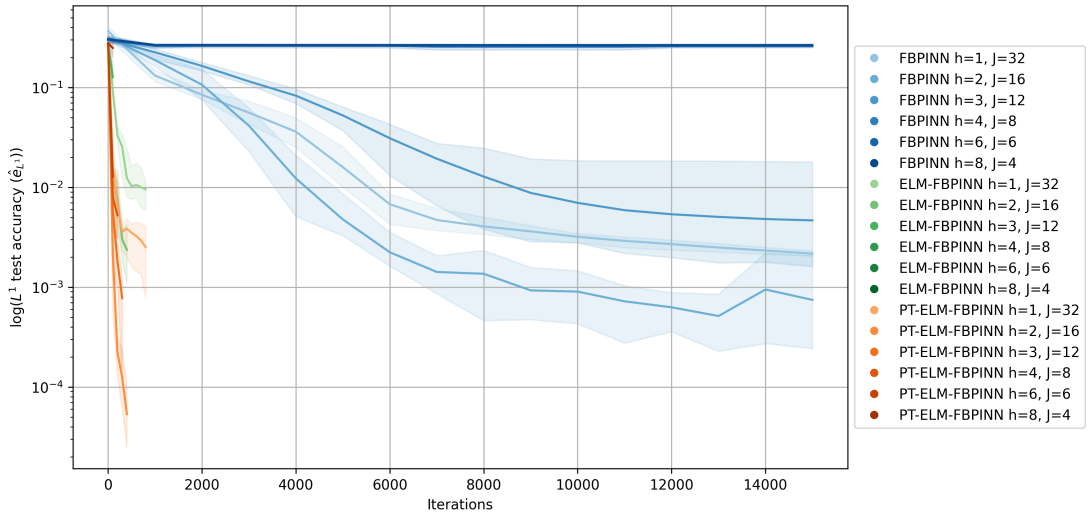


Figure 4.24: Convergence of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of hidden layers, h, and number of subdomains, J.

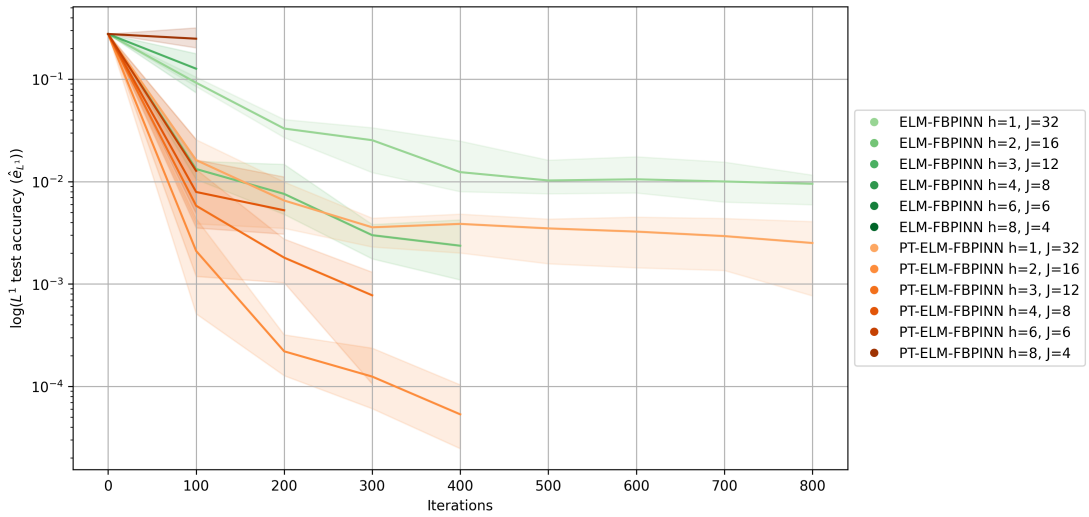


Figure 4.25: Convergence of ELM-FBPINN, and PT-ELM-FBPINN for varying number of hidden layers, h, and number of subdomains, J.

Figure 4.24 shows the convergence of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN. Here, we can see the FBPINN models when $h \leq 3$ all converge, with $h = 2$ achieving the best precision. All other FBPINN models fail to converge, indicating that the hidden-layer weights have saturated. For clarity, the ELM models' convergence is plotted separately in Figure 4.25. The effect of the pre-trained weights is visually clear,

with all six PT-ELM-FBPINN models running for at least 100 iterations. However, for ELM-FBPINN, the least-squares solver LSQR only computes iterates when $h \leq 3$; for larger depths, LSQR fails at the first iteration. This is likely caused by the saturation of the hidden-layer weights. This is more clearly reported in Table 4.11. Despite running for 100 iterations at $h = 8$, the error for PT-ELM-FBPINN has not reduced significantly from its value at initialisation indicating some numerical instability. Nevertheless, PT-ELM-FBPINN successfully ran for all other network depths. The number of iterations for $h = 6$ does indicate some level of instability for the model, but when the predicted solution for FBPINN, ELM-FBPINN and PT-ELM-FBPINN are compared with the exact solution, it is clear that only PT-ELM-FBPINN computes an acceptably accurate solution.

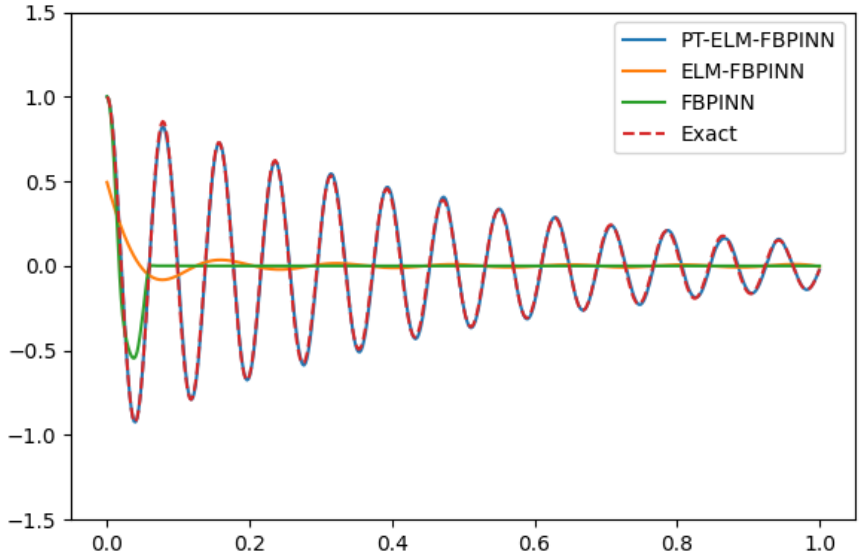


Figure 4.26: Predicted solutions for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN with number of hidden layers $h = 6$, and number of subdomains $J = 6$.

		ELM-FBPINN		PT-ELM-FBPINN	
h	J	$\hat{e}_{L^1}$	Iterations	$\hat{e}_{L^1}$	Iterations
1	32	(9.53±5.64)e-03	800	(2.52±3.31)e-03	800
2	16	(2.37±3.14)e-03	400	(5.34±7.9)e-05	400
3	12	(8.04±15.2)e-02	166	(7.76±12)e-04	300
4	8	N/A	0	(5.25±8.06)e-03	200
6	6	N/A	0	(1.27±2.09)e-02	100
8	4	N/A	0	(2.49±1.15)e-01	100

Table 4.11: Final $\hat{e}_{L^1}$ values and number of iterations for varying network depth, h, and number of subdomains, J.

The final $\hat{e}_{L^1}$ value as h is varied for each model is plotted in Figure 4.27, and reported in Table 4.12. Between the three methods, PT-ELM-FBPINN is the best performing for the majority of the tests, achieving lower $\hat{e}_{L^1}$ values. Crucially, it is also the most numerically stable method, with only $h = 8$ not converging to an acceptable error. Both FBPINN and ELM-FBPINN exhibit saturated weights for $h \geq 4$, while PT-ELM-FBPINN produces significantly more complex basis functions (see Figure 4.28). While performance does decrease as h increases, this can be linked to the complexity of the local solutions also increasing as J is decreased. This suggests that increasing depth alone, resulting in more complex, non-linear basis functions, is not sufficient to overcome the added difficulty of the reduced domain decomposition.

h	J	FBPINN	ELM-FBPINN	PT-ELM-FBPINN	Pre-Trainer
1	32	(2.17±0.259)e-03	(9.53±5.64)e-03	(2.52±3.31)e-03	(5.61±1.99)e-04
2	16	(7.45±18.3)e-04	(2.37±3.14)e-03	(5.34±7.9)e-05	(4.53±5.18)e-04
3	12	(4.67±16.3)e-03	(8.04±15.2)e-02	(7.76±12)e-04	(3.96±2.78)e-04
4	8	(2.65±0.0335)e-01	(2.76±0)e-01	(5.25±8.06)e-03	(8.63±14.9)e-04
6	6	(2.59±0.125)e-01	(2.76±0)e-01	(1.27±2.09)e-02	(2.63±7.6)e-03
8	4	(2.67±0.0153)e-01	(2.77±0)e-01	(2.49±1.15)e-01	(3.84±1.85)e-02
Time (s)		16.7±2.07	8.05±0.134	7.3±0.116	22.6±0.856

Table 4.12: Final $\hat{e}_{L^1}$ values and average run time for varying network depth, h and number of subdomains, J.

The SVD-based condition number and rank of the matrix $\mathbf{M}$ for the ELM models are plotted in Figure 4.29, with exact values reported in Table 4.13. As h increases, and J decreases, we see that the conditioning of the matrix for ELM-FBPINN remains

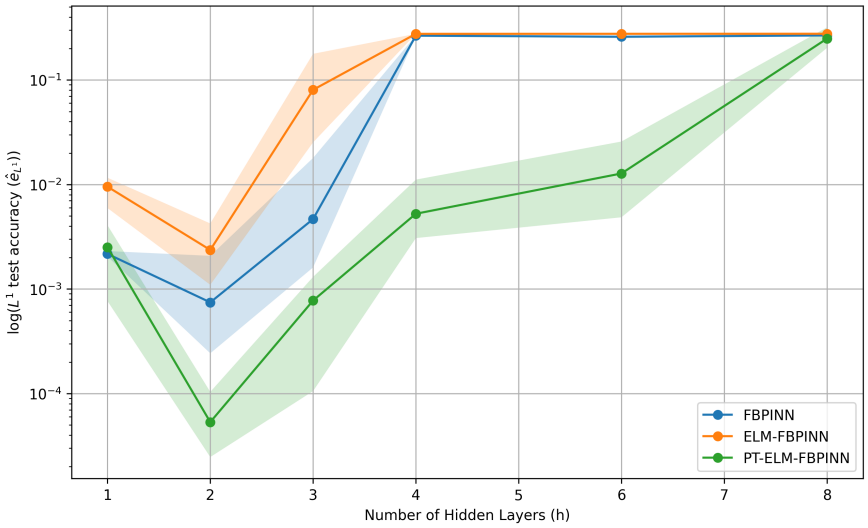


Figure 4.27: Final $\hat{e}_{L^1}$ of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of hidden layers, h, and number of subdomains, J.

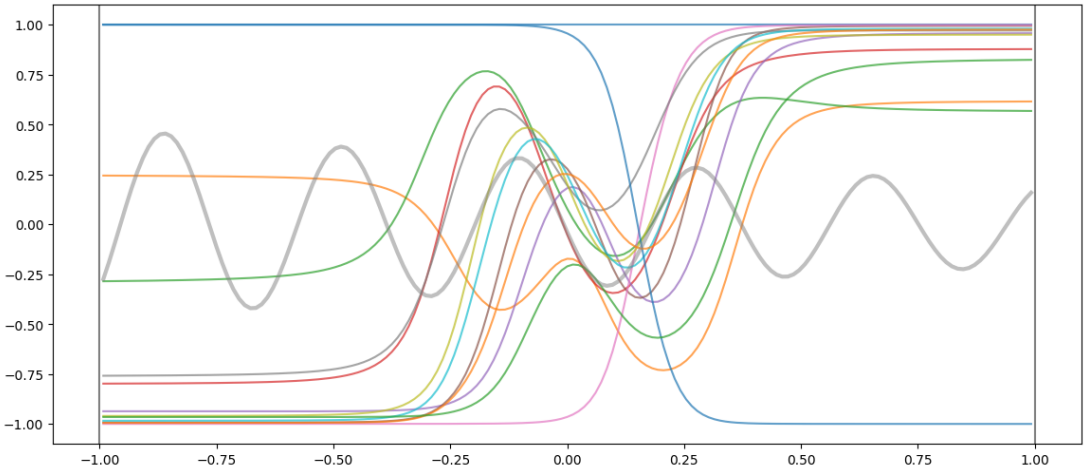


Figure 4.28: Hidden-layer basis functions for PT-ELM-FBPINN with $h = 4$ number of hidden layers, and $J = 8$ number of subdomains. The subdomain corresponding to $j = \frac{J}{2}$ is shown.

stable. This behaviour is reflected in the unchanging appearance of the quasi-linear basis functions at varying depths. The PT-ELM-FBPINN conditioning improves with increasing depth, which correlates with more stable convergence. The rank of $\mathbf{M}$ for both models decreases, but this is due to the decreasing number of subdomains reducing the total size of the matrix. However, it should be noted that for all values of $h > 1$,

the ranks of the PT-ELM-FBPINN matrices are either full or near-full rank. This is indicative of the added complexity of the hidden-layer basis functions, resulting in a reduction in linear dependence.

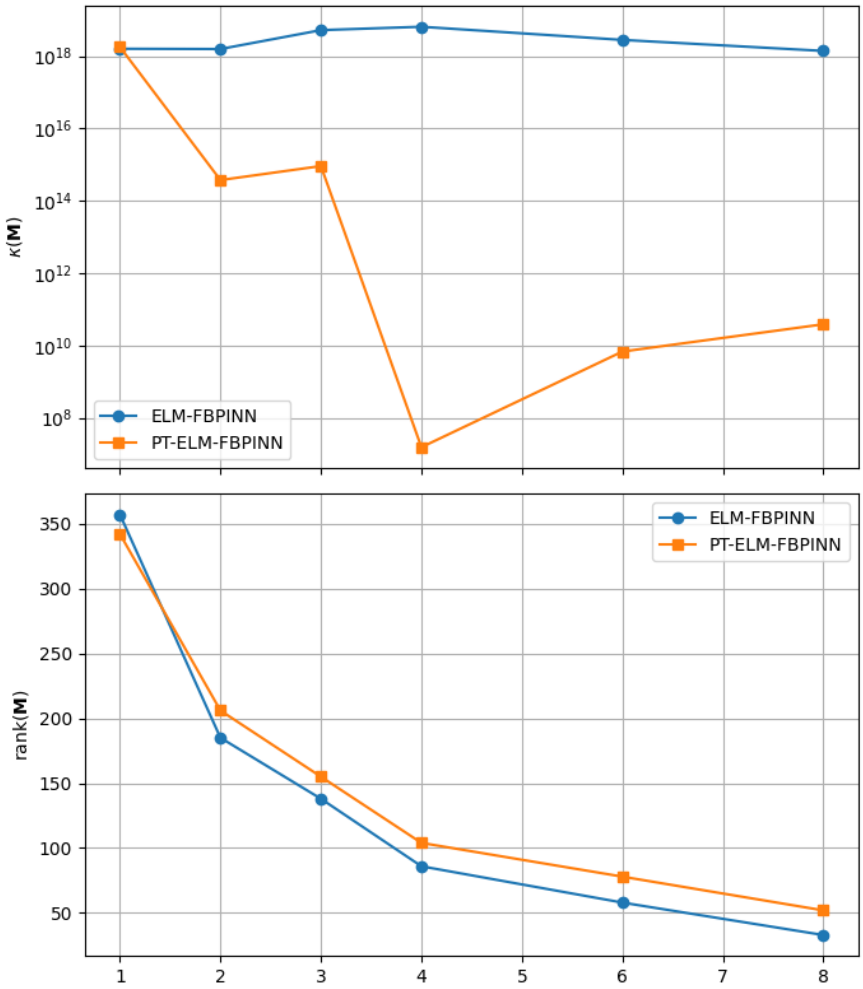


Figure 4.29: SVD-based condition number and rank of the matrix $\mathbf{M}$ for ELM-FBPINN and PT-ELM-FBPINN for varying number of hidden layers, h and number of subdomains, J.

		ELM-FBPINN		PT-ELM-FBPINN	
h	J	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)
1	32	1.60e+18	357	1.79e+18	342
2	16	1.57e+18	185	3.73e+14	206
3	12	5.24e+18	138	9.11e+14	155
4	8	6.50e+18	86	1.52e+7	104
6	6	2.81e+18	58	6.74e+9	78
8	4	1.40e+18	33	3.84e+10	52

Table 4.13: Condition numbers and ranks of ELM-FBPINN and PT-ELM-FBPINN matrices for varying number of hidden layers, h, and number of subdomains, J.

