

Galerkin neural networks for the approximation of partial differential equations with error control

By

Justin Dong

B.A., B.S., Rice University, 2014

Sc.M., Brown University, 2019

Submitted in partial fulfillment of the requirements for the Degree of
Doctor of Philosophy in the Division of Applied Mathematics at Brown
University

Providence, Rhode Island

May 2023

© Copyright 2023 Justin Dong

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

Date _____

Mark Ainsworth, Ph.D., Advisor

Recommended to the Graduate Council

Date _____

Brendan Keith, Ph.D., Reader

Date _____

Panagiotis Stinis, Ph.D., Reader

Approved by the Graduate Council

Date _____

Thomas A. Lewis, Dean of the Graduate School

Curriculum Vitae

Education

2019–2023 Ph.D., Applied Mathematics

Brown University (Providence, RI, USA)

2017–2019

Sc.M., Applied Mathematics

Brown University (Providence, RI, USA)

2010–2014

B.A., Computational and Applied Mathematics

B.S., Mechanical Engineering

Rice University (Houston, TX, USA)

Publications

Papers

Ainsworth, M., **Dong, J.** (in preparation). *On the neural network approximation of problems with corner singularities.*

Howard, A., **Dong, J.**, Patel, R., D’Elia, M., Maxey, M., Stinis, P. (submitted 2023). *Machine learning methods for particle stress development in suspension Poiseuille flows.* Rheologica Acta.

Ainsworth, M., **Dong, J.** (2022). *Galerkin neural network approximation of singularly-perturbed elliptic systems.* Computer Methods in Applied Mechanics and Engineering.

Ainsworth, M., **Dong, J.** (2021). *Galerkin neural networks: A framework for approximating variational equations with error control.* SIAM Journal on Scientific Computing.

Dong, J., Rivière, B (2016). *A semi-implicit method for incompressible three-phase flow in porous media.* Computational Geosciences.

Teaching

Summer 2021

Instructor, *Brown University Pre-College*,

Course: CECS 0915 - Artificial Intelligence: Modeling Human Intelligence with Networks

Summer 2020

Teaching Assistant, *Summer@ICERM*,

Fast Learning Algorithms for Numerical Computation and Data Analysis

Spring 2019

Graduate teaching assistant, *Brown University Division of Applied Mathematics*,

Course: APMA 340 - Methods of Applied Mathematics II

Fall 2018

Graduate teaching assistant, *Brown University Division of Applied Mathematics*,

Course: APMA 330 - Methods of Applied Mathematics I

Professional Development

Invited Lectures

Sept. 2022, Talk: Galerkin neural network approximation of multiscale problems.

SIAM Mathematics of Data Science

July 2021, Talk: Galerkin neural networks: a framework for approximating variational equations with error control

SIAM Annual Meeting

Awards and Fellowships

2018-2022: National Science Foundation Graduate Research Fellowship

Outreach and Service

Applied Math Directed Reading Program, Founder & Organizer @ Brown University.
Applied Math Graduate/Undergraduate Mentoring Program, Organizer @ Brown University.

Faculty-Graduate Liason, Applied Math @ Brown University.

Acknowledgments

Thank you first to my advisor, Professor Mark Ainsworth, for his invaluable support, insights, and guidance over the last several years. In the many times I felt lost and unsure throughout this long journey, he helped keep me steady and focused. I would also like to thank Dr. Panos Stinis and Professor Brendan Keith for serving on my thesis committee, the former whose guidance and mentoring during my time at PNNL I am also grateful for.

Beyond my committee members, I would like to thank: Dr. Amanda Howard and Dr. Arturo Vargas for their mentorship, collaborations, and assistance with navigating academia; Professor Yeonjing Shin and Dr. Marshall Jiang for the many fruitful technical discussions over the years; Professor Béatrice Rivière for her mentorship and inspiring my interest in mathematics research; my office mates Emily Winn, Tingwei Meng, and Yin-Ting Liao for the camaraderie and entertainment; and all of the front office staff, especially Candida Hall and Rosanna Wertheimer, whose tireless dedication to the Division of Applied Mathematics and its graduate students is appreciated more than you know.

To my friends in Providence – Rebecca, Juniper, Sophie, Lucy, Kevin, Adam, Darren, and Heather – and my trail friends and beyond – Holly, Tobin, Shanelle, Ngan, Paulina (and Sammy and Dodo), Marissa, and Jenna – thank you for always providing a much-needed respite from my studies.

Lastly, I am deeply grateful to my parents for their love and support and for making this journey possible; my aunts and grandparents for practically raising me alongside my

parents; and my brother for always lending an ear to my venting.

This thesis is based upon work funded by National Science Foundation Graduate Research Fellowship under Grant No. 1644760 as well as PhILMS grant DE-SC0019453.

Contents

Curriculum Vitae	iv
Acknowledgments	vi
1 Introduction	1
1.1 Outline	3
2 Preliminaries	6
2.1 Sobolev Spaces	6
2.2 Neural Networks	8
2.2.1 Approximation Theory	10
2.2.2 Approximation of PDEs Based on the Strong Formulation	11
2.2.3 Approximation of PDEs Based on the Weak Formulation	13
2.2.4 Other Approaches	14
3 The Galerkin neural network framework	16
3.1 Introduction	16
3.2 Basic Framework & Algorithm	18
3.2.1 Galerkin Approximation of Variational Equations	20
3.2.2 Construction of the Galerkin Subspace	22
3.2.3 Galerkin Subspace Augmentation	28
3.2.4 Neural Network Approximation of Augmented Basis Function	33

3.3	Numerical Examples	37
3.3.1	Function Fitting	37
3.3.2	Second Order Problems	40
3.3.3	Fourth Order Problems	50
3.4	Conclusions	55
3.5	Acknowledgments	56
4	Application to Reissner-Mindlin plates	58
4.1	Introduction	58
4.2	Review of the Galerkin Neural Network Framework	61
4.3	Boundary Layer Resolution using Neural Networks	64
4.3.1	Benchmark Solution of Reissner-Mindlin Plate Exhibiting Boundary Layers	65
4.3.2	Function Fitting of Reissner-Mindlin Shear Stress	68
4.4	Variational Formulations of the Reissner-Mindlin Model	69
4.4.1	Natural Variational Formulation	70
4.4.2	Mixed Least Squares Variational Formulation	74
4.4.3	Least Squares Based on Brezzi-Fortin Formulation	82
4.5	Conclusions	92
4.6	Acknowledgement	95
4.7	Appendix	96
5	Application to Suspension Poiseuille Flows	100
5.1	Introduction	100
5.2	The Force Coupling Method	103
5.3	The Suspension Balance Model	104
5.3.1	Monodisperse Suspensions	104
5.3.2	Bidisperse Suspensions	105
5.4	Galerkin Neural Network Formulation of the Problem	107

5.4.1	The Monodisperse Case	107
5.4.2	The Bidisperse Case	110
5.5	Numerical Examples	113
5.5.1	Monodisperse Suspensions	113
5.5.2	Bidisperse Suspensions	117
5.6	Conclusions	118
5.7	Acknowledgments	119
6	Extension to non-self-adjoint and indefinite problems	120
6.1	Introduction	120
6.2	Galerkin neural networks	123
6.2.1	The basic framework	123
6.2.2	Extension to non-self adjoint positive-definite problems	126
6.2.4	Knowledge-based subspaces	131
6.3	Applications	132
6.3.1	Poisson equation	133
6.3.4	Stokes flow	139
6.4	Conclusions	145
6.5	Acknowledgments	146
	Appendices	147
6.A	Knowledge-based functions for Stokes flow	147
6.B	Training procedure for Galerkin neural networks with knowledge-based functions	150
7	Conclusions	152
7.1	Summary of Contributions	152
7.2	Future Directions	153
7.2.1	Nonlinear Problems	153