Varying Depth of Hidden Layers With Number Of Basis Functions

We now fix the number of subdomains, J and inversely scale the number of basis functions, C with increasing number of hidden layers, h so that depth is increased while maintaining a comparable overall parameter budget. We again apply early stopping to address some numerical instability observed in the ELM models' results, thereby preventing divergence of the LSQR iterations while preserving the dominant convergence trends.

The convergence of the PT models can be seen in Figure 4.30, and the convergence of all other models is shown in Figure 4.31. For visibility, the ELM models' convergence curves have also been separately plotted in Figure 4.32 to allow clearer comparison of their early-stage behaviour. The PT models all converge to a similar error, each with an initial sharp decrease, before a gradual slope for the remaining iterations, indicating rapid alignment of the pre-trained basis with the target solution followed by slower refinement. For all values of h and C, the PT fit the pre-training solutions well, with no significant difference in the final $\hat{e}_{L^1}$ values across the tested configurations. This indicates that increasing h does not adversely affect the ability of the supervised PT model to fit the $P = 250$ solutions.

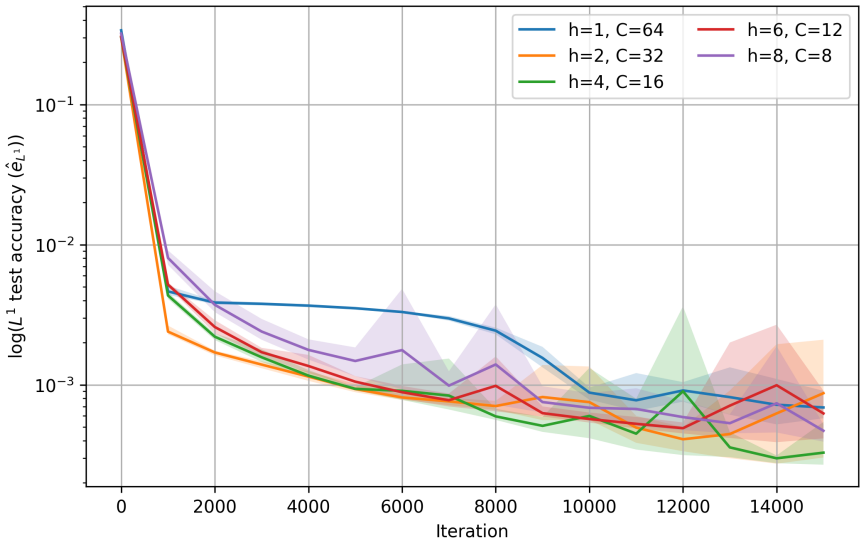


Figure 4.30: Convergence of PT models with varying number of hidden layers, h, and number of basis functions, C.

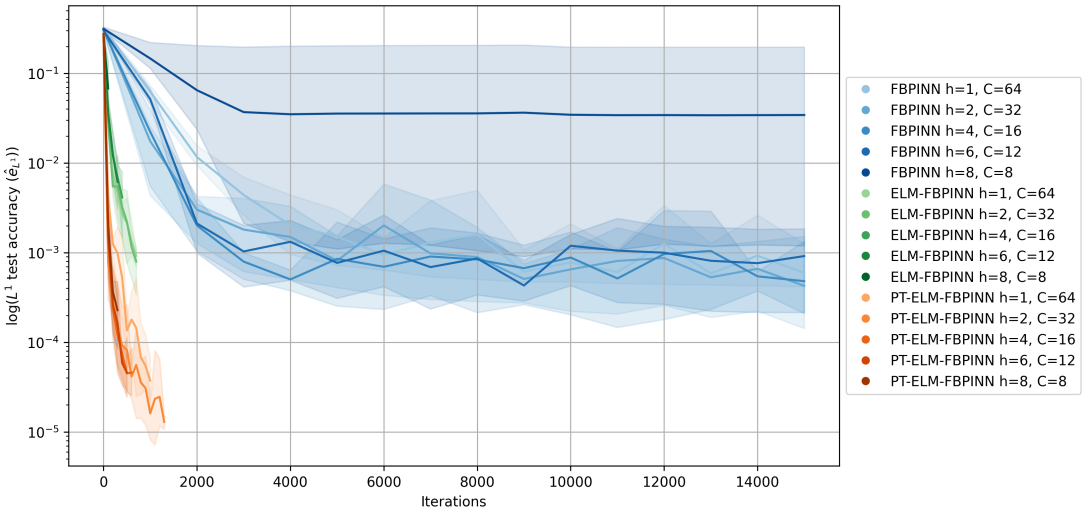


Figure 4.31: Convergence for FBPINN, ELM-FBPINN and PT-ELM-FBPINN with varying number of hidden layers, h and number of basis functions, C.

The FBPINN models have also all converged similarly to each other, except for the $h = 8$ model which exhibits noticeably different behaviour. We also note that, for this model, which has 8 hidden layers and 8 basis functions per subdomain, the variation in performance is very large. This indicates a level of sensitivity of the hidden-

layer weights that can drastically alter the performance of the gradient-descent based optimisation, particularly in deeper architectures where saturation effects are more pronounced. In the best case scenario, the $h = 8$ model also converges to approximately the same $\hat{e}_{L^1}$ as the other models with lower h, demonstrating that depth alone is not inherently detrimental but introduces optimisation instability. For all other values of h, the FBPINN converges acceptably, reproducing the test solution well and maintaining consistent accuracy across seeds.

Each ELM-FBPINN model has converged, despite the numerical instabilities, with the accuracy decreasing as h is increased and C is decreased reflecting the reduced expressive capacity per layer. While $C = 8$ basis functions should be a sufficient number for ELM-FBPINN to perform adequately (see Table 4.9), the $h = 8$ layers results in the LSQR solver only running for < 200 iterations due to early stopping triggered by instability. The number of iterations generally decreases as h increases, but this can be attributed to the numerical instability rather than an improvement in efficiency since premature termination limits refinement of the least-squares solution. Nevertheless, for all values of $h < 8$, the ELM-FBPINN models achieve acceptable solutions with errors remaining within a reasonable range.

For PT-ELM-FBPINN, the results are even more promising, with each models' accuracy outperforming its ELM-FBPINN counterpart by one or two orders of magnitude across nearly all tested configurations. The speed of convergence has also improved, with much sharper gradients in the earliest iterations when compared to ELM-FBPINN. This highlights the benefit of a better aligned pre-trained feature space. Stability is also better for PT-ELM-FBPINN models, with the final accuracy remaining relatively similar for different values of h and C, indicating reduced sensitivity to architectural depth when pre-training is employed.

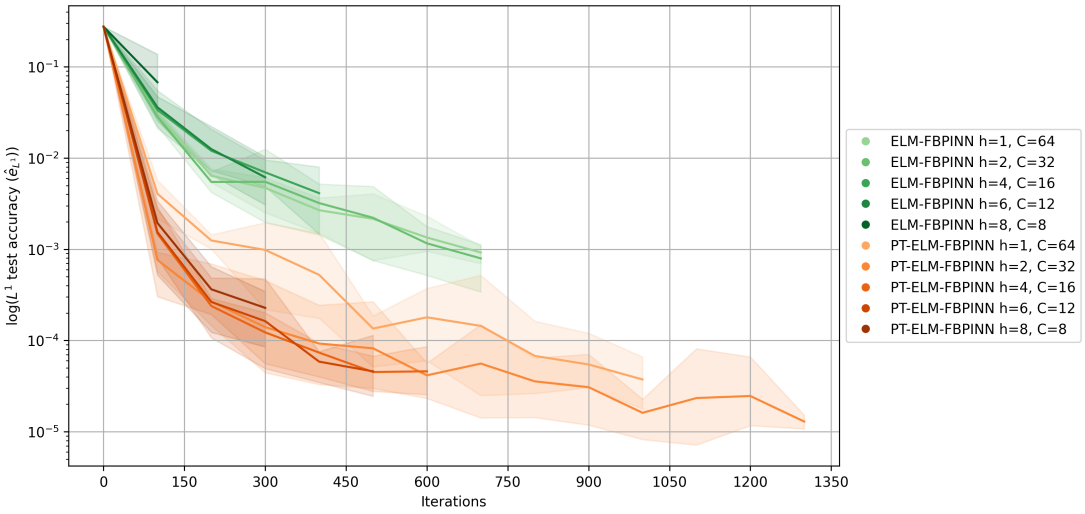


Figure 4.32: Convergence for ELM-FBPINN and PT-ELM-FBPINN with varying number of hidden layers, h and number of basis functions, C.

		ELM-FBPINN		PT-ELM-FBPINN	
h	C	$\hat{e}_{L^1}$	Iterations	$\hat{e}_{L^1}$	Iterations
1	64	(5.06±6.87)e-04	883	(4.53±10)e-05	1466
2	32	(6.29±8.75)e-04	883	(1.29±0.446)e-05	1300
4	16	(3.01±2.57)e-03	566	(4.59±6.03)e-05	600
6	12	(7±13.5)e-03	400	(4.59±8.96)e-05	500
8	8	(4.26±7.66)e-02	166	(2.28±2.6)e-04	300

Table 4.14: Final $\hat{e}_{L^1}$ values and number of iterations for varying network depth, h and number of basis functions, C.

Figures 4.33 and 4.15 better illustrate the improved performance of PT-ELM-FBPINN over the other models. While FBPINN generally is more accurate that ELM-FBPINN, PT-ELM-FBPINN is an order of magnitude more accurate than FBPINN on the test solution. In the case of $h = 8$, which both FBPINN and ELM-FBPINN show some level of instability and sensitivity, PT-ELM-FBPINN has a similar performance to lower values of h. The implication of this is that pre-training the hidden-layer basis, and then freezing the weights, results in a more stable and accurate solver when increasing network hidden layer depth, for this 1D problem.

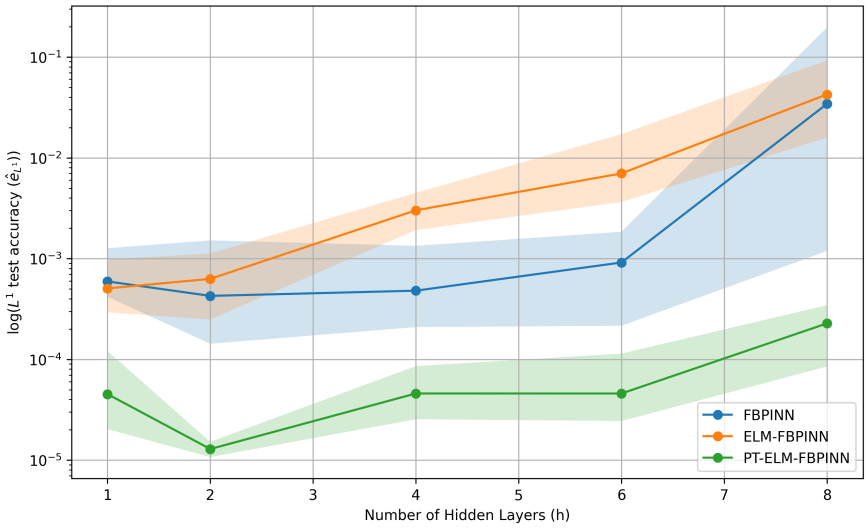


Figure 4.33: Final $\hat{e}_{L^1}$ for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of hidden layers, h, and number of basis functions, C.

h	C	FBPINN	PT	ELM-FBPINN	PT-ELM-FBPINN
1	64	(5.94±8.49)e-04	(6.89±3.28)e-04	(5.06±6.87)e-04	(4.53±10)e-05
2	32	(4.27±13.7)e-04	(8.71±17.9)e-04	(6.29±8.75)e-04	(1.29±0.446)e-05
4	16	(4.81±11.3)e-04	(3.28±2.74)e-04	(3.01±2.57)e-03	(4.59±6.03)e-05
6	12	(9.15±16.3)e-04	(6.25±4.92)e-04	(7±13.5)e-03	(4.59±8.96)e-05
8	8	(3.43±19.5)e-02	(4.7±1.29)e-04	(4.26±7.66)e-02	(2.28±2.6)e-04

Table 4.15: Final $\hat{e}_{L^1}$ values and average run time for varying network depth, h, and number of basis functions, C.

The condition number and rank of the matrix $\mathbf{M}$ against network depth is plotted for both ELM models in Figure 4.34. Table 4.16 contains the exact values for both models. The conditioning of the ELM-FBPINN matrices is relatively stable, with a slight downward trend. In contrast, the conditioning of the PT-ELM-FBPINN matrix is significantly improved as h is increased. For $h = 1$ and $h = 2$, the matrices for both ELM-FBPINN and PT-ELM-FBPINN have similar condition numbers, but the rank of the PT-ELM-FBPINN matrices is much higher. Despite this, both models' matrices at the lower values of h are very rank deficient. While the rank of both models' matrices reduces as C is decreased, reducing the total columns, the ranks of the PT-ELM-FBPINN matrices remain much higher than those of the ELM-FBPINN

matrices. The matrices for PT-ELM-FBPINN with $h \geq 4$ are near-full or full rank, showing the linear independence of the basis functions.

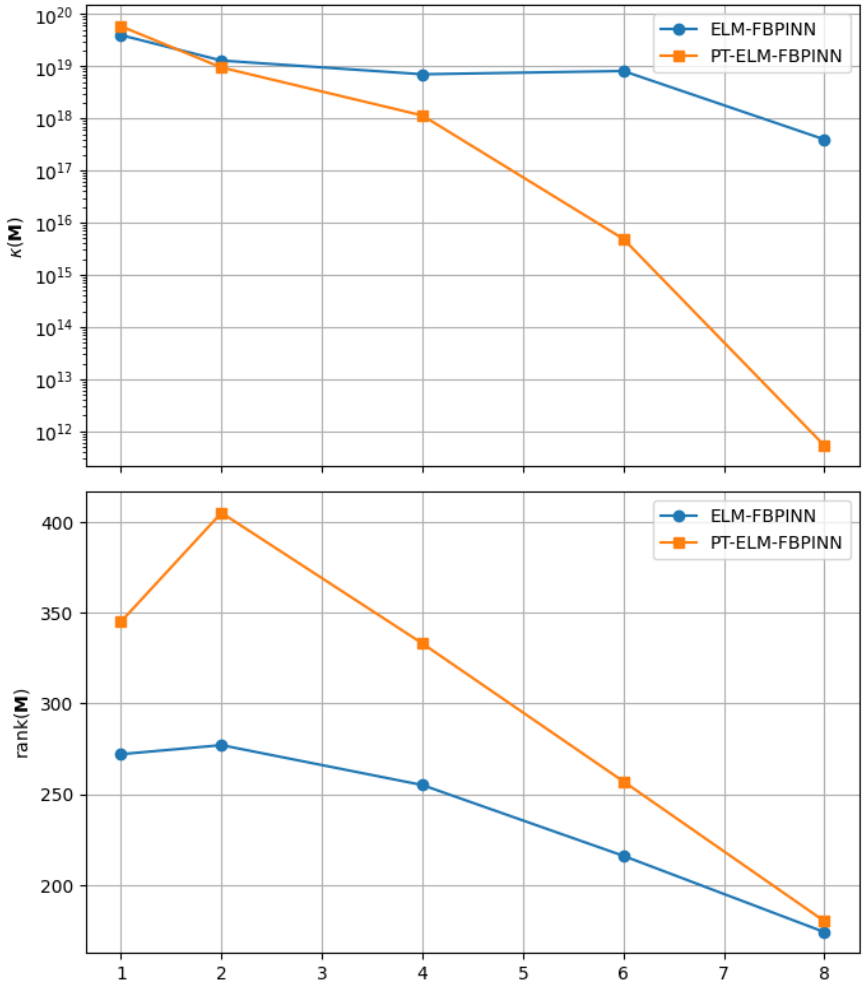


Figure 4.34: SVD-based condition number and rank of the matrix $\mathbf{M}$ for ELM-FBPINN and PT-ELM-FBPINN.

		ELM-FBPINN		PT-ELM-FBPINN	
h	C	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)
1	64	3.96e+19	272	5.85e+19	345
2	32	1.29e+19	277	9.48e+18	405
4	16	7.02e+18	255	1.14e+18	333
6	12	8.11e+18	216	4.89e+15	257
8	8	3.95e+17	174	5.32e+11	180

Table 4.16: Condition numbers and ranks of ELM-FBPINN and Pre-trained ELM-FBPINN matrices for varying number of hidden units, h and number of basis functions C.