7.2.2	Dense Least Squares	153
7.2.3	A Posteriori Error Estimation	154
8	Appendices	156
	Appendices	157
8.A	Network Architecture	157
8.A.1	Effect of network depth	157
8.A.4	Effect of Activation Function	160
8.B	Comparison with other neural network approaches	162

List of Figures

2.1	An example of a feedforward neural network with four hidden layers. . . .	9
3.1	A visualization of the Galerkin Neural Network framework using three shallow networks consisting of two, four, and six neurons each.	29
3.2	(a) Estimated and true L^2 error at each Galerkin iteration. (b) The piecewise constant segments (solid lines) denote the true errors for each Galerkin iteration i , while the dashed red line (resp. blue line) denotes the progress of the loss function in approximating the true errors within each Galerkin iteration for increasing network widths (resp. constant network widths). The x -axis thus denotes the cumulative training epoch over all Galerkin iterations.	37
3.3	(a) True error $u - u_{i-1}$ and approximate error φ_j^{NN} for $i = 2$. (b) True error $u - u_{i-1}$ and approximate error φ_i^{NN} for $i = 4$ with fixed width $n_i = 100$	38
3.4	(a) Error at each Galerkin iteration i using a Gauss-Legendre quadrature rule with varying number of nodes (b) the analogous results using a left Riemann-sum with varying number of nodes.	38

3.5	Displacement of a string (Poisson equation in 1D). (a) Estimated and true error in the L^2 and energy norms at each Galerkin iteration, $n_i = 400$ for all i . (b) The piecewise constant segments (solid lines) denote the true errors for each Galerkin iteration i , while the dashed red line (resp. blue line) denotes the progress of the loss function (resp. L^2 error estimator $\ \varphi_i^{NN}\ _{L^2}$) in approximating the true errors within the each Galerkin iteration with $n_i = 400$. The x -axis thus denotes the cumulative training epoch over all Galerkin iterations. (c) and (d) The analogous results for $n_i = 5 \times 2^{i-1}$	39
3.6	Displacement of a string (Poisson equation in 1D). True error $u - u_{i-1}$ and estimated error φ_i^{NN} for $i = 2, 4, 6, 8$	41
3.7	Displacement of a string. The total norm of the network parameters for φ_i^{NN} and $\varphi_i^{NN}/\ \varphi_i^{NN}\ $	42
3.8	Displacement of a membrane (Poisson equation in 2D). Estimated and exact error in the L^2 and energy norms at each Galerkin iteration (a) and epoch (b). Initial hyperplanes (c).	43
3.9	Displacement of a membrane (Poisson equation in 2D). Exact error $u - u_{i-1}$ (right column) and approximate error φ_i^{NN} (left column) for $i = 2, 3, 4$	45
3.10	Displacement of a membrane. The total norm of the network parameters for φ_i^{NN} and $\varphi_i^{NN}/\ \varphi_i^{NN}\ $	46
3.11	Poisson equation with line source data, $R_e = 1 - 1/\pi^2$ and $R_0 = 1/\sqrt{29}$. (a) true and estimated errors after each Galerkin iteration (b) true and estimated errors after each training epoch (c) approximate solution u_i for $i = 7$	46
3.12	Poisson equation with line source data, $R_e = 1 - 1/\pi^2$ and $R_0 = 7/(6\sqrt{2})$. (a) true and estimated errors after each Galerkin iteration (b) true and estimated errors after each training epoch (c) approximate solution u_i for $i = 7$	48

3.13	Poisson equation in L-shaped domain. Estimated errors after each Galerkin iteration (a) and epoch (b) as well as approximate solution u_i for $i = 8$ (c).	49
3.14	The biharmonic equation in 1D (displacement of a beam). True and estimated error in the L^2 and energy norm at the end of each Galerkin iteration (a) and epoch (b).	51
3.15	Displacement of a beam. Exact error $u - u_{i-1}$ and approximate error φ_i^{NN} for $i = 2, 4, 6, 8$.	52
3.16	The biharmonic equation in 1D with point moment (beam with applied couple). True and estimated error in the L^2 and energy norm at the end of each Galerkin iteration (a) and epoch (b).	53
3.17	Biharmonic equation with point moment in 1D (beam with applied couple). Exact error $u - u_{i-1}$ and approximate error φ_i^{NN} as well as numerical solution u_i and its derivative u_i'' for $i = 2$ (top row), $i = 5$ (middle row), and $i = 7$ (bottom row).	54
3.18	The biharmonic equation in 2D. True and estimated error in the L^2 and energy norm at the end of each Galerkin iteration (a) and epoch (b).	55
3.19	Biharmonic equation in 2D with point load source. True error $u - u_{i-1}$ (middle column) and estimated error φ_i^{NN} (left column) as well as numerical solution u_i (right column) for $i = 2, 3, 5$.	56
4.1	Left: Numerical approximation rate of the function $u(x) = (1 - e^{(x-1)/t} - e^{-x/t} + e^{-1/t})/(1 + e^{-1/t})$ for various t using a ReLU network with one hidden layer. Right: The function $u(x)$ alongside the ReLU approximation. The hidden weights are set to 1, the biases are graded to account for the boundary layers, and the activation coefficients are chosen so that the network interpolates the function u at the points where the ReLU basis functions activate.	60
4.2	Domain for the test problem in Section 4.3.	65

4.3	Left: True x -component of shear stress with $n = 1$ and $t = 10^{-2}$. Right: True y -component of the shear stress with $n = 1$ and $t = 10^{-2}$	66
4.4	Left: True x -component of shear stress with $n = 1$ and $t = 10^{-2}$. Right: Neural network approximation of x -component of shear stress with $n = 1$ and $t = 10^{-2}$	69
4.5	True solution ω (left), β (middle), and σ (right) with $t = 10^{-6}$ and $n = 0$	71
4.6	Relative error after each iteration of Algorithm 1 for the case when $n = 0$ with variational formulation on $\mathfrak{B}_0, \mathfrak{L}_0$	72
4.7	Approximate x -component of the shear stress for the 2nd, 4th, and 6th iterations of the Galerkin Neural Network algorithm for the univariate problem with variational formulation on $\mathfrak{B}_0, \mathfrak{L}_0$	72
4.8	Relative errors after each iteration of Algorithm 1 for the univariate problem with variational formulation on $\mathfrak{B}_{LS}, \mathfrak{L}_{LS}$	75
4.9	For $t = 10^{-6}$, the y -component of the true error $\sigma - \sigma_i$ (top row) and the y -component of the basis function τ_i^{NN} (bottom row) for the 1st, 3rd, and 5th iterations of Algorithm 1 for the univariate problem with variational formulation on $\mathfrak{B}_{LS}, \mathfrak{L}_{LS}$	76
4.10	For $t = 10^{-6}$, the y -component of the true error $\beta - \beta_i$ (top row) and the y -component of the basis function φ_i^{NN} (bottom row) for the 1st, 3rd, and 5th iterations of Algorithm 1 for the univariate problem with variational formulation on $\mathfrak{B}_{LS}, \mathfrak{L}_{LS}$	79
4.11	For $t = 10^{-6}$, the true error $\omega - \omega_i$ (top row) and the basis function v_i^{NN} (bottom row) for the 1st, 3rd, and 5th iterations of Algorithm 1 for the univariate problem with variational formulation on $\mathfrak{B}_{LS}, \mathfrak{L}_{LS}$	80
4.12	Left: True x -component of shear stress. Right: Neural network approximation of x -component of shear stress with variational formulation on $\mathfrak{B}_{LS}, \mathfrak{L}_{LS}$	80

4.13	Relative energy error, relative L^2 error in β and ω , and H^1 error in σ after each iteration of Algorithm 1 for the Reissner-Mindlin plate with boundary layer and variational formulation on $\mathfrak{B}_{LS}, \mathfrak{L}_{LS}$	81
4.14	Errors after each iteration of Algorithm 1 for the univariate problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}, \tilde{\mathfrak{L}}_{BF}$	82
4.15	For $t = 10^{-6}$ in the univariate problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}, \tilde{\mathfrak{L}}_{BF}$: (a)-(c) True error $\beta^{(2)} - \beta_i^{(2)}$. (d)-(f) Basis function $\varphi_i^{(2),NN}$	83
4.16	For $t = 10^{-6}$ in the univariate problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}, \tilde{\mathfrak{L}}_{BF}$: basis function μ_i^{NN}	84
4.17	For $t = 10^{-6}$ in the univariate problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}, \tilde{\mathfrak{L}}_{BF}$: basis function $q_i^{(2),NN}$	84
4.18	For $t = 10^{-2}$ in the univariate problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}, \tilde{\mathfrak{L}}_{BF}$: approximation $\sigma_i^{(2)} = -\partial p_i / \partial x - \partial r_i / \partial y$	85
4.19	Energy error and L^2 error in β and ω for the problem with sinusoidal load with variational formulation on $\tilde{\mathfrak{B}}_{BF}, \tilde{\mathfrak{L}}_{BF}$	85
4.20	For $t = 10^{-2}$ in the problem with sinusoidal load with variational formulation on $\tilde{\mathfrak{B}}_{BF}, \tilde{\mathfrak{L}}_{BF}$: (a)-(c) True error $\beta^{(1)} - \beta_i^{(1)}$. (d)-(f) Basis function $\varphi_i^{(1),NN}$	87
4.21	For $t = 10^{-2}$ in the problem with sinusoidal load with variational formulation on $\tilde{\mathfrak{B}}_{BF}, \tilde{\mathfrak{L}}_{BF}$: (a)-(c) True error $\beta^{(2)} - \beta_i^{(2)}$. (d)-(f) Basis function $\varphi_i^{(2),NN}$	88
4.22	For $t = 10^{-2}$ in the problem with sinusoidal load with variational formulation on $\tilde{\mathfrak{B}}_{BF}, \tilde{\mathfrak{L}}_{BF}$: basis function μ_i^{NN}	89
4.23	For $t = 10^{-2}$ in the problem with sinusoidal load with variational formulation on $\tilde{\mathfrak{B}}_{BF}, \tilde{\mathfrak{L}}_{BF}$: basis function q_i^{NN}	90
4.24	For $t = 10^{-2}$ in the problem with sinusoidal load with variational formulation on $\tilde{\mathfrak{B}}_{BF}, \tilde{\mathfrak{L}}_{BF}$: (a)-(c) Approximation $\sigma_i^{(1)} = \partial p_i / \partial y - \partial r_i / \partial x$. (d)-(f) Approximation $\sigma_i^{(2)} = -\partial p_i / \partial x - \partial r_i / \partial y$	90

4.25	The triangular wave function g_T	91
4.26	Energy error and L^2 error in β , ω , and σ for the problem with triangular wave load with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$	91
4.27	For $t = 10^{-6}$ in the triangular wave forcing term problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: (a)-(c) True error $\beta^{(1)} - \beta_i^{(1)}$. (d)-(f) Basis function $\varphi_i^{(1),NN}$	92
4.28	For $t = 10^{-6}$ in the triangular wave forcing term problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: (a)-(c) True error $\beta^{(2)} - \beta_i^{(2)}$. (d)-(f) Basis function $\varphi_i^{(2),NN}$	93
4.29	For $t = 10^{-2}$ in the triangular wave forcing term problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: basis function μ_i^{NN}	93
4.30	For $t = 10^{-2}$ in the triangular wave forcing term problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: basis function q_i^{NN}	94
4.31	For $t = 10^{-2}$ in the triangular wave forcing term problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: (a)-(c) Approximation $\sigma_i^{(1)} = \partial p_i / \partial y - \partial r_i / \partial x$. (d)-(f) Approximation $\sigma_i^{(2)} = -\partial p_i / \partial x - \partial r_i / \partial y$	94
4.32	Coefficients $A_n(t)$, $B_n(t)$, $C_n(t)$, $D_n(t)$ and their series expansions for $n = 1, 10, 20, 30$	98
4.33	Errors in the true solution of the model problem in Section 4.4.3.3 with $t = 10^{-2}$ when using exact coefficients compared to their series expansions.	98
5.1	Top row: from left to right, the Galerkin Neural Network approximation to ϕ using 1, 3, 5, and 7 basis functions. Middle row: left to right, one-dimensional volume fraction profiles at $\gamma = 20$ using 1, 3, 5, and 7 basis functions. Bottom row: left to right, one-dimensional volume fraction profiles at $\gamma = 121$ using 1, 3, 5, and 7 basis functions.	113

5.2	Top row: left to right, one-dimensional flux profiles at $\gamma = 20, 59, 121, 198$ using 7 basis functions. Bottom row: left to right, one-dimensional stress profiles at $\gamma = 20, 59, 121, 198$ using 7 basis functions.	114
5.3	Top row: left to right, one-dimensional flux profiles at $\gamma = 20, 59, 121, 198$ using 7 basis functions. Bottom row: left to right, one-dimensional stress profiles at $\gamma = 20, 59, 121, 198$ using 7 basis functions.	115
5.4	Top two rows: left to right, one-dimensional flux and stress profiles for phase a (large particles) at $\gamma = 20, 59, 121, 198$ using 6 basis functions. Bottom two rows: left to right, one-dimensional flux and stress profiles for phase b (small particles) at $\gamma = 20, 59, 121, 198$ using 6 basis functions. The Shauly hindrance function is used for both phases.	116
5.5	Top two rows: left to right, one-dimensional flux and stress profiles for phase a (large particles) at $\gamma = 20, 59, 121, 198$ using 6 basis functions. Bottom two rows: left to right, one-dimensional flux and stress profiles for phase b (small particles) at $\gamma = 20, 59, 121, 198$ using 6 basis functions. The Davis-Gecol hindrance function is used for both phases.	118
6.1	Errors with respect to the number of basis functions for $\delta = 1$, $\mathcal{W} = L^2(\partial\Omega)$, and the sequence of problems $n = 2, 4, 8, 16$	129
6.2	Errors with respect to the number of basis functions for several choices of penalty parameter $\delta = n^\alpha$	130
6.3	The true solution $u(r, \theta) = r^\lambda \sin \theta$ (left) and approximate solutions obtain using Galerkin neural networks and (6.27) with $\beta = 0$ (middle) and $\beta = 1$ (right).	133
6.4	Example 6.3.2: for $\beta = 0$, the errors $u - u_{i-1}$ and learned basis functions φ_i^{NN}	134
6.5	Example 6.3.2: for $\beta = 1$, the errors $u - u_{i-1}$ and learned basis functions φ_i^{NN}	135