Transfer Learning Study

Finally, we conduct a small transfer learning study where transfer learning refers to the practice of reusing parameters learned on one task to accelerate or improve performance on a related task. The purpose of this study is to investigate the potential for pre-training simpler problems, and reusing the weights to solve more difficult, high frequency problems. The idea is to perform a feature space approximation for the hidden basis on a lower frequency problem which should be a quick, easy, and efficient task relative to directly training on a highly oscillatory solution. This trains the hidden-layer weights towards the chosen problem space, taking advantage of the more expressive, non-linear hidden-layer basis, without necessarily requiring an expensive pre-train stage on a potentially difficult, high frequency problem. In this way, the computational cost is reduced while structural information about the solution class is retained. We also investigate the effect of pre-training on a higher frequency, and testing on lower frequencies to evaluate whether a more complex learned feature space generalises across scales. This gives insight into whether the more non-linear, complex basis can provide more accurate results at the cost of more expensive pre-training or whether simpler pre-training is sufficient for effective knowledge transfer.

For our tests, we fix the number of basis functions for all models at $C = 12$ to ensure consistent representational capacity. We reuse the baseline PT model, which was trained on a solution frequency of $\omega_0 = 80$ and $J = 20$ subdomains. For FBPINN, ELM-FBPINN, and PT-ELM-FBPINN, we scale the number of subdomains alongside

the solution frequency to maintain comparable local resolution as the problem difficulty increases. The goal is to ensure that as the frequency increases, the local subdomain solutions remain similar in terms of solution complexity between tests, isolating the effect of the networks scaling in size, as well as the effect of the pre-trained weights for PT-ELM-FBPINN without conflating these factors with changes in local approximation difficulty.

The FBPINN is the worst performing model overall, with the slowest convergence and lowest accuracy values across each test, particularly at high values of ω_0. The ELM models have all converged, with accuracy and convergence speed reducing as solution frequency increases. This drop in performance is not unexpected as even with scaling J, the higher-frequency global solutions place more stress on the networks' expressive power. Nevertheless, both ELM models outperform the FBPINN across all values of ω_0, with the performance gap increasing at higher frequencies. This highlights the advantage of linear least-squares optimisation in more challenging regimes. The performance of PT-ELM-FBPINN is comparable to ELM-FBPINN at the lowest values of ω_0, showing that the pre-trained weights of a higher frequency do not add any notable expressive power for the simpler, lower frequency regimes. When solution frequency matches the test frequency at $\omega_0 = 80$, the PT-ELM-FBPINN model achieves the lowest $\hat{e}_{L^1}$ with the fastest convergence. This trend continues for the highest frequencies, with PT-ELM-FBPINN proving to be the most robust architecture in terms of convergence speed and accuracy.

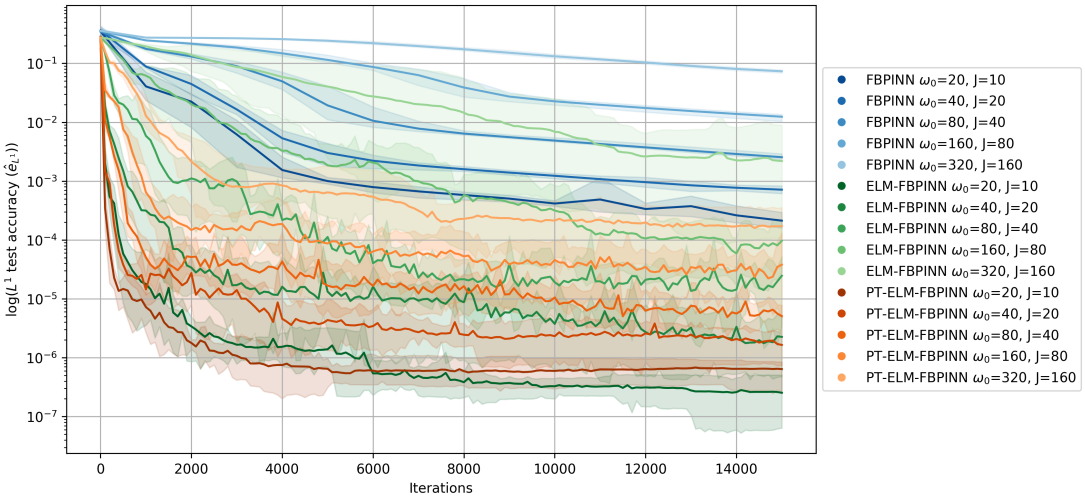


Figure 4.35: Convergence of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying solution frequency, ω_0, and number of subdomains, J.

The final $\hat{e}_{L^1}$ for each model is plotted in Figure 4.36 with exact values reported in Table 4.17. Here it is clear to see that FBPINN has the worst accuracy, while PT-ELM-FBPINN has the best. ELM-FBPINN has still performed adequately for $\omega_0 < 320$, but with a performance gap between the ELM models that is widest at the highest value of ω_0. This demonstrates the potential for the transfer learning of lower frequency pre-trained weights to higher frequency solutions.

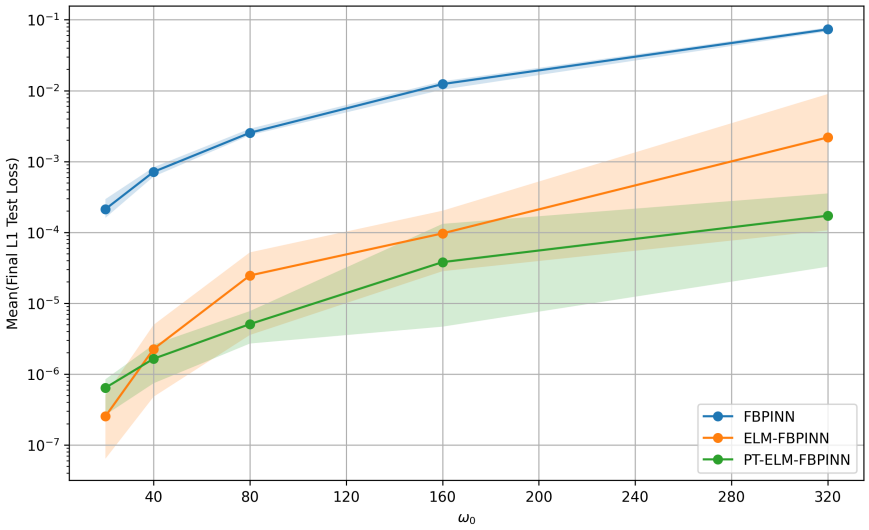


Figure 4.36: Final $\hat{e}_{L^1}$ of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying solution frequency, ω_0, and number of subdomains, J.

ω_0	J	FBPINN	ELM-FBPINN	PT-ELM-FBPINN
20	10	(2.13±1.34)e-04	(2.54±4.42)e-07	(6.41±5.79)e-07
40	20	(7.14±2.17)e-04	(2.26±4.47)e-06	(1.65±1.87)e-06
80	40	(2.54±0.553)e-03	(2.47±4.86)e-05	(5.08±5.04)e-06
160	80	(1.24±0.322)e-02	(9.67±17.4)e-05	(3.8±12.7)e-05
320	160	(7.33±0.918)e-02	(2.19±8.85)e-03	(1.72±3.21)e-04

Table 4.17: Final $\hat{e}_{L^1}$ values for each model at different solution frequencies, ω_0 and number of subdomains, J.

The condition number and rank of the matrix $\mathbf{M}$ against ω_0 are plotted in Figure 4.37 and exact values are reported in Table 4.18. The condition number of the ELM-FBPINN matrix remains relatively stable across all values of solution frequency, with the rank scaling approximately linearly with J. These trends are mirrored in the PT-ELM-FBPINN results, however, the condition number is consistently lower for all $\omega_0 < 320$. Additionally, the rank is improved for all values of ω_0, with all matrices being close to full rank, which is consistent with the reduction in linear dependence of basis functions seen in other tests.

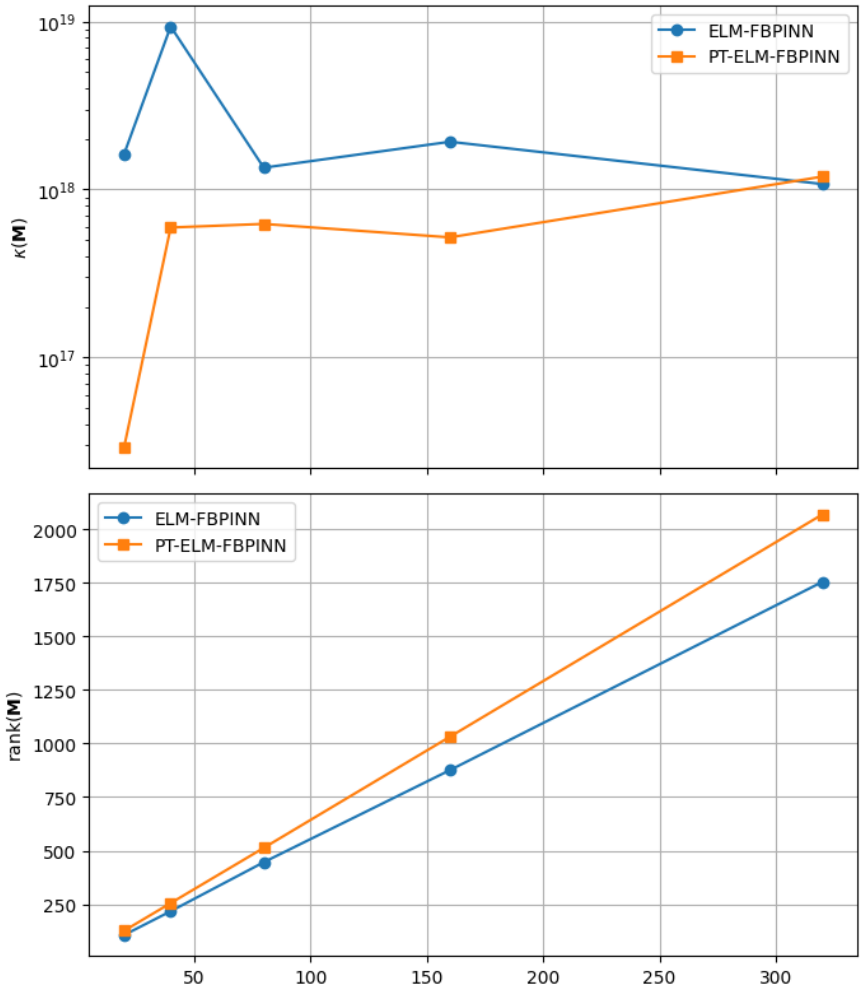


Figure 4.37: Condition number and rank of matrix $\mathbf{M}$ for ELM-FBPINN and PT-ELM-FBPINN when varying solution frequency, ω_0 and number of subdomains, J.

		ELM-FBPINN		PT-ELM-FBPINN	
ω_0	J	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)	$\kappa(\mathbf{M})$	rank($\mathbf{M}$)
20	10	1.63e+18	106	2.93e+16	126
40	20	9.34e+18	217	5.94e+17	255
80	40	1.35e+18	446	6.24e+17	513
160	80	1.92e+18	876	5.19e+17	1032
320	160	1.08e+18	1755	1.19e+18	2070

Table 4.18: Condition numbers and ranks of ELM-FBPINN and Pre-trained ELM-FBPINN matrices for varying solution frequency, ω_0 and number of subdomains J.

In summary, for the 1D damped harmonic oscillator, we find that PT-ELM-FBPINN

is a promising extension of ELM-FBPINN that overcomes some of the issues with expressivity that the randomly-chosen weights of ELM-FBPINN encounter and that consistently outperforms both FBPINN and ELM-FBPINN in terms of accuracy and convergence speed. For example, recall that, when varying the network depth, h, ELM-FBPINN was less stable, and produced less accurate solutions, than FBPINN, because the randomly-chosen weights saturated. With pre-trained weights providing more robust, expressive basis functions, PT-ELM-FBPINN performed significantly better than FBPINN (and, hence, ELM-FBPINN) with deeper layer networks. We note that the one exception to the superior performance of PT-ELM-FBPINN over ELM-FBPINN is for a large number of subdomains J and static solution frequency. In this case, subdomain solutions become simple enough for the networks that the quasi-linear, random basis functions typically generated by ELM-FBPINN may actually be preferred. In all other tests, pre-trained basis functions are significantly more expressive and provide better convergence and accuracy. While the conditioning of the matrix $\mathbf{M}$ is not notably/consistently improved through the pre-training stage, the rank is significantly better for PT-ELM-FBPINN matrices, often being near-full or full rank.

4.4.2 2D Multi-scale Laplace

Baseline

We now evaluate each model on the 2D multi-scale Laplace problem, first establishing baseline results to provide a reference point for subsequent ablation studies. The 2D problem presents a more demanding test, combining the increased dimensionality with a multi-frequency spatial structure which significantly enlarges the effective hypothesis space. Consequently, the networks must be sufficiently expressive to represent the larger solution space, while the optimisation must remain robust under the resulting increase in complexity, particularly in the presence of interacting frequency modes. To ensure sufficient data inside each subdomain, and to guarantee an overdetermined least-squares system for the ELM models, we scale the number of training and test points with the number of basis functions and subdomains thereby preserving a consistent

sampling density across configurations.

The models are trained on $\left(\lceil\sqrt{C+1}\rceil\, J\right)^2$ points and tested on $\left(\lceil\sqrt{30}\rceil\, J\right)^2$ points where $\lceil\cdot\rceil$ denotes the ceiling function, so that both interior coverage and generalisation assessment remain balanced. Each model is trained for 15,000 iterations unless states otherwise to allow direct comparison of convergence behaviour. The PT models are trained on $P = 100$ pre-training solutions, with the coefficients sampled as stated in Section 4.3.2 ensuring diversity within the supervised pre-training stage. We fix $n_\omega = 4$, resulting in frequency components $\omega_1 = 4$, $\omega_2 = 8$, $\omega_3 = 12$, $\omega_4 = 16$. This number of components is chosen to create a solution with sufficient multi-scale features, with increasingly fine oscillations, that prove difficult for global neural networks to represent due to spectral bias, while remaining computationally tractable. The number of subdomains in each axis is $J = 16$, and the number of basis functions is chosen as $C = 16$, which matches the 1D baseline in terms of relative capacity.

The convergence of the PT model is plotted in Figure 4.38. Within 15,000 iterations, the PT model has converged well on the $P = 100$ solutions with a rapid initial reduction in training loss followed by gradual refinement. While there is potential for the PT to have converged further, the approximated solutions are accurate (see Figure 4.39) and visually capture the dominant multi-scale structures. Given that the purpose of pre-training the weights is to better fit the hidden-layer basis to the solution space, while being generic across the solutions, this level of convergence in the baseline is acceptable as it indicates successful feature shaping without overfitting to individual instances.

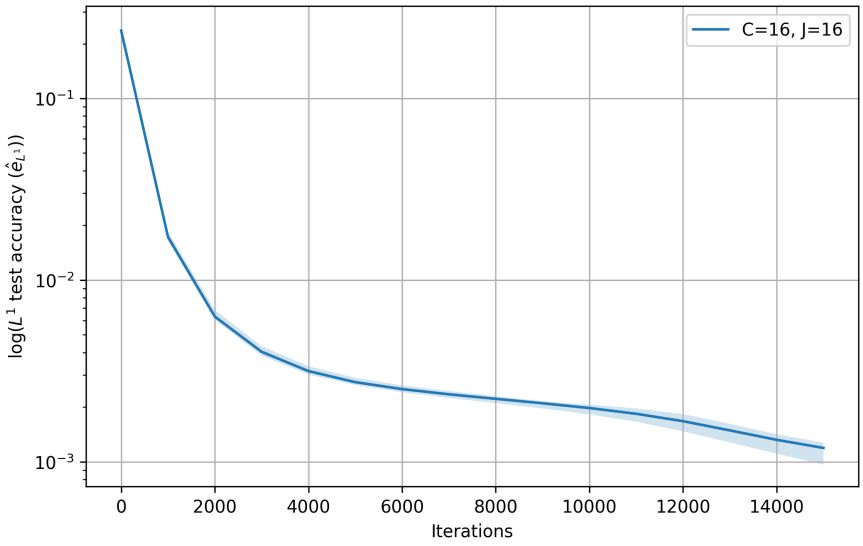


Figure 4.38: Convergence of PT baseline model for the 2D multi-scale Laplace problem.

The convergence of all other models is plotted in Figure 4.40. All models have converged to the solution, demonstrating that the chosen capacity is sufficient for this baseline configuration, with FBPINN and PT-ELM-FBPINN attaining the best accuracy and exhibiting the lowest final $\hat{e}_{L^1}$ values. Both ELM models show rapid convergence in the first $\approx 1,000$ iterations before a more gradual evolution, reflecting the efficiency of the linear least-squares formulation in capturing the dominant solution modes early. The FBPINN model took the full 15,000 iterations to converge fully, and there is some indication that more iterations could result in marginal improvements to accuracy given the persistent, albeit shallow, downward slope near the end of training. The ELM-FBPINN convergence has a very minor downwards trend following the rapid initial decrease suggesting limited additional refinement after the principal components of the solution have been captured, while the PT-ELM-FBPINN appears to have converged fully within the first ≈ 7000 iterations indicating faster stabilisation of the loss.