6.6	Example 6.3.2: for $\beta = 1$, the errors in the L^2 and energy norms after each basis function is added to the learned subspace.	136
6.7	Example 6.3.3: for $\beta = 4/3$, the approximate solutions u_i and learned basis functions φ_i^{NN}	136
6.8	Example 6.3.3: for $\beta = 4/3$, the approximate solutions u_i and learned basis functions φ_i^{NN}	138
6.9	Example 6.3.3: training of λ_1 and over several runs with initialization $\lambda \sim \mathcal{U}(0, 1)$	139
6.10	Channel with triangular cavity for Example 6.3.5.	141
6.11	Example 6.3.5: velocity and pressure with $\beta = 0$ (top) and $\beta = 5/3$ (bottom) in (6.30).	143
6.12	Example 6.3.5: velocity and pressure with $\beta = 5/3$ in (6.30) and basis functions learned from enriched neural network with knowledge-based functions.	143
6.13	Example 6.3.5: velocity and pressure basis functions with $\beta = 5/3$ in (6.30) and basis functions learned from enriched neural network with knowledge-based functions.	144
6.14	Example 6.3.5: velocity and pressure basis functions with $\beta = 4$ in (6.30) and basis functions learned from enriched neural network with knowledge-based functions.	145
8.A.1	Convergence of the error for plain feedforward networks (top) and ResNets (bottom) when networks of depth $\ell = 1, 2, 3, 4$ are used.	158
8.A.2	Convergence of the error for different activation functions.	160
8.A.3	Approximation of the target function f for Example 8.A.3 using one (top), two (middle), and three (bottom) hidden layers in a ResNet architecture.	161
8.B.1	L^2 error in Galerkin neural networks (left) and PINNs (right).	162
8.B.2	Pointwise error in Galerkin neural networks and PINNs.	163

8.B.3 Pointwise error in Galerkin neural networks and PINNs.	164
--	-----

List of Tables

3.1	Displacement of a string. Condition number for the Galerkin matrix at each Galerkin iteration i	42
3.2	Displacement of a membrane. Condition number for the Galerkin matrix at each Galerkin iteration i . We observe that $\text{cond}(A^{(i)}) = \mathcal{O}(1)$	44

CHAPTER 1

Introduction

Partial differential equations (PDEs) model a wide array of phenomena that describe our physical world. The accurate approximation of PDEs is thus a subject of great interest to mathematicians and domain scientists alike, and numerical methods such as finite differences, finite volumes, and finite elements have been analyzed and refined over the last century in order to facilitate this task [27, 51, 65, 81, 84, 119]. More recently, there has been growing interest in the use of machine learning techniques, especially deep neural networks, to assist or even use as an alternative to traditional numerical methods [17, 71, 106, 123].

Deep neural networks (DNNs) in theory offer a number of attractive features that have facilitated their rise as one of the most active research areas in scientific computing today. First, they are known as universal approximators [33, 62, 101] in the sense that given a sufficiently smooth function, one can always find a neural network that approximates this function to within an arbitrary order of accuracy. Second, the nonlinear nature of neural networks allows them to be naturally adaptive to a wide class of functions, particularly those with singularities and sharp gradients. Indeed, one can even interpret ReLU neural networks as linear finite element methods whose underlying meshes are adapted as the neural network is trained [59]. Finally, DNNs allow practitioners to tackle

high-dimensional PDEs and applications in which (perhaps noisy) measurement data should be incorporated with the underlying governing equations.

However, in practice there are many issues that must be addressed in order that DNNs can be made a viable means of accurately approximating PDEs. The first is that there are numerous considerations that must be made regarding the PDE formulation itself. For example, many current approaches propose minimizing the strong residual of the PDE measured in a functional norm such as the L^2 -norm in order to train a DNN to approximate the PDE [42, 77, 106, 114]. While this approach is simple and straightforward to implement for a wide variety of problems, it is not well-posed in the presence of singular or low-regularity data. Other more mathematically sophisticated approaches operate on the variational formulation of the PDE, for instance [136], which minimizes the operator norm of the weak residual or [134], which minimizes the associated energy functional. Such approaches have the advantage of generally allowing for lower regularity data but are more computationally intensive, e.g., minimizing the operator norm of the weak residual equates to solving a saddle point min-max problem. The second major issue relates to error stagnation. It is frequently observed, irrespective of the PDE formulation considered, that the approximation error of DNNs stagnates and does not decrease beyond a certain threshold even when more parameters (i.e., neurons and layers) are added to the network [7, 49]. This is due in large part to the fact that DNNs must be trained by solving large optimization problems over highly non-convex landscapes. There has been a great deal of work done in the last decade in order to address these difficulties in numerical optimization, including improved optimization algorithms [43, 73, 137], improved initialization techniques [16, 138], and domain-specific initializers and optimizers that are well-suited to scientific computing applications [8, 35], and yet the development of techniques that allow for accurate training of DNNs is an active area of research in numerical optimization.

When considering the forward PDE problem, the aforementioned issues mean that,

even for many simple PDEs with smooth solutions, neural networks do not appear to confer any advantages in terms of efficiency or when compared with traditional methods, let alone for challenging real-world problems which contain features such as boundary layers and singularities. Indeed, problems exhibiting such features are typically resolved using techniques such as adaptive mesh refinement, *hp*-finite elements, etc. [19, 111]. While DNNs have thus far demonstrated a surprising ability to directly capture interesting features such as shocks and discontinuities in some scenarios [106], much of the literature focuses on relatively standard and straightforward model problems. Validation on more complex models is necessary in order to better understand and improve on neural network methodologies for the forward PDE problem.

With these issues in mind, the overarching goal of this work is the development of a new DNN framework for approximating PDEs that allows for rigorous control of the approximation error. To this end, we introduce Galerkin neural networks which seek to represent the solutions to the variational forms of PDEs as projections onto finite dimensional subspaces whose basis functions are the realizations of a sequence of DNNs.

1.1 Outline

An outline of this thesis is as follows. Chapter 3 introduces the basic concept of Galerkin neural networks in the limited setting of self-adjoint positive-definite problems and aims to address the issue of error stagnation and the low-order accuracy of DNN approximations of PDEs. It is based upon the published work [3]. Rather than attempting to approximate the solution using a single DNN, our approach expresses the solution as the sum of several very shallow networks which, roughly speaking, each encode different frequency scales in the solution; these shallow networks are easier to train and may be viewed as a sequence of corrections to an initial approximation.

In Chapter 4, we consider the application of Galerkin neural networks to a multiscale

singularly-perturbed elliptic problem in the form of Reissner-Mindlin plate deformation. The former exhibits behavior known as “locking” in which the numerical approximation is known to deteriorate for small values of a given parameter despite the true solution not being overly sensitive to the parameter value. We present several new variational formulations for the Reissner-Mindlin plate which demonstrate (1) that neural networks also suffer from convergence issues reminiscent of locking if improper formulations of the problem are utilized, and (2) the basic considerations required to construct locking-free formulations that are compatible with Galerkin neural networks and are capable of capturing desired multiscale features such as boundary layers. Chapter 4 is based upon the published work [4].

Chapter 5 presents an application of Galerkin neural networks to continuum models of suspension Poiseuille flows. This work offers a glimpse of how to extend Galerkin neural networks to formulations on anisotropic Sobolev spaces involving time and additionally provides validation of the method for problems involving chaotic and noisy data. This work appears in [64] (currently under review) and is part of a larger work examining machine learning methods for stress development in suspension flows. The author’s contribution to this project is the analysis and application of Galerkin neural networks to monodisperse and bidisperse suspensions in the case when physics-based models are available.

While Chapters 4 and 5 provide specific examples of how Galerkin neural networks may be extended to non-self-adjoint and/or non-positive-definite problems, we generalize this idea in Chapter 6 for a very general class of problems. Particular attention is given to problems with low-regularity data or solutions. Specifically, two novel ideas are presented in (1) weighted least squares variational formulations posed on high-order weighted Sobolev spaces which are suitable for problems with singular solutions, and (2) knowledge-based neural network architectures which utilize known solution structures that can be readily incorporated into the Galerkin neural network framework. Results are presented for several challenging examples including Stokes flow in a channel-driven cavity.

We conclude with a discussion of future directions of research and aspects of this work which the author considers particularly ripe for further investigation.

CHAPTER 2

Preliminaries

In this chapter, we briefly cover basic notation and concepts from functional analysis and deep learning that arise in subsequent chapters. Additionally, given the fast-paced nature of the field, the author finds it useful to summarize the relevant major works that utilize deep learning techniques to approximate the forward PDE problem.

2.1 Sobolev Spaces

In this work, we seek weak solutions to variational forms of PDEs. These variational forms will most commonly be posed on the Sobolev space $H^s(\Omega)$, $s \in \mathbb{N}$ [2].

Definition 2.1.1. *The **Sobolev space** $H^s(\Omega)$, $s \in \mathbb{N}$ is given by*

$$H^s(\Omega) := \left\{ v \in L^2(\Omega) : \int_{\Omega} |D^{\alpha} v|^2 dx < \infty, 0 \leq |\alpha| \leq s \right\}$$

and endowed with the norm

$$\|v\|_{H^s(\Omega)} = \|v\|_{s,\Omega} := \left(\sum_{0 \leq |\alpha| \leq s} \int_{\Omega} |D^{\alpha} v|^2 dx \right)^{1/2}.$$

Here, $D^\alpha v$ denotes the weak derivative of v . When $s = 0$, $H^s(\Omega)$ coincides with the space Lebesgue space $L^2(\Omega)$. For non-integer $0 < s < 1$, we have the Sobolev-Slobodeckij space.

Definition 2.1.2. For $0 < s < 1$, the **Sobolev-Slobodeckij space** H^s is given by

$$H^s(\Omega) := \left\{ v \in L^2(\Omega) : \int_{\Omega} \int_{\Omega} \frac{|v(x) - v(y)|^2}{|x - y|^{n+2s}} dx dy < \infty \right\}$$

and is endowed with the norm

$$\|v\|_{H^s(\Omega)} := \left(\|v\|_{L^2(\Omega)}^2 + \int_{\Omega} \int_{\Omega} \frac{|v(x) - v(y)|^2}{|x - y|^{n+2s}} dx dy \right)^{1/2}. \quad (2.1)$$

The singular integral in (2.1) is referred to as the Gagliardo seminorm [41]. Fractional-order Sobolev spaces arise as the natural range of the trace operator $\gamma : H^s(\Omega) \rightarrow H^{s-1/2}(\partial\Omega)$ provided that Ω is sufficiently regular.

Theorem 2.1.1. [90] Define the trace operator $\gamma : C_c^\infty(\bar{\Omega}) \rightarrow C_c^\infty(\partial\Omega)$ by

$$\gamma u = u|_{\partial\Omega}.$$

1. If Ω is a $C^{k-1,1}$ domain and if $1/2 < s \leq k$, then γ has a unique extension to bounded linear operator

$$\gamma : H^s(\Omega) \rightarrow H^{s-1/2}(\partial\Omega) \quad (2.2)$$

and this extension has a continuous right inverse.

2. If Ω is Lipschitz and $1/2 < s < 3/2$, then (2.2) holds and the extension has a continuous right inverse.

We will commonly use $H^{-s}(\Omega)$ to denote the dual space of $H^s(\Omega)$; that is, the space $(H^s(\Omega))^*$ of bounded linear functionals from $H^s(\Omega)$ to $\mathbb{R}$. The dual space is often the

most natural space for PDE data. For example, given the Poisson equation $-\Delta u = f$ in the domain Ω with suitable boundary conditions, a common assumption on the data f is $f \in H^{-1}(\Omega)$. As well, in this work we will often reference the Riesz representation theorem, which allows us to characterize linear functionals in $H^{-s}(\Omega)$ as integration against a suitable measure.

Theorem 2.1.2 (Riesz representation [48]). *Let H be a Hilbert space with inner product $(\cdot, \cdot)_H$. For each $f \in H^*$, there exists $\varphi_f \in H$ such that*

$$f(v) = (\varphi_f, v)_H \quad \forall v \in H.$$

Moreover, $\|f\|_{H^*} = \|\varphi_f\|_H$.

2.2 Neural Networks

We provide a precise mathematical description of feedforward DNNs here.

Definition 2.2.1. *Given an input $x \in \mathbb{R}^{d \times 1}$, the **feedforward neural network** consisting of ℓ hidden layers and corresponding widths $n_0, \dots, n_\ell$ is described by the sequence of transformations $T_i(x) = x \cdot W^{(i)} + b^{(i)}$, $i = 1, \dots, \ell - 1$ and $T_\ell(x) = x \cdot c$; and the function $u_{NN} : \mathbb{R}^d \rightarrow \mathbb{R}$*

$$u_{NN}(x; \theta) = T_\ell \circ (\sigma \circ T_{\ell-1} \cdots \circ \sigma \circ T_i \circ \cdots \circ \sigma \circ T_1)(x), \quad (2.3)$$

with $W^{(i)} \in \mathbb{R}^{n_{i-1} \times n_i}$ and $b^{(i)} \in \mathbb{R}^{1 \times n_i}$ the weights and biases of the network and $c \in \mathbb{R}^{n_\ell \times 1}$ the activation coefficients.

Collectively, the parameters $W^{(i)}$, $b^{(i)}$, and c are denoted by θ and are known as the trainable parameters of the network. The function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is known as the (nonlinear) activation function. A visualization of (2.3) is shown in Figure 2.1. The function in (2.3)

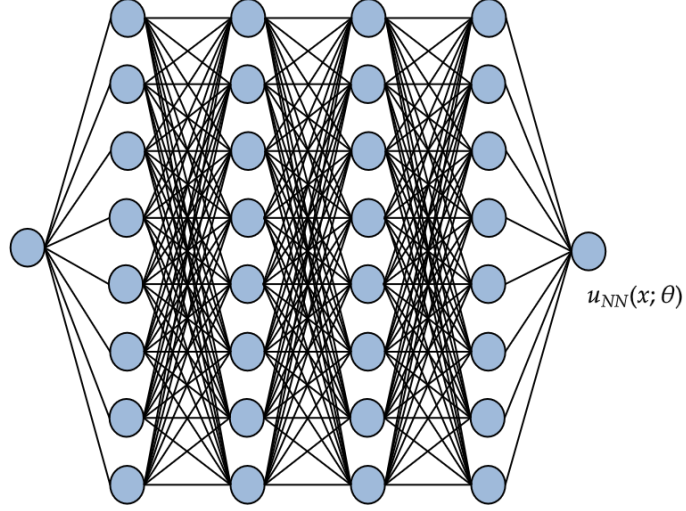


Figure 2.1: An example of a feedforward neural network with four hidden layers.

may be viewed as a linear combination of the nonlinear functions. For instance, if $\ell = 1$, we can expand u_{NN} as

$$u_{NN}(x; \theta) = \sum_{i=1}^{n_1} c_i \sigma(x \cdot W_i^{(1)} + b_i^{(1)}).$$

Remark 2.2.1. *We note that there are many other neural network architectures including but not limited to convolutional [56], residual [60], graph [139], and recurrent [46] neural networks. For the purposes of this work, we choose only to expound on feedforward DNNs as this is the only architecture we will consider, and indeed we find that this architecture is wholly sufficient for all of the applications studied here.*

Typically, one “learns” or “trains” the parameters θ to suit a particular task correlated with the minimization of a prescribed objective function. For instance, if one desires to approximate a function $f \in L^2(\Omega)$, a common choice of objective function is the L^2 norm of the residual $f(x) - u_{NN}(x; \theta)$. The parameters θ corresponding to the minimizer of this objective function are denoted by θ^* :

$$\theta^* = \arg \min_{\theta} \|f - u_{NN}(\cdot; \theta)\|_{L^2(\Omega)}^2. \quad (2.4)$$

In practice, (2.4) is approximated using numerical optimization techniques including but not limited to stochastic gradient descent [9], Adam [73], BFGS [128], etc. We make note also of the Levenberg-Marquardt algorithm [80, 89] which is a less popular option for approximating (2.4) due to its computational cost for larger problems. Nevertheless, the concept of viewing (2.4) as a nonlinear least squares problem bears some similarity to the training approach we propose in our work and is worth mentioning here.