The exact values of the final $\hat{e}_{L^1}$ are reported in Table 4.19. Here, we note that the performance of FBPINN and PT-ELM-FBPINN is nearly exactly the same, with PT-ELM-FBPINN taking a slightly shorter time to complete the runs on average. That is, ELM-FBPINN exhibits a modest efficiency gain without sacrificing accuracy. It

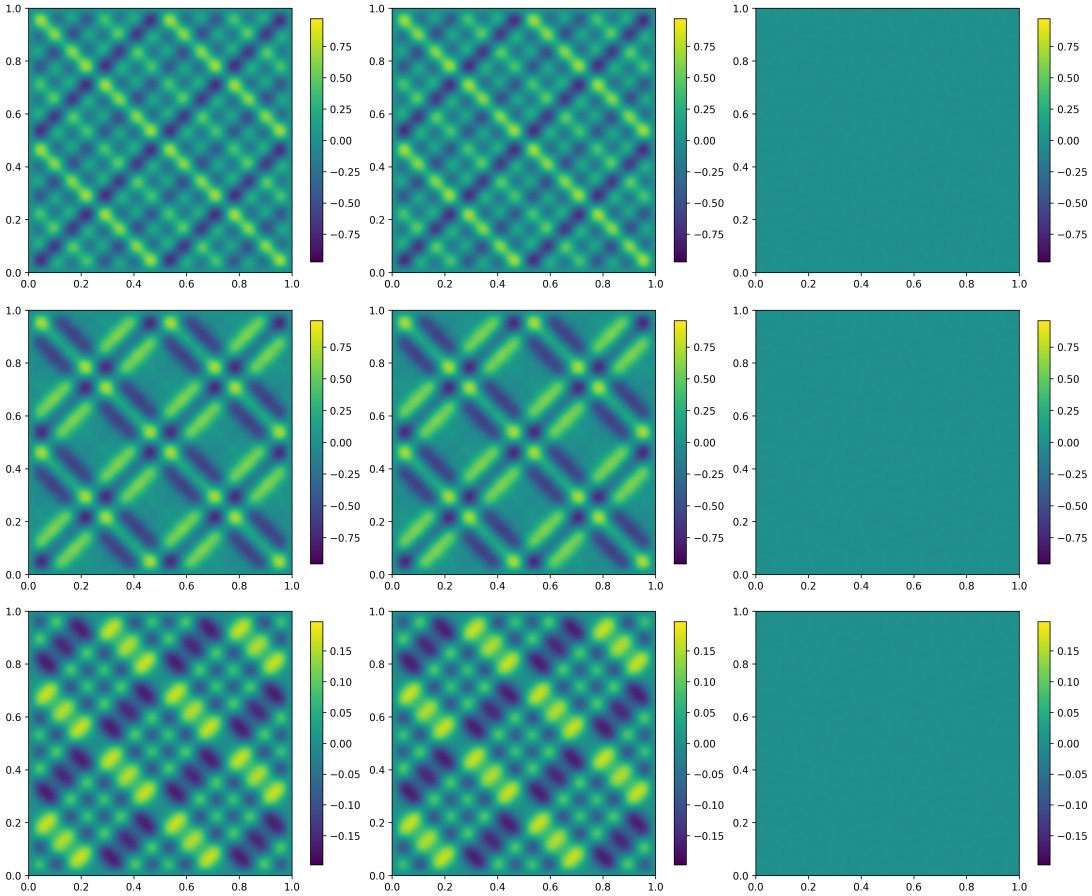


Figure 4.39: PT predictions versus the exact solution for the 2D multi-scale Laplace problem. Left: PT predicted solutions; Middle: exact solutions; Right: Difference.

should be noted that despite this similarity in error, if early stopping was employed for the LSQR solver, PT-ELM-FBPINN would have completed the runs significantly earlier because convergence is effectively reached well before the full iteration budget is exhausted.

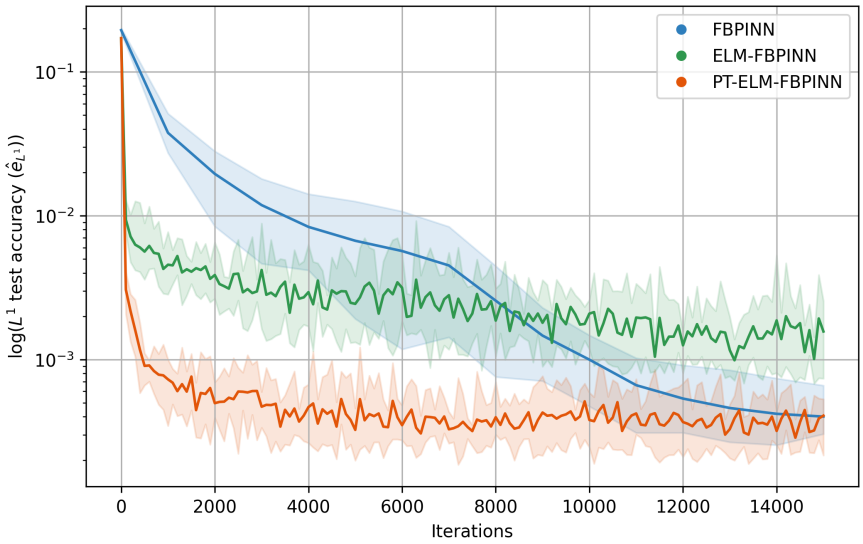


Figure 4.40: Convergence of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN baseline models for the 2D multi-scale Laplace problem.

Model	$\hat{e}_{L^1}$	Time (s)
PT	(1.19±0.313)e-03	281±3.92
FBPINN	(4.02±3.62)e-04	25.3±0.0227
ELM-FBPINN	(1.55±1.84)e-03	21.4±0.111
PT-ELM-FBPINN	(4.09±3.13)e-04	23.5±0.572

Table 4.19: Baseline results for the 2D multi-scale Laplace problem for PT, FBPINN, ELM-FBPINN and PT-ELM-FBPINN. PT has been included for reference and is not a direct comparison of performance against the other models.

From our baseline tests, it is clear that pre-training the hidden-layer weights has allowed PT-ELM-FBPINN to overcome to performance gap between FBPINN and ELM-FBPINN.

Pre-training Iterations

Next, we reuse the PT baseline model settings, saving the weights every $1,000$ iterations to analyse how progressively refined feature spaces influence the downstream linear solve. Each set of saved weights is then used in PT-ELM-FBPINN, providing insight into how the number of PT iterations affect the final linear least-squares solve in the more demanding 2D multi-scale setting. We compare each PT-ELM-FBPINN model, along with an ELM-FBPINN baseline result that uses randomly initialised hidden-layer weights. For earlier PT iterations, the weights are not well fit to the $P = 100$ solutions, but have still been trained partially into the problem space, capturing coarse structural features of the solution manifold. The expectation is that these basis functions are a more generic, but less tailored basis for the test problem, potentially improving robustness but limiting optimal alignment. At later PT iterations, the weights are well fit to the solutions, providing a slightly less generic basis, but one that is well tailored to the test problem and therefore better aligned with the dominant multi-scale structures.

The convergence of each PT-ELM-FBPINN model, along with a baseline ELM-FBPINN model, can be seen in Figure 4.41. Note that the convergence of the PT models can be seen in Figure 4.38. All PT-ELM-FBPINN models outperform the baseline ELM-FBPINN in terms of final accuracy and speed of convergence. Between the PT-ELM-FBPINN models there is little variation in the final accuracy achieved, with all models converging to the same order of magnitude of error, suggesting that moderate levels of pre-training are already sufficient to shape an effective basis. As the PT iterations are increased, the speed of the initial convergence is improved for PT-ELM-FBPINN, which is possibly an indication that the basis functions are better suited to the test problem and more closely aligned with the true solution subspace, thereby accelerating the least-squares optimisation.

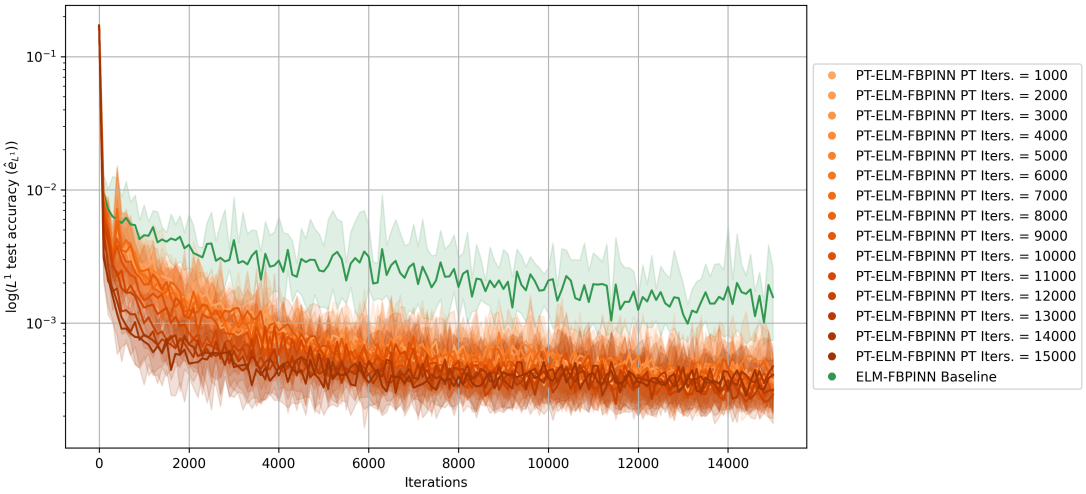


Figure 4.41: Convergence of each PT-ELM-FBPINN model with varying number of PT iterations for the 2D multi-scale Laplace problem. An ELM-FBPINN baseline result has also been plotted for comparison.

The final $\hat{e}_{L^1}$ for PT-ELM-FBPINN as PT iterations are varied is plotted in Figure 4.42, with exact values reported in Table 4.20. The ELM-FBPINN baseline result is included for comparison to highlight the effect of feature pre-training. Here, it is clear that between the PT-ELM-FBPINN models, the final $\hat{e}_{L^1}$ is not affected significantly despite the differing degrees of refinement in the hidden-layer weights. Each model is more accurate than the ELM-FBPINN baseline, but performance remains relatively flat across the models as PT iterations are varied, indicating that even modest pre-training is sufficient to capture the dominant structure of the solution space. The main delineator between models is the speed of convergence, with more PT iterations resulting in faster convergence on the test problem due to improved alignment between the pre-trained basis and the multi-scale features. If early stopping is employed, it is likely that the models initialised with weights trained using more PT iterations would converge in fewer LSQR iterations, thereby reducing computational cost without sacrificing final accuracy.

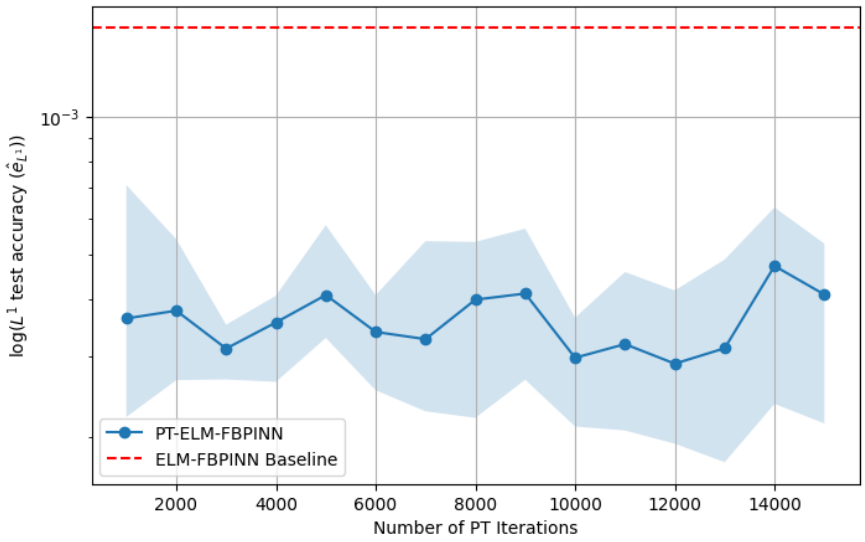


Figure 4.42: Final $\hat{e}_{L^1}$ for PT-ELM-FBPINN as PT iterations are varied for the 2D multi-scale Laplace problem. An ELM-FBPINN baseline has been included for comparison.

PT Iterations	$\hat{e}_{L^1}$
1000	(3.84±4.71)e-04
2000	(3.88±2.4)e-04
3000	(3.2±0.854)e-04
4000	(3.52±1.43)e-04
5000	(4.05±2.95)e-04
6000	(3.32±1.7)e-04
7000	(3.34±2.83)e-04
8000	(4.06±3.03)e-04
9000	(4.23±3.19)e-04
10000	(3.11±1.59)e-04
11000	(3.32±3.6)e-04
12000	(2.89±2.1)e-04
13000	(3.15±3.03)e-04
14000	(4.72±3.9)e-04
15000	(4.08±3.16)e-04
ELM-FBPINN	(1.55±1.84)e-03

Table 4.20: PT-ELM-FBPINN final $\hat{e}_{L^1}$ values for varying numbers of pre-trainer iterations for the 2D multi-scale Laplace problem.

Varying Number of Subdomains

We now examine the effect of varying the number of subdomains per axis, J, which directly controls the degree of domain localisation in the 2D setting. As the domain is two-dimensional, the total number of subdomains is J^2, resulting in a rapid growth in the capacity of the model and least-squares system with increasing J and a corresponding quadratic increase in the size of the linear system for the ELM-based methods.

The convergence of the PT models can be seen in Figure 4.43, with the remaining models' convergence plotted in Figure 4.44. For all models, there is a correlation between increasing J and improved accuracy since smaller subdomains reduce the local approximation difficulty. For PT, all models with $J > 8$ have achieved approximately similar convergence on the $P = 100$ generated training solutions indicating stable supervised fitting once sufficient localisation is introduced. The models with $J < 6$ have not converged on sufficiently accurate solutions. However, from the PT iteration tests we know that this does not inherently indicate that PT-ELM-FBPINN performance will be affected significantly, since the downstream least-squares stage may still compensate partially.

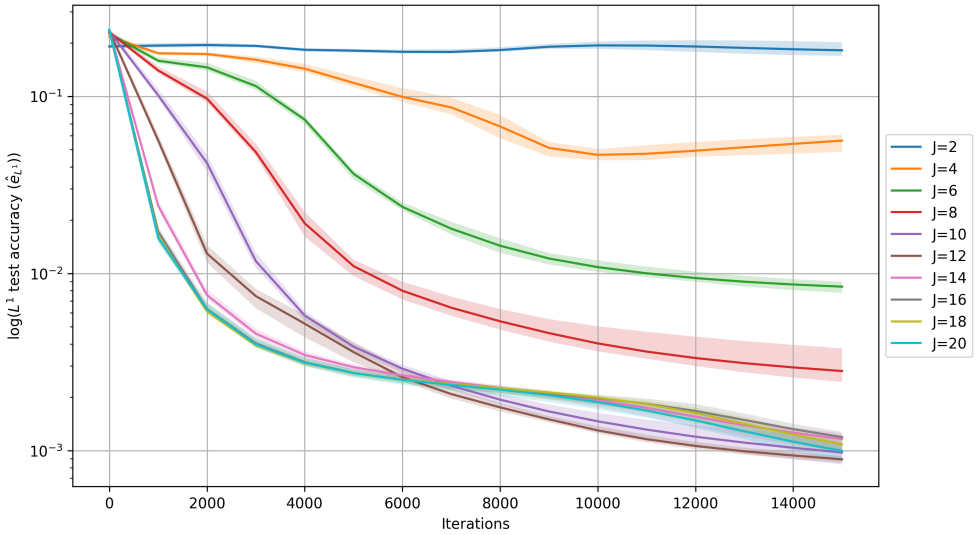


Figure 4.43: Convergence of PT models for varying number of subdomains per axis, J, for the 2D multi-scale Laplace problem.

For FBPINN, ELM-FBPINN, and PT-ELM-FBPINN, the models with $J \leq 4$ do

not converge acceptably, with the ELM models showing some numerical instability due to the increased oscillatory content within each large subdomain. This indicates that the local subdomain solutions with small J are too high frequency for the networks to overcome spectral bias, particularly when localisation is insufficient. For $J > 4$, the FBPINN models have all converged to acceptable errors, with the results being the most consistent across all tests. This demonstrates robustness of gradient-based optimisation under increasing localisation. ELM-FBPINN only converges acceptably for $J \geq 10$, with lower values of J resulting in relatively large final $\hat{e}_{L^1}$, which reflects the limited adaptability of fixed random features in wider subdomains. Convergence speed is improved over the FBPINN for $J > 10$, however, all FBPINN models achieve better final accuracy across the runs, highlighting a trade-off between speed and precision. PT-ELM-FBPINN shows some improvement over ELM-FBPINN for lower values of J, consistently outperforming in both accuracy and convergence for $J \geq 6$, but still results in a worse precision when compared to FBPINN in the moderately localised regime. However, for $J \geq 12$, PT-ELM-FBPINN achieves similar precision to the FBPINN models, with significantly faster convergence, suggesting that supervised feature shaping becomes increasingly effective as subdomain widths decrease.