Applying numerical optimization techniques to minimize the objective function associated with a neural network necessarily requires the computation of complicated derivatives with respect to the network parameters θ . While these derivatives may be approximated using finite differences, most modern software libraries and APIs – for instance, Tensorflow [1], PyTorch [102], and Jax [24] – utilize automatic differentiation [55] to accurately and cheaply evaluate derivatives.

2.2.1 Approximation Theory

Neural networks are known to have a high-degree of expressivity and are known as *universal approximators*. We summarize some results here which are utilized in our work. The universal approximation property was first explored by Cybenko in [33] in which he considered networks consisting of a single hidden layer and the sigmoid activation function $\sigma(t) = 1/(1 + \exp(-t))$.

Theorem 2.2.1 ([33]). *Let σ be a continuous function such that $\lim_{t \rightarrow -\infty} \sigma(t) = 0$ and $\lim_{t \rightarrow \infty} \sigma(t) = 1$. Then finite sums of the form*

$$G(x) = \sum_{i=1}^N c_i \sigma(x \cdot W_i + b_i) \quad (2.5)$$

are dense in $C([0, 1]^d)$.

Hornik, et. al. considered approximation results in the Sobolev space $H^s(\mathbb{R}^d, \mu)$, where

μ is a finite measure on $\mathbb{R}^d$. One can of course choose the measure $\mu(A) := \mu_{\mathcal{L}}(\Omega \cap A)$, where $A \subset \mathbb{R}^d$, $\mu_{\mathcal{L}}$ is the Lebesgue measure, and $\Omega \subset \mathbb{R}^d$ is an open and connected set with $\mu_{\mathcal{L}}(\Omega) < \infty$, which results in a finite measure μ for functions in the region of interest Ω .

Theorem 2.2.2 ([62]). *If $\sigma \in C^m(\mathbb{R}^d)$ is nonconstant and its derivatives up to order m are bounded, then sums of the form (2.5) are dense in $H^m(\mathbb{R}^d, \mu)$ for all finite measures μ .*

Leshno, et. al. [79] generalized these approximation results to provide necessary and sufficient conditions for universal approximation.

Theorem 2.2.3 ([79]). *Let $\sigma \in L_{loc}^\infty(\mathbb{R})$ and suppose the closure of points at which σ is discontinuous has Lebesgue measure zero. Then sums of the form (2.5) are dense in $C([0, 1]^d)$ if and only if σ is not an algebraic polynomial almost everywhere.*

We note that the proofs of these results are not constructive. They are accomplished by establishing contradictions and thus do not quantify how, for a given function $f \in H^s(\Omega)$, an error quantity such as $\|f - u_{NN}(\cdot; \theta)\|_{H^s(\Omega)}$ might behave with respect to the number of parameters in the neural network. Only somewhat recently have results been established that quantify this relationship, for example DNNs with the ReLU activation $\sigma(t) = \max\{0, t\}$ in [131]; explicit constructions for ReLU DNNs which demonstrate that such networks can achieve approximation rates at least as good as spectral and hp -adaptive spline approximations in [98]; and DNNs with tanh activation in [38]. We will note in the proceeding chapters where such results may be combined with our own approximation results.

2.2.2 Approximation of PDEs Based on the Strong Formulation

We briefly summarize past approaches to using DNNs to approximate PDEs. The first such approach was proposed in 1994 in [42] in which the authors considered a general

boundary value problem

$$\begin{cases} \mathcal{L}[u] = f & \text{in } \Omega \\ \mathcal{B}[u] = g & \text{on } \partial\Omega, \end{cases} \quad (2.6)$$

where $\Omega \subset \mathbb{R}^d$ and $\mathcal{L}$ and $\mathcal{B}$ are given differential operators. The authors propose to approximate $u(x) \approx u_{NN}(x; \theta)$ with the parameters θ learned by computing

$$\theta^* = \arg \min_{\theta} \|\mathcal{L}[u_{NN}(\cdot; \theta)] - f\|_{\ell^2}^2 + \|\mathcal{B}[u_{NN}(\cdot; \theta)] - g\|_{\ell^2}^2 \quad (2.7)$$

and achieved modest results on problems with smooth solutions using small feedforward networks.

It should be noted that the limiting factor in [42] was computing power. For instance, the largest networks considered were feedforward networks consisting of two hidden layers and 10 neurons in each layer for a total of 150 network parameters). This is in sharp contrast to modern applications and software libraries which are capable of quickly performing arithmetic operations of networks containing hundreds of thousands to millions of parameters. As well, derivatives, when necessary, were computed using finite differences and optimization was performed using a quasi-Newton method [128]. Today, practitioners most often utilize automatic differentiation to compute the necessary derivatives via software libraries such as Jax and Tensorflow [1, 24] while specialized optimization algorithms such as Adam [73] have become popular choices for a broad array of machine learning applications.

In [77], Lagaris, et al. build on the approach in [42] by considering trial functions for the solutions which automatically satisfy the boundary conditions. Namely, they consider

seeking a solution of the form

$$\Psi(x; \theta) = A(x) + F(x, u_{NN}(x; \theta)),$$

where A and F are chosen so that $\mathcal{B}[\Psi(x)] = g(x)$ and θ is computed as

$$\theta^* = \arg \min_{\theta} \|\mathcal{L}[\Psi(\cdot; \theta)] - f\|_{\ell^2}^2 + \|\mathcal{B}[\Psi(\cdot; \theta)] - g\|_{\ell^2}^2. \quad (2.8)$$

This approach in conjunction with the BFGS algorithm [47] allowed the authors to obtain modestly accurate approximations with small neural networks consisting of a single hidden layer and 10 neurons. However, it is restricted to simple geometries where even then, the boundary functions A and F can be quite complicated.

More modern approaches use the general approach of (2.7) in conjunction with much deeper and wider neural networks and improved numerical optimization techniques, see, for instance, the seminal physics-informed neural networks (PINNs) [106] which demonstrated the potential of neural networks for both forward and inverse PDE problems and considered significantly more challenging problems and contexts than had been considered prior. We also note the foundational work in [18, 85, 114].

2.2.3 Approximation of PDEs Based on the Weak Formulation

The approaches discussed thus far all consider the strong form of the PDE (2.6). In contrast, there are several works worth noting in the literature which leverage properties of the variational form of the PDE to develop neural network approaches to approximating (2.6). For instance, for self-adjoint problems the Deep Ritz method in [134] seeks a solution $u_{NN}(\cdot; \theta)$ whose parameters are obtained from minimizing the energy functional:

$$\theta^* = \arg \min_{\theta} \frac{1}{2} a(u_{NN}(\cdot; \theta), u_{NN}(\cdot; \theta)) - L(u_{NN}(\cdot; \theta)),$$

where $a : X \times X \rightarrow \mathbb{R}$ is a symmetric, continuous, and coercive bilinear operator and $L : X \rightarrow \mathbb{R}$ is a continuous linear operator associated with the variational problem

$$u \in X : a(u, v) = L(v) \quad \forall v \in X.$$

Another approach, the so-called Weak Adversarial Network [136], is to minimize the operator norm of the weak residual $\langle r(u), v \rangle = L(v) - a(u, v)$:

$$\theta_u^* = \arg \min_{\theta_u} \max_{\theta_v} \frac{\langle r(u_{NN}(\cdot; \theta_u)), v(\cdot; \theta_v) \rangle}{a(v(\cdot; \theta_v), v(\cdot; \theta))^{1/2}}.$$

The Weak Adversarial Network approach solves a saddle point min-max problem to obtain an approximate solution reminiscent of Generative Adversarial Networks (GANs) [53]. It allows for more general data such as $L = \delta$, but introduced another unknown in the test function v . The more recent weak physics-informed neural networks (wPINNs) [40] follows a similar approach to obtain entropy solutions of hyperbolic conservation laws. It should be noted, however, that the authors observed small spurious oscillations in the vicinity of shocks.

2.2.4 Other Approaches

While the work in this thesis is only concerned with approximating the forward PDE problem (2.6), we nevertheless conclude this section with a brief discussion of other neural network approaches that have interesting connections to the work presented in proceeding chapters.

A growing number of works have utilized machine learning techniques to accelerate traditional numerical methods. For instance, in [17] the authors utilize fully-connected convolutional networks to learn essentially non-oscillatory finite volume discretizations for conservation laws. In a similar vein, convolutional networks are used in [75] to learn the

lowest order terms of the truncation error in a finite difference scheme, which may then be used to supplement the discretization. These approaches are computationally inexpensive and leverage the approximation properties of the underlying numerical methods while utilizing DNNs to accelerate convergence. The recent work of [91] utilizes deep operator networks [87] to generate candidate basis functions for a spectral method.

Still further, DNNs have even been used in lieu of error indicators to guide adaptive mesh refinement for finite element methods. For instance, in [22] recurrent networks are shown to be capable of learning optimal mesh refinement strategies. Multilayer perceptrons have also been used as surrogates for slope limiters [107] and for learning tunable parameters in total-variation bounded (TVB) limiters for shock capturing [135]. In the same spirit, we note that the work in this thesis may be used to generate an a posteriori error estimator to guide adaptive mesh refinement (see Chapter 3, Section 3.2.2).

CHAPTER 3

The Galerkin neural network framework

3.1 Introduction

The current surge in interest in neural network approximation has witnessed the application of neural networks to a range of traditional problems in scientific computing and, in particular, the numerical solution of differential equations. The first problem encountered in applying neural networks to approximate the solution of a partial differential equation (PDE) is to decide on the basic formulation and how to choose the loss function to be used to train the network to approximate the true solution of the equation in hand. One popular formulation is to take the loss to be the square of the L^2 -norm of the residual in the equation augmented by additional terms to take account of the boundary conditions [18, 42, 77, 106]. Similar approaches also form the basis for least squares type finite element methods, which are more mainstream approaches based on formulating the PDE and boundary conditions as a variational equation.

Variational approaches for constructing neural network approximations of PDEs have been explored in [59, 72, 134, 136]. While such approaches have enjoyed success in

particular cases, in practice, the achievable relative error tends to stagnate around 0.1-1.0% irrespective of the number of neurons or layers used to define the underlying network architecture. In addition, despite recent progress [93, 114], the current level of theoretical support and understanding of the behavior and convergence of neural network schemes pales in comparison to the vast literature on more traditional approaches. It seems that, at the time of writing, neural network methods are unable to offer a competitive alternative to more traditional approaches such as finite elements or spectral methods.

The current work follows a quite different approach to using neural networks to approximate variational equations, based on the adaptive construction of a sequence of finite-dimensional subspaces whose basis functions are realizations of a sequence of neural networks. The finite-dimensional subspaces can be used to define a standard Galerkin approximation of the variational equation. This approach enjoys a number of advantages: the need to train a single, large network is replaced by the solution of a sequence of smaller problems combined with a Galerkin approach to select the best linear combination of the resulting corrections; the sequential nature of the algorithm offers a systematic approach to enhancing the accuracy of a given approximation (without having to resort to training a new, larger network); the sequential enhancements can be shown to provide a useful indicator for the error in the current neural network approximation that can be used as a criterion for terminating the sequential updates; and the basic approach is to some extent oblivious to the nature of the PDE under consideration beyond assuming that a stable variational formulation is available. A notable feature of the approach is that it may be applied seamlessly to a broad class of problems including PDEs of various orders and spatial dimensions without major modifications. One may contrast this with more traditional approaches, such as finite element methods, which generally require significant structural modification if the spatial dimension or order of the PDE is changed. Finally, some basic theoretical results are presented regarding the convergence (or otherwise) of the method which are used to formulate some basic guidelines for applying the method.

The remainder of this chapter is organized as follows. Section 3.2.1 summarizes some basic properties of Galerkin approximations that are needed in Section 3.2.2 where we describe a new approach for approximating the solutions of variational equations using neural networks. The main idea, developed in Section 3.2.3, is to adaptively construct a sequence of subspaces consisting of neural network functions from which a Galerkin approximation is obtained. Implementation details are provided in Section 3.2.4. The flexibility of the approach is illustrated in Section 3.3 by applying the method to approximate solutions of both second order and fourth order differential equations, including cases where the data and solutions are singular.

3.2 Basic Framework & Algorithm

A feed-forward neural network with a single hidden layer of $n \in \mathbb{N}$ neurons defines a function $u_{NN} : \mathbb{R}^d \rightarrow \mathbb{R}$ as follows:

$$u_{NN}(x; \theta) = \sum_{j=1}^n c_j \sigma(x \cdot W_j + b_j), \quad x \in \Omega \subset \mathbb{R}^d, \quad (3.1)$$

where $d \in \mathbb{N}$ is fixed; Ω is a compact subset of $\mathbb{R}^d$; $W_j \in \mathbb{R}^d$, $b_j \in \mathbb{R}$ are nonlinear parameters; $c_j \in \mathbb{R}$ are linear parameters; and $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is a smooth, bounded function. On occasion, we will use the shorthand notation $W = [W_1, \dots, W_n] \in \mathbb{R}^{d \times n}$, $b = [b_1, \dots, b_n] \in \mathbb{R}^{1 \times n}$, and $c = [c_1, \dots, c_n]^T \in \mathbb{R}^n$. In the standard nomenclature, n is referred to as the *width* of the network, W as the *weights*, b as the *biases*, and σ as the *activation function* of the hidden layer. The set of nonlinear and linear parameters is collectively denoted by $\theta = \{W, b, c\}$. The set of all functions of the form (3.1) is denoted by

$$V_n^\sigma := \{v \in W^{s,p}(\Omega) : v(x) = \sum_{i=1}^n c_i \sigma(x \cdot W_i + b_i), \quad b_i, c_i \in \mathbb{R}, \quad W_i \in \mathbb{R}^d, \quad x \in \Omega\}. \quad (3.2)$$

We shall also define the subset of V_n^σ consisting of functions with bounded parameters. Given $C > 0$, we define

$$V_{n,C}^\sigma := \{v \in V_n^\sigma : \|\theta\| \leq C\}, \quad (3.3)$$

where $\|\theta\| := \max_{ij} |W_{ij}| + \max_j |b_j| + \max_j |c_j|$.