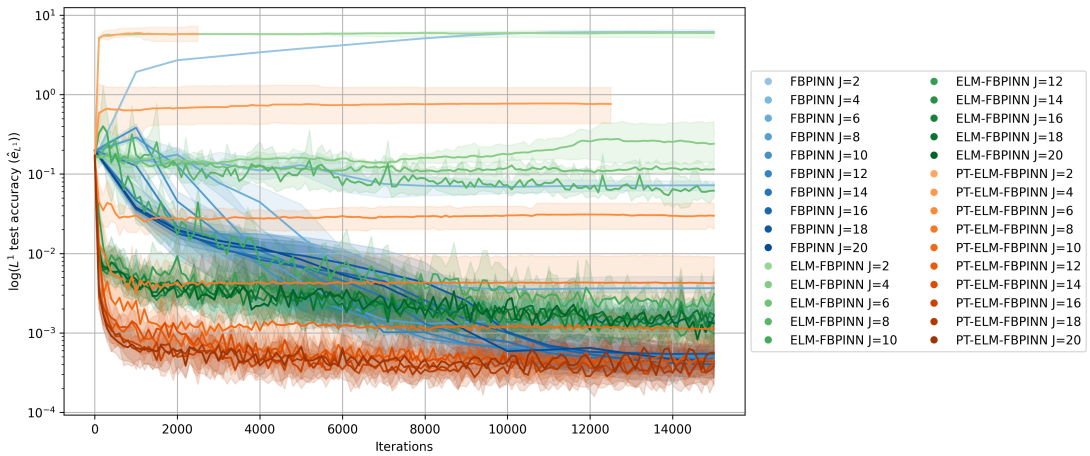


Figure 4.44: Convergence of FBPINN, ELM-FBPINN and PT-ELM-FBPINN for varying number of subdomains, J, for the 2D multi-scale Laplace problem.

The final $\hat{e}_{L^1}$ for all models is plotted in Figure 4.45, and exact values are reported

in Table 4.21. These results further demonstrate the close results of FBPINN and PT-ELM-FBPINN at the higher values of J. This indicates that the hidden-layer basis being more expressive and adapted to the problem domain is beneficial as the localisation becomes more refined. For the higher values of J, PT-ELM-FBPINN is the most attractive model, with convergence time potentially being less than half that of the FBPINN. However, at lower values, FBPINN is more robust, with the flexibility of gradient-descent based optimisation proving more efficient with wider subdomains and more iterations.

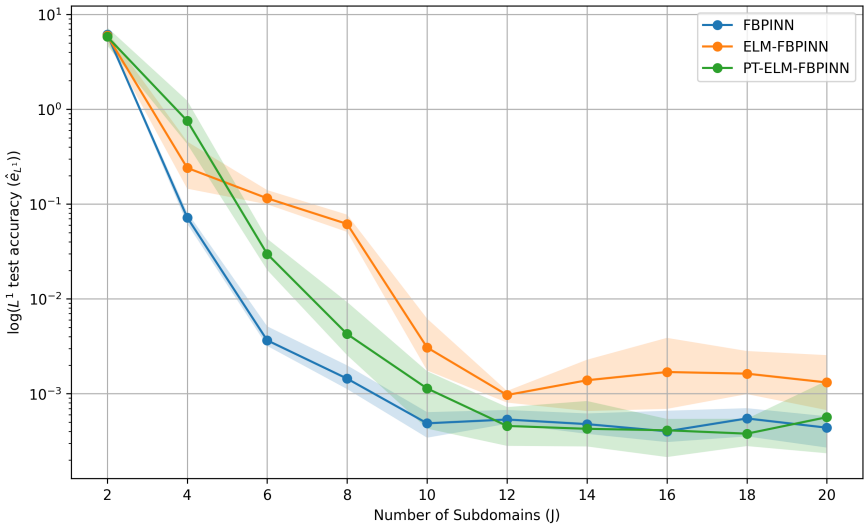


Figure 4.45: Final $\hat{e}_{L^1}$ for varying number of subdomains per axis, J, for the 2D multi-scale Laplace problem.

J	FBPINN	ELM-FBPINN	PT-ELM-FBPINN
2	(6.15±0.264)e+00	(5.96±1.43)e+00	(5.82±2.67)e+00
4	(7.19±1.99)e-02	(2.41±3.12)e-01	(7.54±7.97)e-01
6	(3.65±1.93)e-03	(1.16±0.464)e-01	(3.01±2.31)e-02
8	(1.44±0.864)e-03	(5.72±2.27)e-02	(4.26±6.65)e-03
10	(5.01±2.82)e-04	(1.97±0.694)e-03	(1.15±1.31)e-03
12	(5.31±1.95)e-04	(1.04±0.688)e-03	(4.54±4.33)e-04
14	(4.79±2.4)e-04	(1.57±2.88)e-03	(4.19±5.58)e-04
16	(4.02±3.52)e-04	(1.46±1.25)e-03	(4.07±3.2)e-04
18	(5.48±3.49)e-04	(1.45±1.97)e-03	(3.81±2.63)e-04
20	(4.38±3.03)e-04	(1.3±1.81)e-03	(5.62±11.1)e-04

Table 4.21: Final $\hat{e}_{L^1}$ values for varying number of subdomains per axis, J, for the 2D multi-scale Laplace problem.

Varying Number of Basis Functions

We now fix the number of subdomains per axis again, and vary the number of basis functions, C, while keeping the localisation structure unchanged. This study investigates the effect of increasing representational capacity of the networks, with a fixed domain decomposition so that improvements can be attributed primarily to feature richness rather than spatial refinement.

Convergence plots for PT can be seen in Figure 4.46 and for all other models in Figure 4.47. The PT models have all attained a similar final error, with the exception of $C = 2$, likely due to the lack of representational capacity to adequately span the multi-scale solution space. There is also an increase in the initial speed of convergence as C increases. However, all models appear to not have fully converged within the 15,000 iterations, suggesting that additional training could yield marginal refinement.

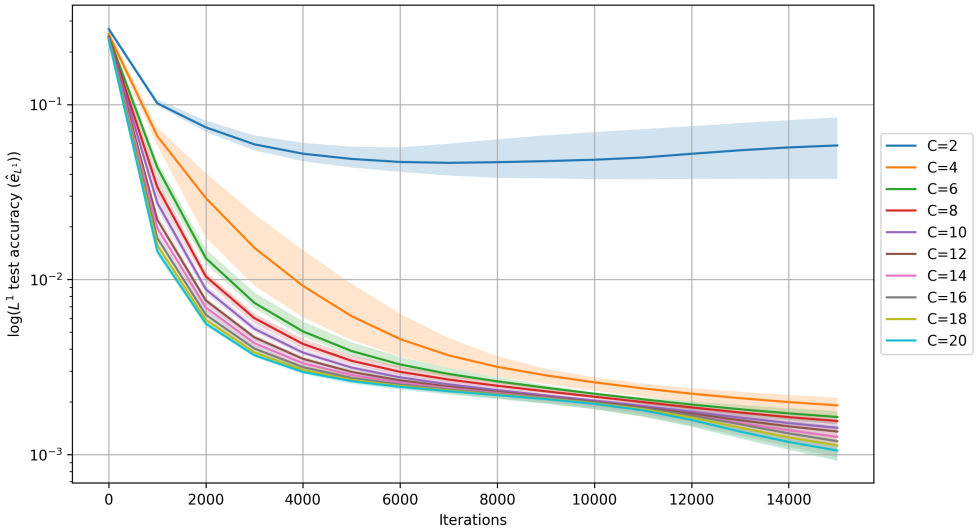


Figure 4.46: Convergence for PT models for varying number of basis functions, C, for the 2D multi-scale Laplace problem.

For all the other models, convergence speed and accuracy increase with C as additional basis functions enhance expressivity. The ELM models both exhibit instability at $C = 2$, with neither model completing the full 15,000 iterations due to rapid deterioration of the least-squares residual. PT-ELM-FBPINN also exhibits this instability for $C = 4$, while ELM-FBPINN does not, indicating sensitivity of the pre-trained basis

at very low capacities. It should be noted that ELM-FBPINN would have likely also ended in a similar number of iterations if early stopping was employed, with the model accuracy progressively worsening after ≈ 4000 iterations as over-iteration amplifies numerical errors. Both ELM models exhibit this behaviour for a number of values of C, indicating that employing early stopping is a necessity for the linear least-squares solver in this regime to prevent divergence and preserve optimal solutions. Despite this behaviour, for $C \geq 10$, PT-ELM-FBPINN converges most rapidly, and generally to the best accuracy across the tested configurations. At the largest values of C, FBPINN attains similar final errors, but this requires the full 15,000 iterations. By comparison, PT-ELM-FBPINN converges in $\approx 7,000$ iterations, demonstrating a clear efficiency advantage of supervised feature shaping in higher-capacity settings.

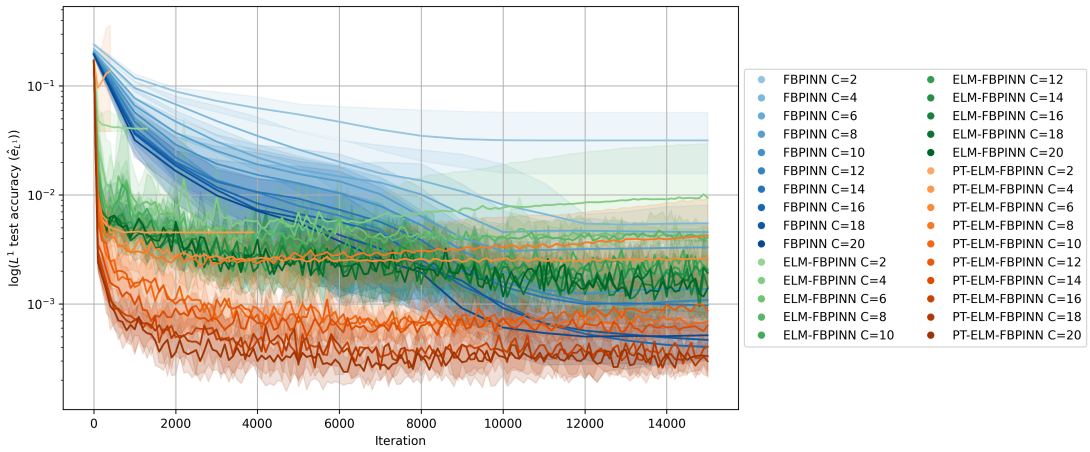


Figure 4.47: Convergence for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of basis functions, C, for the 2D multi-scale Laplace problem.

The results are further demonstrated in Figure 4.48 and Table 4.22. Here it is clearer to see that PT-ELM-FBPINN is the most accurate model for the majority of the tests, however, FBPINN is not significantly worse. Both models are a notable improvement over ELM-FBPINN at the higher values of C, indicating that the more complex, linearly independent basis functions of FBPINN and PT-ELM-FBPINN allow for more expressive power compared to the randomly initialised, quasi-linear weights of ELM-FBPINN.

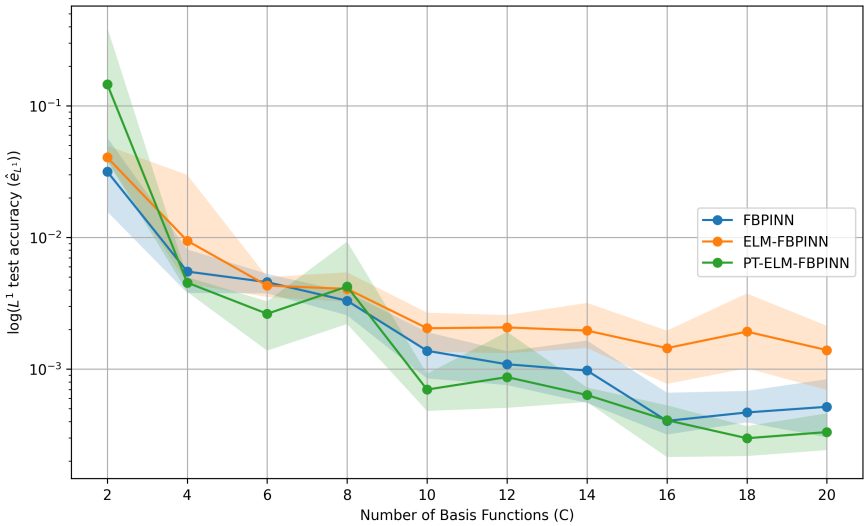


Figure 4.48: Final $\hat{e}_{L^1}$ for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of basis functions, C, for the 2D multi-scale Laplace problem.

C	FBPINN	ELM-FBPINN	PT-ELM-FBPINN	PT
2	(3.16±4.16)e-02	(4.05±1.13)e-02	(1.46±3.56)e-01	(5.83±4.68)e-02
4	(5.51±4.32)e-03	(9.47±25.2)e-03	(4.53±1.13)e-03	(1.91±0.473)e-03
6	(4.57±1.53)e-03	(4.32±1.62)e-03	(2.63±1.91)e-03	(1.64±0.279)e-03
8	(3.3±1.29)e-03	(4.05±2.01)e-03	(4.23±7.11)e-03	(1.55±0.149)e-03
10	(1.38±1.06)e-03	(1.81±0.935)e-03	(6.74±3.33)e-04	(1.42±0.18)e-03
12	(1.09±0.614)e-03	(1.75±0.855)e-03	(8.71±14.2)e-04	(1.35±0.189)e-03
14	(9.73±11)e-04	(1.83±1.08)e-03	(6.31±1.75)e-04	(1.26±0.357)e-03
16	(3.99±3.47)e-04	(1.54±1.79)e-03	(4.09±3.21)e-04	(1.19±0.312)e-03
18	(4.7±2.91)e-04	(1.97±2.84)e-03	(3.02±1.67)e-04	(1.13±0.337)e-03
20	(5.17±5.28)e-04	(1.61±1.07)e-03	(3.31±2.11)e-04	(1.05±0.226)e-03

Table 4.22: Final $\hat{e}_{L^1}$ values for varying number of basis functions, C, for the 2D multi-scale Laplace problem. PT model results are included for reference and are not a direct comparison.

Varying Depth of Hidden Layers With Number Of Subdomains

Now we evaluate how each model performs with a varying number of hidden layers, h in the 2D multi-scale setting. We inversely vary the number of subdomains per axis, J, to isolate the effect of network depth changing by preserving the total network parameters. Consequently, improvements or degradations can be attributed primarily to architectural depth. Both ELM models employ early stopping to address numerical in-

stability seen when increasing h since deeper architectures tend to amplify conditioning issues in the linear solve.

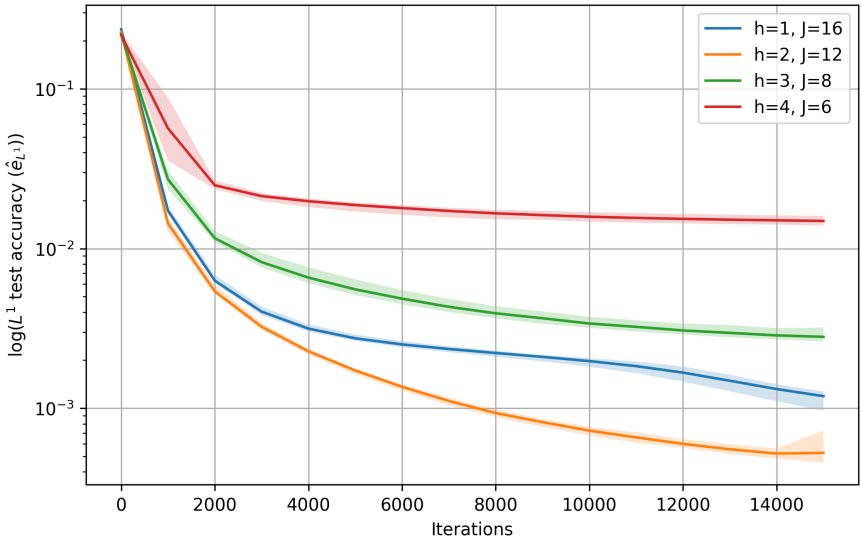


Figure 4.49: Convergence for PT models with varying number of hidden layers, h, and number of subdomains, J, for the 2D multi-scale Laplace problem.

The convergence for the PT models can be seen in Figure 4.49. The best performing model for both accuracy and initial convergence speed is $h = 2$, suggesting that there is some benefit to the supervised optimisation of the solutions when moderate depth is combined with sufficient localisation. The wider domains when $J = 12$ forces the basis functions to learn more complex features than at $J = 16$, with the additional layer introducing more non-linearity as well, thereby enriching the feature space. For $h = 3$ and $h = 4$, accuracy reduces, while convergence speed still remains relatively fast, indicating that optimisation remains stable even as representational balance shifts. This is likely due to the number of subdomains becoming too small, making it difficult for the basis functions to learn the subdomain solutions as each local problem becomes increasingly oscillatory.

The convergence of FBPINN, ELM-FBPINN and PT-ELM-FBPINN can be seen in Figure 4.50. All models perform best at $h = 1$ and $h = 2$, with the shallow networks the most accurate, confirming that increased depth does not automatically translate into improved performance. While FBPINN and PT-ELM-FBPINN have converged

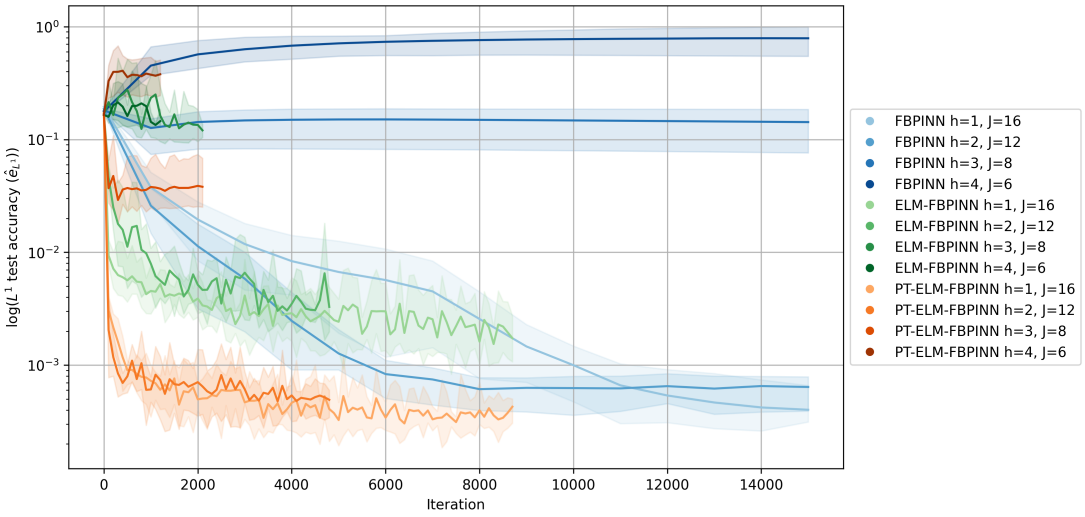


Figure 4.50: Convergence for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN with varying number of hidden layers, h, and number of subdomains per axis, J, for the 2D multi-scale Laplace problem.

to the same order of magnitude of error, PT-ELM-FBPINN has significantly faster convergence, highlighting the efficiency of pre-shaped features in the linear stage.

All models fail to converge acceptably for $h = 4$, which is likely due, in part, to the over-saturation of hidden-layer weights, and also to the reduced domain decomposition resulting in high frequency subdomain solutions that are difficult to approximate. Both FBPINN and ELM-FBPINN also perform poorly in terms of accuracy and convergence for $h = 3$, however, while not fully within acceptable accuracy range, the PT-ELM-FBPINN model improves performance relative to its non-pre-trained counterpart.

The predicted solutions at $h = 3$ for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN can be seen in Figure 4.51. Despite some noticeable difference, the PT-ELM-FBPINN has reproduced the solution with the highest accuracy among the three architectures at this depth.

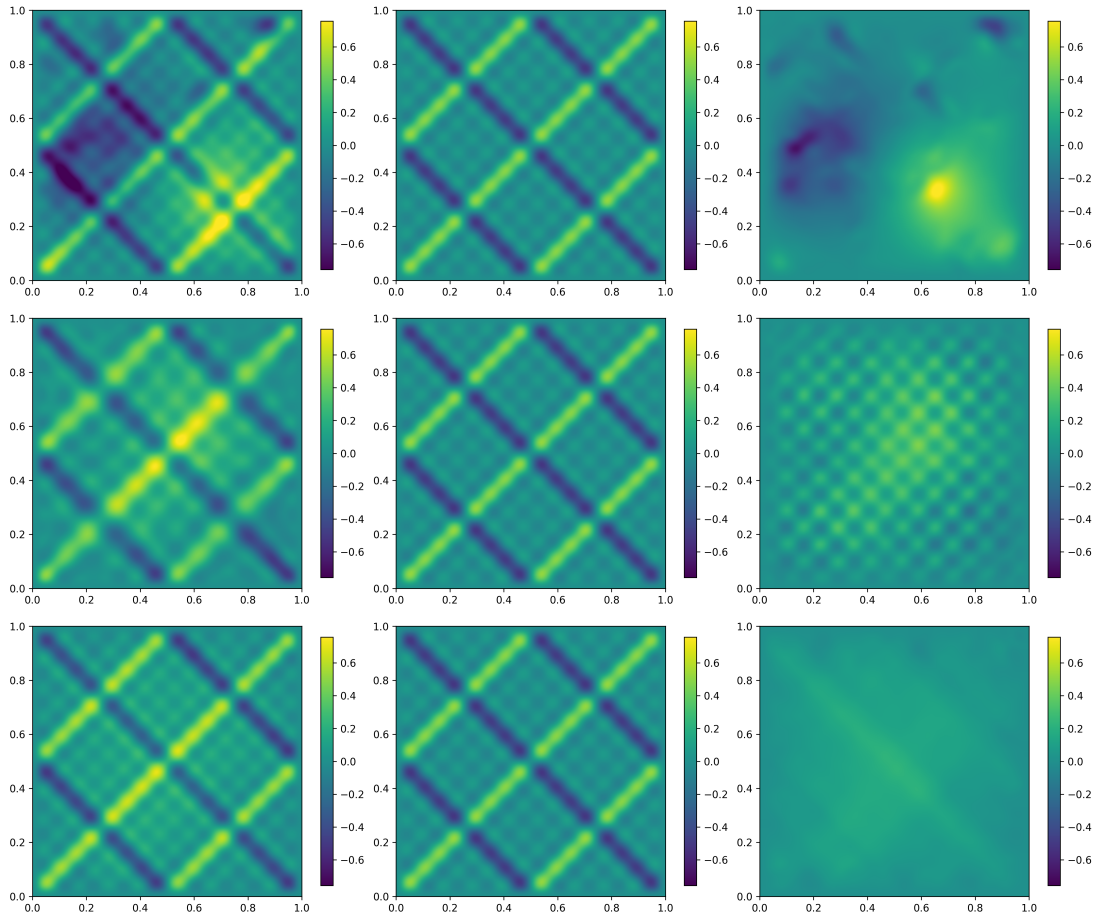


Figure 4.51: Predicted solutions for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN with $h = 3$ hidden layers and $J = 8$ subdomains per axis for the 2D multi-scale Laplace problem. Top row: FBPINN; Middle row: ELM-FBPINN; Bottom row: PT-ELM-FBPINN. Left column: Predicted solution; Middle column: Exact solution; Right column: Difference.

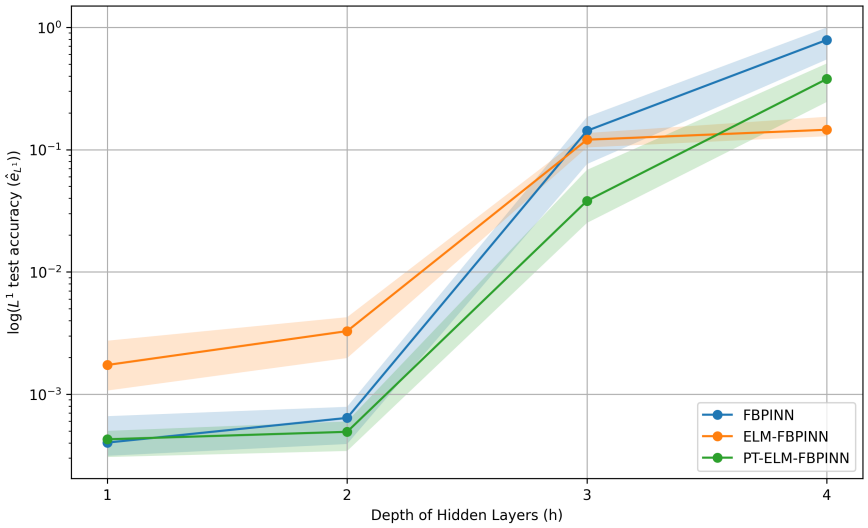


Figure 4.52: Final $\hat{e}_{L^1}$ for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN with varying number of hidden layers, h, and number of subdomains per axis, J, for the 2D multi-scale Laplace problem.

The final $\hat{e}_{L^1}$ for each model is plotted in Figure 4.52 and is reported in Table 4.23. The correlation between reduced accuracy and increasing h is clear for all models, indicating that the added complexity from deeper layers is not sufficient to overcome the increase in subdomain solution complexity as J reduces and may instead exacerbate saturation effects. Pre-training the weights improves performance over the randomly initialised weights, with accuracy more in line with FBPINN at $h = 1$ and $h = 2$. Moreover, the best-performing model at $h = 3$ is PT-ELM-FBPINN, illustrating the stabilising effect of supervised feature shaping. This demonstrates the requirement for a more expressive hidden-layer basis for both shallow and deeper layer networks, particularly in multi-scale problems. While ELM-FBPINN has achieved the best accuracy at $h = 4$, it should be noted that all models exhibit numerical instability at this depth, making this configuration unreliable in practice.

h	J	FBPINN	ELM-FBPINN	PT-ELM-FBPINN	PT
1	16	(4.01±3.52)e-04	(1.48±0.968)e-03	(4.26±2.21)e-04	(1.19±0.312)e-03
2	12	(6.34±4.48)e-04	(5.9±14.4)e-03	(4.93±2.45)e-04	(4.92±0.731)e-04
3	8	(1.43±1.09)e-01	(1.27±0.931)e-01	(3.77±4.1)e-02	(2.8±0.464)e-03
4	6	(8.05±3.88)e-01	(1.43±0.494)e-01	(3.78±2.6)e-01	(1.49±0.208)e-02

Table 4.23: Final $\hat{e}_{L^1}$ values for varying number of hidden layers, h and varying number of subdomains, J, for the 2D multi-scale Laplace problem. PT results are for reference only, and are not a direct comparison.

Varying Depth of Hidden Layers With Number Of Basis Functions

We now fix the number of subdomains per axis, and inversely decrease the number of basis functions, C, as the number of hidden layers, h, is increased so that depth increases while per-layer capacity decreases. Again, this isolates the effect of varying the depth of the network, while keeping the number of parameters stable and avoids confounding effects from increased model size.

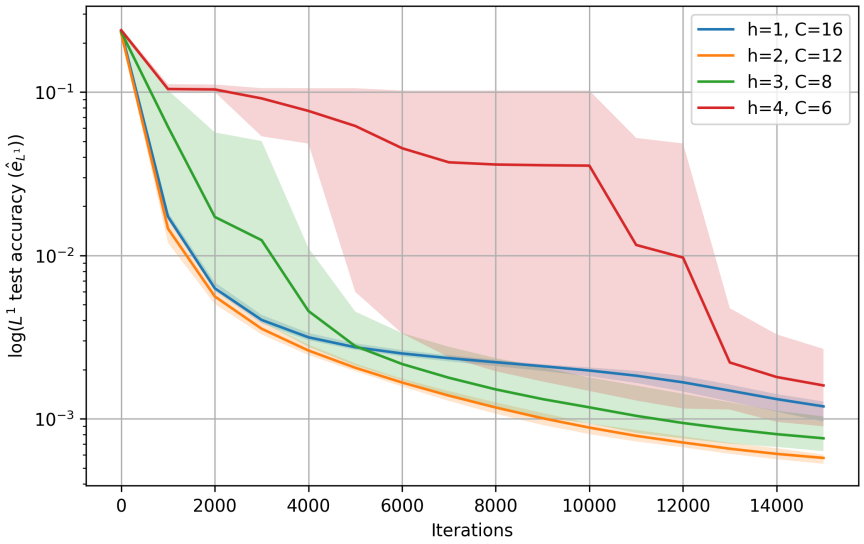


Figure 4.53: Convergence of PT models for varying number of hidden layers, h, and number of basis functions, C, for the 2D multi-scale Laplace problem.

Instead of reducing the domain decomposition capacity, we instead reduce the representational capacity in each subdomain, thereby forcing deeper architectures to compensate with more complex hierarchical features. To address the numerical instabilities,

we employ early stopping for the ELM models as in previous tests to prevent divergence of the least-squares solver.

The convergence of the PT models is plotted in Figure 4.53. For models with $h \leq 3$, the average convergence profile and final error are similar, with $h = 4$ showing significantly slower convergence, and the largest final error indicating diminishing returns from increased depth. The best performing model is $h = 2$ with $C = 12$, which suggests that a slight reduction in the number of basis functions, while increasing the linear independence between them, has a positive effect on the ability to fit the $P = 100$ solutions. This demonstrates the possibility of a beneficial trade-off between feature diversity and capacity. The same behaviour is also observed when $h = 3$, where the final error is also improved over the shallow network, albeit with more variance in the speed of convergence, showing a level of stability sensitivity as h increases particularly in the optimisation dynamics.

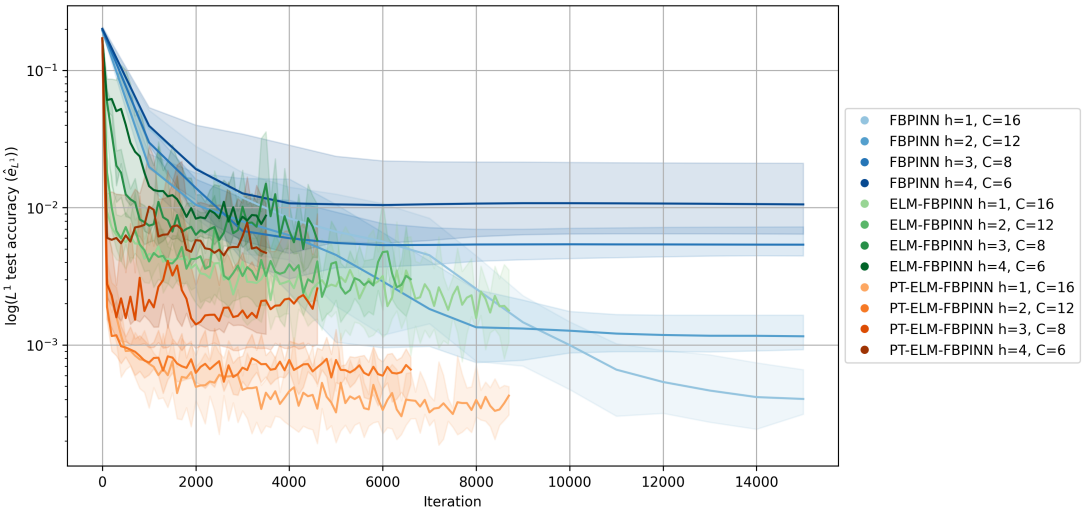


Figure 4.54: Convergence of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of hidden layers, h, and number of basis functions, C, for the 2D multi-scale Laplace problem.

The convergence of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN models are plotted in Figure 4.54. All models have converged, with a degradation in performance as h increases and C decreases, reflecting the reduced expressivity per subdomain. The increase in hidden-layer basis function complexity introduced through deeper layer

network architecture is not sufficient to overcome the reduction in representational capacity, highlighting the limitations of depth without adequate width. However, the pre-training of hidden-layer weights has had a positive effect on convergence, with PT-ELM-FBPINN models outperforming ELM-FBPINN for all tests in terms of both speed and final accuracy. FBPINN has similar performance for the shallow network, $h = 1$, but is also worse than PT-ELM-FBPINN at all other values of h, demonstrating the advantage of supervised feature shaping in deeper configurations.

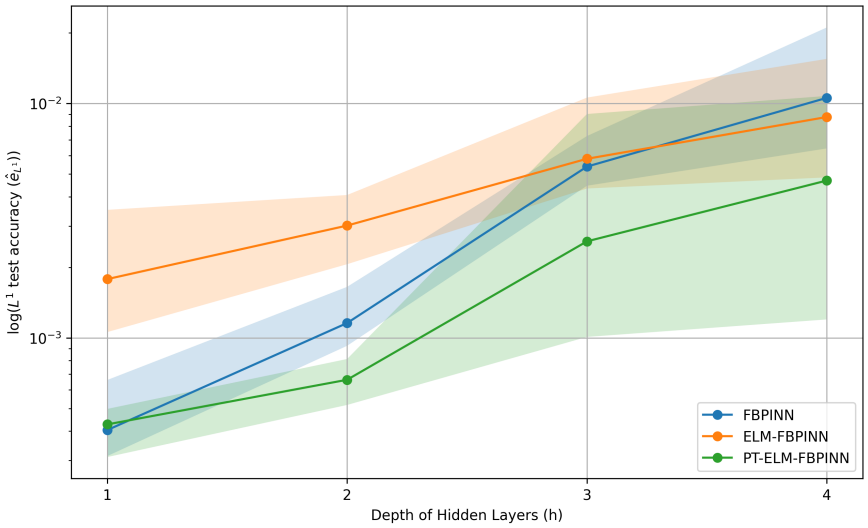


Figure 4.55: Final $\hat{e}_{L^1}$ of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying number of hidden layers, h, and number of basis functions, C, for the 2D multi-scale Laplace problem.

The final $\hat{e}_{L^1}$ for all models is reported in Table 4.24 and plotted for FBPINN, ELM-FBPINN and PT-ELM-FBPINN in Figure 4.55. The precision improvement of PT-ELM-FBPINN over ELM-FBPINN is demonstrated clearly for all values of h, which confirms the robustness of pre-training across architectural variations. The increased variance of the PT-ELM-FBPINN models as h increases shows that despite the more expressive basis functions, the reduction in network capacity is less stable overall and deeper configurations remain sensitive to optimisation and conditioning effects.

h	C	FBPINN	ELM-FBPINN	PT-ELM-FBPINN	PT
1	16	(3.99±3.5)e-04	(1.92±0.948)e-03	(4.33±1.93)e-04	(1.19±0.312)e-03
2	12	(1.16±0.72)e-03	(2.86±2.14)e-03	(6.62±3.3)e-04	(5.68±0.74)e-04
3	8	(5.37±2.79)e-03	(5.57±5.16)e-03	(2.78±9.21)e-03	(7.58±3.95)e-04
4	6	(1.06±1.46)e-02	(9.19±10.8)e-03	(4.7±9.64)e-03	(1.6±1.78)e-03

Table 4.24: Final $\hat{e}_{L^1}$ values and average run time for varying number of hidden layers, h, with decreasing number of basis functions, C, for the 2D multi-scale Laplace problem.

Transfer Learning Study

Finally, we investigate the potential for pre-training simpler, lower frequency solutions, and then solving more difficult problems to assess cross-scale generalisation in the 2D multi-scale setting. Specifically, we vary the difficulty of the problem by increasing or reducing the total number of frequency components, n_ω, and scaling the number of subdomains per axis, J, to stabilise the subdomain solution frequency as much as possible, so that increases in global complexity are partially offset by additional localisation. The pre-trainer is trained on $n_\omega = 4$ frequency components, with the same scaling coefficient schedule as all other tests, providing a mid-range reference problem. The FBPINN, ELM-FBPINN, and PT-ELM-FBPINN are then tested on $n_\omega = 1, \ldots, 7$ frequency components, with $\omega_1 = 4$, $\omega_2 = 8$, $\omega_3 = 12$, $\omega_4 = 16$, $\omega_5 = 20$, $\omega_6 = 24$, and $\omega_7 = 32$, thereby progressively increasing oscillatory complexity.

The convergence of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN is plotted in Figure 4.56. Both ELM models initially converge significantly faster than FBPINN, with PT-ELM-FBPINN being the fastest to plateau in accuracy. This reflects the efficiency of the least-squares stage when the feature space is well aligned. FBPINN achieves the best accuracy for $n_\omega = 1$, but the accuracy generally reduces as the number of frequency components increases, resulting in the $n_\omega = 7$ models performing worst because of the increasing difficulty of representing multiple interacting modes. ELM-FBPINN has the lowest accuracy for $n_\omega \leq 6$, but has more consistent results than FBPINN since its fixed feature space avoids some optimisation variability. The convergence of ELM-FBPINN models is also initially fast, but is followed by a longer period of gradual decline indicating limited late-stage refinement. PT-ELM-FBPINN

models are also more consistent than FBPINN, generally appearing to achieve the lowest final accuracies particularly in the higher-frequency regimes.

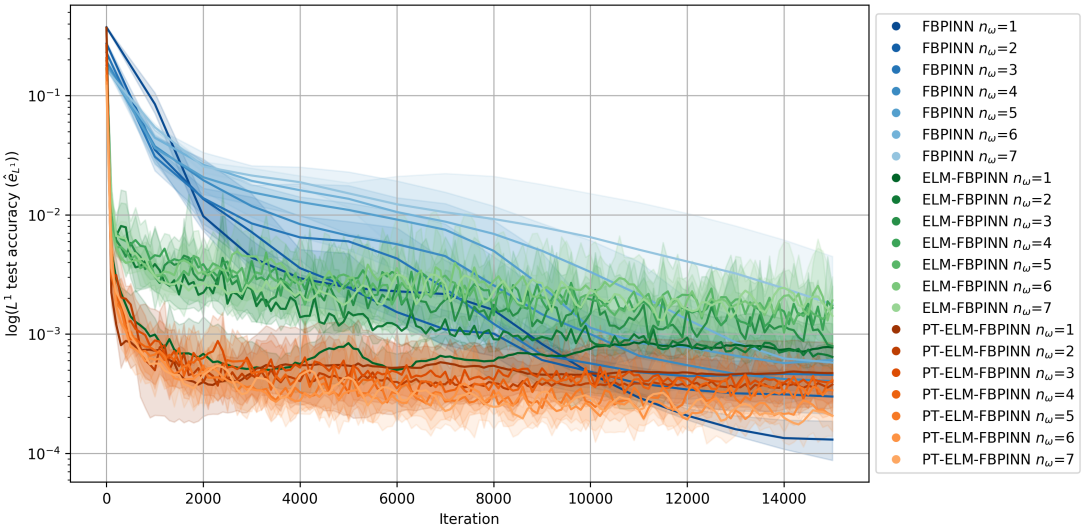


Figure 4.56: Convergence for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying n_ω and number of subdomains per axis, J, for the 2D multi-scale Laplace problem.

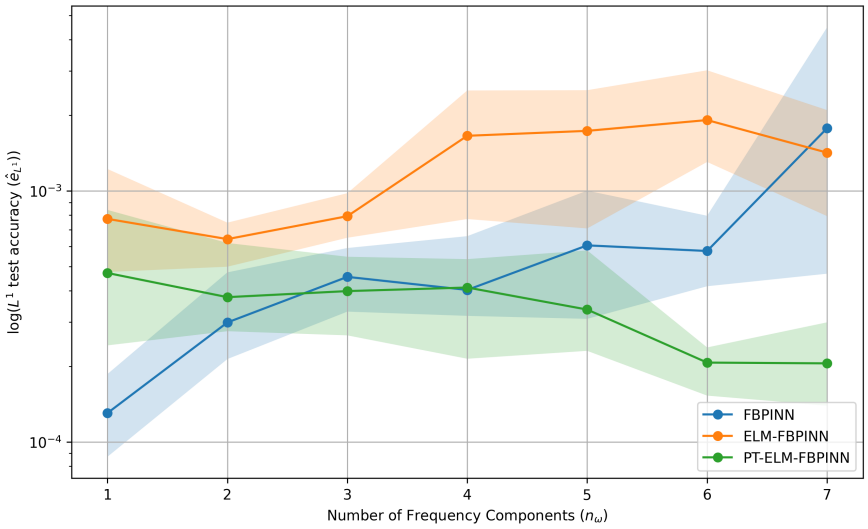


Figure 4.57: Final $\hat{e}_{L^1}$ for FBPINN, ELM-FBPINN, and PT-ELM-FBPINN for varying n_ω and number of subdomains per axis, J, for the 2D multi-scale Laplace problem.

The final $\hat{e}_{L^1}$ for all models as n_ω and J are varied is plotted in Figure 4.57, with exact values reported in Table 4.25. PT-ELM-FBPINN achieves the lowest final $\hat{e}_{L^1}$

values for $n_\omega > 4$ indicating improved robustness as frequency complexity increases. For $1 < n_\omega < 4$, the performance of PT-ELM-FBPINN is similar to FBPINN, and only at higher frequency components does the performance gap between the models appear, suggesting that the benefit of pre-training becomes more pronounced in harder regimes. The lowest accuracy models are consistently ELM-FBPINN models, and the difference between the maximum and minimum error values is negligibly different to FBPINN showing comparable variability. PT-ELM-FBPINN has the most consistent performance, with the difference between maximum and minimum error values being an order of magnitude smaller than that of the other models. This demonstrates improved stability under problem scaling. The results indicate that for the 2D multi-scale Laplace problem, a more expressive hidden-layer basis is preferable due to the improved precision particularly when multiple frequency components interact. Pre-training weights on $n_\omega = 4$ does not notably improve performance for test problems with fewer frequency components, especially when comparing PT-ELM-FBPINN to FBPINN results since simpler problems can already be handled effectively by gradient-based optimisation. However, for higher values of n_ω, PT-ELM-FBPINN is the most accurate, and has the most stable performance as the problem difficulty is scaled highlighting the scalability advantage of supervised feature shaping in complex multi-scale regimes.

n_ω	FBPINN	ELM-FBPINN	PT-ELM-FBPINN
1	(1.33±1)e-04	(7.8±7.79)e-04	(4.7±5.71)e-04
2	(2.97±2.57)e-04	(6.75±2.98)e-04	(3.51±2.28)e-04
3	(4.52±2.62)e-04	(8.38±8.29)e-04	(3.99±3.07)e-04
4	(3.98±3.6)e-04	(1.86±2.17)e-03	(4.1±3.29)e-04
5	(6.06±6.87)e-04	(1.7±1.53)e-03	(3.35±3.46)e-04
6	(5.8±3.83)e-04	(1.86±1.79)e-03	(2.07±0.841)e-04
7	(1.78±4.01)e-03	(1.37±1.42)e-03	(2.06±1.61)e-04
Max–Min	1.65e-03	1.19e-03	2.64e-04

Table 4.25: Final $\hat{e}_{L1}$ values and average run time for varying n_ω and number of subdomains, J, for the 2D multi-scale Laplace problem.

4.5 Discussion and Conclusions

In this chapter, we conducted a systematic comparison of FBPINN, ELM-FBPINN, and PT-ELM-FBPINN across extensive ablation studies for both the 1D damped harmonic oscillator and the 2D multi-scale Laplace problem. The experiments were carefully designed to isolate the effects of domain decomposition (J), representational capacity (C), hidden-layer depth (h), solution frequency content (n_ω), and pre-training configuration (number of iterations and number of pre-training samples P). Performance was assessed through convergence behaviour, numerical stability of the least-squares system, conditioning and rank properties of the matrix $\mathbf{M}$, and final solution accuracy.

Core Empirical Findings

Across both 1D and 2D settings, PT-ELM-FBPINN consistently exhibits significantly faster initial convergence than both FBPINN and ELM-FBPINN. This improvement is systematic and robust across architectural and problem variations. The key mechanisms are clear: Pre-training performs a feature space approximation step, in turn shaping the hidden-layer basis toward the solution manifold before freezing. The subsequent LSQR solve then operates in a better-conditioned, better-aligned feature space.

Thus, pre-training does not merely provide a better initialisation — it fundamentally alters the geometry of the least-squares problem.

1D Behaviour: Expressivity vs. Local Linearity

In the 1D studies, PT-ELM-FBPINN generally achieves the best trade-off between convergence speed, stability, and accuracy. Its advantages are particularly visible in regimes where subdomains are moderately wide, resulting in local solutions exhibiting strong oscillatory structure, in which random ELM features are insufficiently expressive.

However, when the number of subdomains becomes very large, the local problems become nearly linear. In this regime, the quasi-linear random features of ELM-FBPINN are already well aligned with the target space, and the advantage of pre-training diminishes. This is an important observation: the benefit of pre-training is directly

proportional to the nonlinearity of the local solution manifold.

Even in regimes where it does not improve accuracy, PT-ELM-FBPINN demonstrates improved robustness, mitigating hidden-layer weight saturation effects observed in deeper or poorly conditioned ELM-FBPINN models.

2D Behaviour: Feature Flexibility Becomes Critical

The 2D multi-scale Laplace problem provides a more stringent test. Increased dimensionality and interacting frequency components enlarge the effective solution manifold substantially. Here, the flexibility of the hidden-layer basis becomes essential. FBPINN performs better in 2D than in 1D relative to ELM-FBPINN, highlighting the importance of adaptive hidden-layer weights. ELM-FBPINN remains competitive in convergence speed but consistently lags in final accuracy. PT-ELM-FBPINN either matches or closes the performance gap to FBPINN, often with significantly faster convergence.

This indicates that in higher-dimensional multi-scale problems, *feature expressivity dominates randomisation*. Pre-training restores much of the flexibility lost by freezing hidden weights, while retaining the computational efficiency of a linear least-squares solve.

Matrix Conditioning and Rank Structure

The conditioning improvements due to pre-training are subtle but consistent. While the condition number of $\mathbf{M}$ remains large in absolute terms, pre-training frequently increases matrix rank, with many configurations approaching full or near-full rank under PT.

The improvement in rank is particularly pronounced in regimes with oscillatory local solutions. This is a direct consequence of supervised shaping of hidden-layer functions. We observe that pre-training results in nonlinear feature diversity in the hidden-layer basis functions, which results in reduced redundancy between functions. When subdomains become extremely small and local solutions approach linearity, PT basis functions revert toward quasi-linear forms, and rank improvements diminish. This reinforces the interpretation that pre-training enhances expressivity precisely where

nonlinear structure exists.

Scalability and Transfer Across Frequencies

The transfer-learning studies further clarify the role of pre-training. Pre-training does not significantly improve performance on easier, lower-frequency tasks. However, it becomes increasingly beneficial as solution complexity increases. In higher-frequency regimes, PT-ELM-FBPINN achieves the lowest final $\hat{e}_{L^1}$, the most consistent performance, and the smallest performance variance across runs. This demonstrates that pre-training produces a feature space that scales better with problem difficulty. It should be noted that our tests were performed by using the same exact solution between pre-trainer and ELM-FBPINN. In many real-world cases, there is not a guarantee of an exact solution that can be relied on. However, our tests are at least somewhat a promising indication that given a function with similar frequency components, one could pre-train using a dataset that differs from a chosen PDE solution. One shortcoming of our transfer learning study is that there is no guarantee of performance gains if the training set does not represent the PDE solution well. One could argue that this would result in essentially a random set of basis functions and performance should be comparable to an ELM-FBPINN with no pre-training, however, this is dependent on the training set being at least somewhat within reasonable similarity to the PDE. Should the training set be significantly different in behaviours, pre-training could result in a basis unfit for a given PDE solution and performance would likely deteriorate.

Overarching Takeaways

The results of this chapter suggest several conclusions. Random features are sufficient only when local solution is simple. Feature shaping is critical when local nonlinear structure is rich. PT accelerates optimisation, but also enhances stability and rank. In 2D multi-scale settings, expressive hidden-layer bases are essential.

Importantly, PT-ELM-FBPINN closes much of the gap between classical random-feature ELM methods and gradient-descent-based architectures such as FBPINN, while retaining the efficiency and scalability of a linear least-squares formulation.

It should be noted that PT-ELM-FBPINN has not introduced any changes to the linear system, but only to the randomised hidden-layer basis functions. As such, PT-ELM-FBPINN poses the same limitations as ELM-FBPINN when considering more difficult problems, such as higher dimensions, nonlinear PDEs, problems with irregular geometries, and problems with complex boundary conditions. The supervised learning stage has no specific additional complexity to consider, and can flexibly be applied to more complex problems. The main bottleneck for the model then is with the ELM-FBPINN stage which has been discussed in Chapter 3.

Final Perspective

Pre-training the hidden-layer weights on a representative sample of exact solutions fundamentally reshapes the feature space in which the least-squares optimisation is performed. Rather than relying on randomly initialised basis functions, the network is endowed with features that are already aligned with the underlying solution manifold. This alignment improves the numerical stability of the linear system, accelerates convergence of the LSQR solver, and enhances robustness when architectural parameters such as depth, width, or domain decomposition are varied. The benefits become particularly pronounced in complex multi-scale regimes, where richer nonlinear structure demands a more expressive and well-conditioned basis.

From a conceptual standpoint, PT-ELM-FBPINN occupies an intermediate position between classical random feature methods and fully trainable PINN architectures. By shaping the hidden-layer basis through supervised pre-training, it recovers much of the expressive flexibility typically associated with gradient-descent-based neural networks. These findings suggest that one can improve performance by carefully designing the feature space prior to solving the linear system.

Chapter 5

Conclusion

In this thesis, we have conducted a numerical study on the combination of ELMs with FBPINNs for solving multiscale PDEs. By comparing to a vanilla PINN and FBPINN, we have established the performance potential that ELM-FBPINN has by eliminating the need for costly gradient-descent based optimisation, replaced instead by a linear least-squares system. Two problems have been studied to assess the models presented – the 1D damped harmonic oscillator and a 2D multiscale Laplace. Both problems were chosen for the high frequency, multiscale features they contain. In Chapter 3, we presented ELM-FBPINN, which replaces the subdomain networks of FBPINN with ELMs, thus changing the optimisation problem from a gradient-descent based solve, to a linear least-squares solve. In Chapter 4, we extended ELM-FBPINN by introducing a pre-training stage to the hidden-layer basis functions, with the goal of performing feature space approximation, thereby allowing the ELM-FBPINN to better suit the problem domain.

Both ELM-FBPINN and PT-ELM-FBPINN were compared directly to PINN and FBPINN across a number of ablation studies in 1D and 2D. The experiments were chosen such that they isolated the effects of domain decomposition, network capacity, depth, and solution frequency. All models were assessed on convergence, stability, and computational efficiency. The central questions posed in Chapter 1 motivated our choice of ablation tests and our results have answered each fully. The questions were as follows:

1. How does linearising the optimisation of FBPINNs affect convergence speed, solution accuracy, and computational efficiency?

2. Can supervised pre-training of hidden-layer basis functions improve the performance of ELM-FBPINN models?

3. How does pre-training influence matrix conditioning, singular values, and rank properties of the resulting least-squares systems?

4. Under what regimes (frequency, dimensionality, depth, domain decomposition) does pre-training provide the greatest benefit?

Across all studies, the ELM models presented the fastest convergence, characterised by a sharp initial reduction in error before settling into a more gradual convergence. By increasing the training iterations of the PT stage, this initial convergence speed can be further increased in PT-ELM-FBPINN models. This is in stark contrast to the PINN models, which failed to converge on any of the tests, even with additional network capacity compared to FBPINN or ELM-FBPINN. FBPINN is still a robust method, converging consistently with sufficient model capacity, but often in roughly double the number of required iterations compared to the ELM models.

In 1D, FBPINN, ELM-FBPINN and PT-ELM-FBPINN all proved to be well suited to the problem. While ELM-FBPINN outperformed FBPINN in nearly all test cases, the one exception was for tests involving increased network depth. In these cases, there was an observed oversaturation of activation functions resulting in worse performance, with the FBPINN proving to be slightly more robust. PT-ELM-FBPINN, however, addressed some of the oversaturation effects of deep layer networks, and improved the precision and convergence behaviour compared to ELM-FBPINN. This performance gain was observed across all tests, except in the case of a large number of subdomains. In this regime, local solutions become significantly more linear, which suits the quasi-linear basis functions of ELM-FBPINN better than the more non-linear PT-ELM-FBPINN basis functions. In all other cases, PT-ELM-FBPINN proved to be the more stable, and accurate method, with the fastest convergence. The conditioning of the ELM matrices

sees little improvement with pre-training, but there is a significant improvement in the linear independence of the columns. The PT stage introduces non-linearity to the hidden-layer basis, allowing for more expressive power and unique basis functions, resulting in a more stable and accurate least-squares solve.

In 2D, FBPINN proved to be a well suited architecture for the spatially complex problem domain, outperforming ELM-FBPINN in precision consistently across tests. While this comes at a cost of requiring a higher iteration budget, the difference in the error is often an order of magnitude in improvement. PT-ELM-FBPINN either closes the gap or exceeds the performance of FBPINN, showing the potential for pre-training the hidden-layer basis as a means of enhancing the linear solve. Again, convergence speed is improved with pre-training, with PT-ELM-FBPINN consistently presenting the characteristic sharp initial reduction in $\hat{e}_{L^1}$.

One major aspect of pre-training is the potential for a transfer learning schedule. In both 1D and 2D, the pre-training of the hidden-layer weights on a lower frequency set of exact solutions, resulted in a notable improvement to precision and stability, with the PT-ELM-FBPINN results having the smallest overall difference in performance on our multiscale tests.

The conditioning of the matrix, $\mathbf{M}$, is relatively large for both ELM models, but has been shown to not affect the LSQR solvers' abilities to converge on accurate predictions. Increasing the number of subdomains or basis functions, directly increases the number of columns in the ELM matrices. For ELM-FBPINN, this results in a progressively more rank deficient matrix, in part due to the quasi-linear basis functions. PT-ELM-FBPINN overcomes this progressive rank deficiency through the addition of non-linearity to the basis, but only in regimes not related to refinement of the subdomain widths. In these large J tests, the ELMs' performance converges, with ELM-FBPINN presenting the less rank deficient matrices at very large J.

Future work The next step for PT-ELM-FBPINN is to apply the method to higher dimensional, time-dependent problems, as well as non-linear problems. As mentioned in Chapter 4, PT-ELM-FBPINN can already be applied to higher dimensional problems,

including space-time problems, inverse problems, and parametric PDEs. For application to non-linear PDEs, some additional alterations to the solver of ELM-FBPINN is required, primarily through the addition of some iterative solver such as a Newton's method. Assuming this addition, PT-ELM-FBPINN can easily be applied to non-linear PDEs as well. Given that this is already possible, applying PT-ELM-FBPINN to more difficult problems is a natural next step.

In [67], pre-conditioning ELM-FBPINN has been shown to reduce the condition number, which in turn improves the convergence speed and precision. Additionally, they introduced a filtering procedure to reduce redundancies in the columns, which decreases the overall size of $\mathbf{M}$, thus improving computational efficiency. Combining this pre-conditioning with PT-ELM-FBPINN would be a clear next step, though the filtering schedule is likely to only aid the architecture in small subdomain size regimes.

There is also a potential for introducing some regularisation terms to the loss function that more actively train the hidden-layer basis to a better conditioned system. This would be as simple as adding regularisers to the PT loss function and adjusting the scaling of terms to ensure a good balance. The hope is that by adding these terms, the resulting ELM matrix weights will not only be well suited to the problem domain, but also result in a more stable system overall, further improving convergence speed and precision.

Bibliography

[1] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, K. Willcox, and S. Lee, "Workshop Report on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence," tech. rep., USDOE Office of Science (SC) (United States), Feb. 2019.

[2] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations," Nov. 2017.

[3] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations," Nov. 2017.

[4] 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, Feb. 2019.

[5] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, pp. 503–528, Aug. 1989.

[6] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017.

[7] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Inferring solutions of differential equations using noisy multi-fidelity data," *Journal of Computational Physics*, vol. 335, pp. 736–746, Apr. 2017.

[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*, OSDI'16, (USA), pp. 265–283, USENIX Association, Nov. 2016.

[9] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: an imperative style, high-performance deep learning library," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, no. 721, pp. 8026–8037, Red Hook, NY, USA: Curran Associates Inc., Dec. 2019.

[10] 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.

[11] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A Deep Learning Library for Solving Differential Equations," *SIAM Review*, vol. 63, pp. 208–228, Jan. 2021.

[12] PhysicsNeMo Contributors, "NVIDIA PhysicsNeMo: An open-source framework for physics-based deep learning in science and engineering," Feb. 2023.

[13] E. Haghighat and R. Juanes, "SciANN: A Keras/Tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 373, p. 113552, Jan. 2021.

[14] A. Yazdani, L. Lu, M. Raissi, and G. E. Karniadakis, "Systems biology informed deep learning for inferring parameters and hidden dynamics," *PLOS Computational Biology*, vol. 16, p. e1007575, Nov. 2020.

[15] F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, and E. Kuhl, "Physics-Informed Neural Networks for Cardiac Activation Mapping," *Frontiers in Physics*, vol. 8, Feb. 2020.

[16] M. A. Calicchia, R. Mittal, J.-H. Seo, and R. Ni, "Reconstructing the pressure field around swimming fish using a physics-informed neural network," *Journal of Experimental Biology*, vol. 226, p. jeb244983, Apr. 2023.

[17] J. O. Dabiri, S. Bose, B. J. Gemmell, S. P. Colin, and J. H. Costello, "An algorithm to estimate unsteady and quasi-steady pressure fields from velocity field measurements," *The Journal of Experimental Biology*, vol. 217, pp. 331–336, Feb. 2014.

[18] M. Rasht-Behesht, C. Huber, K. Shukla, and G. E. Karniadakis, "Physics-Informed Neural Networks (PINNs) for Wave Propagation and Full Waveform Inversions," *Journal of Geophysical Research: Solid Earth*, vol. 127, no. 5, p. e2021JB023120, 2022.

[19] C. Song, T. Alkhalifah, and U. B. Waheed, "Solving the frequency-domain acoustic VTI wave equation using physics-informed neural networks," *Geophysical Journal International*, vol. 225, pp. 846–859, Apr. 2021.

[20] C. Shen, A. P. Appling, P. Gentine, T. Bandai, H. Gupta, A. Tartakovsky, M. Baity-Jesi, F. Fenicia, D. Kifer, L. Li, X. Liu, W. Ren, Y. Zheng, C. J. Harman, M. Clark, M. Farthing, D. Feng, P. Kumar, D. Aboelyazeed, F. Rahmani, Y. Song, H. E. Beck, T. Bindas, D. Dwivedi, K. Fang, M. Höge, C. Rackauckas, B. Mohanty, T. Roy, C. Xu, and K. Lawson, "Differentiable modelling to unify machine learning and physical models for geosciences," *Nature Reviews Earth & Environment*, vol. 4, pp. 552–567, Aug. 2023.

[21] W. Ji, J. Chang, H.-X. Xu, J. R. Gao, S. Gröblacher, H. P. Urbach, and A. J. L. Adam, "Recent advances in metasurface design and quantum optics applications with machine learning, physics-informed neural networks, and topology optimization methods," *Light: Science & Applications*, vol. 12, p. 169, July 2023.

[22] Y. Chen, L. Lu, G. E. Karniadakis, and L. D. Negro, "Physics-informed neural networks for inverse problems in nano-optics and metamaterials," *Optics Express*, vol. 28, pp. 11618–11633, Apr. 2020.

[23] S. Amini Niaki, E. Haghighat, T. Campbell, A. Poursartip, and R. Vaziri, "Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture," *Computer Methods in Applied Mechanics and Engineering*, vol. 384, p. 113959, Oct. 2021.

[24] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, "Physics-Informed Neural Networks for Heat Transfer Problems," *Journal of Heat Transfer*, vol. 143, Apr. 2021.

[25] R. Rossi, M. Gelfusa, A. Murari, and o. b. o. J. contributors, "On the potential of physics-informed neural networks to solve inverse problems in tokamaks," *Nuclear Fusion*, vol. 63, p. 126059, Nov. 2023.

[26] Y. Tang, "High-Fidelity prediction of radioisotope production Cross-Sections using Bayesian neural networks and Auto Machine learning," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 569, p. 165887, Dec. 2025.

[27] Elsevier, "Scopus," 2026.

[28] 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, p. 62, July 2023.

[29] 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, pp. A3055–A3081, Jan. 2021.

[30] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, Mar. 2010.

[31] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the Spectral Bias of Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning*, pp. 5301–5310, PMLR, May 2019.

[32] T. Luo, Z. Ma, Z.-Q. J. Xu, and Y. Zhang, "On the Exact Computation of Linear Frequency Principle Dynamics and Its Generalization," *SIAM Journal on Mathematics of Data Science*, Nov. 2022.

[33] T. Luo, Z. Ma, Z.-Q. J. Xu, and Y. Zhang, "Theory of the Frequency Principle for General Deep Neural Networks," *CSIAM Transactions on Applied Mathematics*, vol. 2, no. 3, pp. 484–507, 2021.

[34] Z.-Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma, "Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks," *Communications in Computational Physics*, vol. 28, pp. 1746–1767, June 2020.

[35] Z.-Q. J. Xu, L. Zhang, and W. Cai, "On understanding and overcoming spectral biases of deep neural network learning methods for solving PDEs," *Journal of Computational Physics*, vol. 530, p. 113905, June 2025.

[36] A. Jacot, F. Gabriel, and C. Hongler, "Neural Tangent Kernel: Convergence and Generalization in Neural Networks," Feb. 2020.

[37] 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, Jan. 2022.

[38] S. Wang, H. Wang, and P. Perdikaris, "On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 384, p. 113938, Oct. 2021.

[39] A. Guzmán-Cordero, F. Dangel, G. Goldshlager, and M. Zeinhofer, "Improving Energy Natural Gradient Descent through Woodbury, Momentum, and Randomization," Oct. 2025. arXiv:2505.12149 [cs.LG].

[40] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, Jan. 1989.

[41] H. Gao, L. Sun, and J.-X. Wang, "PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain," *Journal of Computational Physics*, vol. 428, p. 110079, Mar. 2021.

[42] Y. Jiang, Z. Liu, P. Kabirzadeh, Y. Wu, Y. Li, N. Miljkovic, and P. Wang, "Multi-fidelity physics-informed convolutional neural network for heat map prediction of battery packs," *Reliability Engineering & System Safety*, vol. 256, p. 110752, Apr. 2025.

[43] R. Hassannejad, M. M. Ettefagh, and Y. B. Mossayebi, "Adaptive Wavelet-Based Physics-Informed CNN for Bearing Fault Diagnosis," *International Journal of Prognostics and Health Management*, vol. 16, Apr. 2025.

[44] N. Wandel, M. Weinmann, M. Neidlin, and R. Klein, "Spline-PINN: Approaching PDEs without Data Using Fast, Physics-Informed Hermite-Spline CNNs," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 8529–8538, June 2022.

[45] M. Sammartino, L. Della Cioppa, S. Colella, and B. Buongiorno Nardelli, "A Physics-informed deep neural network for the joint prediction of 3D chlorophyll-a and hydrographic fields in the Mediterranean Sea," *Environmental Modelling & Software*, vol. 194, p. 106660, Oct. 2025.

[46] P. Ren, C. Rao, Y. Liu, J.-X. Wang, and H. Sun, "PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs," *Computer Methods in Applied Mechanics and Engineering*, vol. 389, p. 114399, Feb. 2022.

[47] L. Yang, D. Zhang, and G. E. Karniadakis, "Physics-Informed Generative Adversarial Networks for Stochastic Differential Equations," *SIAM Journal on Scientific Computing*, Feb. 2020.

[48] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, (Red Hook, NY, USA), pp. 7537–7547, Curran Associates Inc., Dec. 2020.

[49] Z. Uddin, S. Ganga, R. Asthana, and W. Ibrahim, "Wavelets based physics informed neural networks to solve non-linear differential equations," *Scientific Reports*, vol. 13, p. 2882, Feb. 2023.

[50] D. Broomhead and D. Lowe, "Multivariable Functional Interpolation and Adaptive Networks," *Complex Syst.*, 1988.

[51] 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, Oct. 2023.

[52] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, "Variational Physics-Informed Neural Networks For Solving Partial Differential Equations," Nov. 2019.

[53] E. Kharazmi, Z. Zhang, and G. E. M. Karniadakis, "*hp*-VPINNs: Variational physics-informed neural networks with domain decomposition," *Computer Methods in Applied Mechanics and Engineering*, vol. 374, p. 113547, Feb. 2021.

[54] 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, pp. 2002–2041, 2020.

[55] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 365, p. 113028, June 2020.

[56] A. Alhubail, X. He, M. AlSinan, H. Kwak, and H. Hoteit, "Extended Physics-Informed Neural Networks for Solving Fluid Flow Problems in Highly Heterogeneous Media," OnePetro, Feb. 2022.

[57] C. Dong, J. Tao, H. Sun, Q. Wei, H. Tan, and C. Liu, "Innovative fault diagnosis for axial piston pumps: A physics-informed neural network framework predicting pump flow ripple," *Mechanical Systems and Signal Processing*, vol. 225, p. 112274, Feb. 2025.

[58] 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.

[59] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, Dec. 2006.

[60] D. R. Nayak, D. Das, R. Dash, S. Majhi, and B. Majhi, "Deep extreme learning machine with leaky rectified linear unit for multiclass classification of pathological brain images," *Multimedia Tools and Applications*, vol. 79, pp. 15381–15396, June 2020.

[61] D. Wu, Z. Qu, F. Guo, and Q. Wan, "Multilayer Incremental Hybrid Cost-Sensitive Extreme Learning Machine With Multiple Hidden Output Matrix and Subnetwork Hidden Nodes," *IEEE Access*, vol. 7, pp. 118422–118434, 2019.

[62] V. Dwivedi and B. Srinivasan, "Physics Informed Extreme Learning Machine (PIELM)–A rapid method for the numerical solution of partial differential equations," *Neurocomputing*, vol. 391, pp. 96–118, May 2020.

[63] J. Chen, X. Chi, W. E, and Z. Yang, "Bridging Traditional and Machine Learning-Based Algorithms for Solving PDEs: The Random Feature Method," *Journal of Machine Learning*, vol. 1, no. 3, pp. 268–298, 2022.

[64] 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, I. 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, and P. van Mulbregt, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, Mar. 2020.

[65] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient BackProp," in *Neural Networks: Tricks of the trade* (G. Orr and M. K, eds.), Springer, 1998.

[66] G. Meurant, *The Lanczos and conjugate gradient algorithms: from theory to finite precision computations.* SIAM, 2006.

[67] J. W. van Beek, V. Dolean, and B. Moseley, "Local feature filtering for scalable and well-conditioned domain-decomposed random feature methods," *Computer Methods in Applied Mechanics and Engineering*, vol. 449, Feb. 2026.