Neural networks are known to be *universal approximators* [33, 79] in the sense that the following result holds:

Theorem 3.2.1 (Universal Approximation [62]). *Suppose $1 \leq p < \infty$, $0 \leq s < \infty$, and $\Omega \subset \mathbb{R}^d$ is compact. If $\sigma \in C^s(\Omega)$ is nonconstant and bounded, then V_n^σ is dense in the Sobolev space $W^{s,p}(\Omega)$.*

In particular, the universal approximation theorem means that given a function $f \in W^{s,p}(\Omega)$ and $\tau > 0$, there exists $n(\tau, f) \in \mathbb{N}$ and $\tilde{f} \in V_{n(\tau,f)}^\sigma$ such that $\|f - \tilde{f}\|_{W^{s,p}(\Omega)} < \tau$.

However, neural networks exhibit several undesirable properties as well. In particular, V_n^σ is not a vector space since it is not closed under addition. For instance, consider the case with $n = 1$ and $\sigma(t) = \max\{0, t\}$. Then $\sigma(t) \in V_1^\sigma$ and $\sigma(t-1) \in V_1^\sigma$, but $\sigma(t) + \sigma(t-1) \notin V_1^\sigma$. The fact that V_n^σ is not a vector space means we cannot draw upon many of the powerful results from functional analysis which are typically employed in analyzing the well-posedness of variational problems soon to be discussed. Moreover, V_n^σ is neither closed nor compact in $C(\Omega)$ or $L^p(\Omega)$. As demonstrated in [103] however, by restricting the network parameters as in $V_{n,C}^\sigma$, we can recover both closedness and compactness in $C(\Omega)$ and $L^p(\Omega)$. These topological properties of $V_{n,C}^\sigma$ become important in ensuring existence of a solution to the optimization problem we shall introduce in Section 3.2.2 (see Remark 3.2.2).

3.2.1 Galerkin Approximation of Variational Equations

Let $s \geq 0$ be fixed, let $X \subset H^s(\Omega) = W^{s,2}(\Omega)$ be a nonempty, closed subspace with $V_n^\sigma \subset X$ a subset, and consider the variational equation: find

$$u \in X : \quad a(u, v) = L(v), \quad \forall v \in X, \quad (3.4)$$

where $L : X \rightarrow \mathbb{R}$ is a bounded linear operator and $a : X \times X \rightarrow \mathbb{R}$ is a bounded, symmetric, bilinear form on X such that $a(\cdot, \cdot)$ is an equivalent inner product on $H^s(\Omega)$ with associated energy norm given by $|||v||| := \sqrt{a(v, v)}$.

A standard application of the Riesz Representation Theorem [48] implies the existence of a unique $u \in X$ satisfying (3.4) with $|||u||| = \|L\|_*$, where $\|\cdot\|_*$ denotes the dual, or operator, norm of L given by

$$\|L\|_* := \sup_{v \in X \setminus \{0\}} \frac{|L(v)|}{|||v|||} = \sup_{v \in B} |L(v)|, \quad (3.5)$$

with $B = \{v \in X : |||v||| = 1\}$ being the unit sphere in X . As $X \subset H^s(\Omega)$, we observe that the universal approximation property holds for the norm $|||\cdot|||$ whenever $s \in \mathbb{N}$ [62]. That is, given $f \in X$ and $\tau > 0$, there exists $n(\tau, f) \in \mathbb{N}$ and $\tilde{f} \in V_{n(\tau, f)}^\sigma$ such that $|||f - \tilde{f}||| < \tau$. The result is extended to $s \geq 0$ by the embedding $H^s(\Omega) \subset H^{\lfloor s \rfloor}(\Omega)$.

Remark 3.2.1. *The width n of the network needed to ensure $|||f - \tilde{f}||| < \tau$ depends on both τ and f . For instance, suppose $f \in H^2(0, 1)$ and σ is chosen as the ReLU function $t \mapsto \max\{0, t\}$. Standard results for piecewise linear approximation show that there exists a constant $\mathcal{C} > 0$ such that for all $m \in \mathbb{N}$, there exists $f_m \in V_m^\sigma$ satisfying $\|f - f_m\|_{L^2(0,1)} \leq \mathcal{C}m^{-2}\|f''\|_{L^2(0,1)}$. Hence, in order to achieve $\|f - f_m\|_{L^2(0,1)} < \tau$, it suffices to choose $m > \sqrt{\mathcal{C}\|f''\|_{L^2}/\tau}$. Observe that as the required tolerance decreases or the structure of f grows more complex, the width of the network needed to meet the tolerance will increase.*

In general, the solution of the variational problem cannot be obtained in closed form and instead must be approximated. Given a finite-dimensional subspace $S \subset X$, the Galerkin approximation $u_S \in S$ of problem (3.4) consists of seeking

$$u_S \in S : \quad a(u_S, v_S) = L(v_S) \quad \forall v_S \in S. \quad (3.6)$$

Equally well, we have $u_S = P_S u$, where $P_S : X \rightarrow S$ is the orthogonal projection given by the rule $a(P_S z, v_S) = a(z, v_S)$, $\forall v_S \in S$ for each $z \in X$. Galerkin approximation schemes enjoy widespread usage due to the fact that they are not only stable since $|||P_S u||| \leq |||u|||$, but deliver the best approximation to the true solution u from the subspace S [27], i.e.

$$|||u - u_S||| \leq |||u - v_S|||, \quad \forall v_S \in S. \quad (3.7)$$

Moreover, if $\{\varphi_j\}_{j=1}^m$ is a basis for S , then the approximation u_S may be computed explicitly by solving the linear system

$$\begin{cases} K \hat{u} = F \\ K_{ij} = a(\varphi_i, \varphi_j), \quad F_i = L(\varphi_i) \end{cases} \quad (3.8)$$

and forming $u_S = \sum_{j=1}^m \hat{u}_j \varphi_j$.

It is tempting to seek a neural network approximation $u_{NN} \approx u$ by employing a Galerkin scheme based on the choice $S = V_n^\sigma$. Unfortunately, the fact that V_n^σ is not a vector space nullifies much of the above discussion and precludes us from using it to define a Galerkin scheme of the form (3.6). Nonetheless, we observe that V_n^σ may still be utilized to generate approximations to u in two different ways.

First, by fixing the nonlinear parameters (W, b) and forming the subspace $\Phi := \text{span}\{\sigma(x \cdot W_i + b_i)\}_{i=1}^n$, a Galerkin approximation of the form (3.2) can be computed from

Φ by choosing the linear parameters c as follows:

$$\begin{cases} Kc = F \\ K_{ij} = a(\sigma(x \cdot W_i + b_i), \sigma(x \cdot W_j + b_j)), \quad F_i = L(\sigma(w \cdot W_i + b_i)). \end{cases} \quad (3.9)$$

This process defines a function $c = \text{GALERKINLSQ}(W, b, \sigma, a, L)$ which computes the expansion coefficients of the projection of u onto the subspace $\Phi = \text{span}\{\sigma(x \cdot W_i + b_i)\}_{i=1}^n$ for fixed W and b . In practice, however, the freedom to vary W and b is largely responsible for the rich approximation power of neural networks. Nevertheless, the function GALERKINLSQ will play a role in Section 3.2.4.

Second, if $\{\psi_i\}_{i=1}^m$ is a linearly independent set of functions in V_n^σ , then we may similarly define $\Phi := \text{span}\{\psi_1, \dots, \psi_m\}$ and solve the linear system

$$\begin{cases} K\hat{u} = F \\ K_{ij} = a(\psi_i, \psi_j), \quad F_i = L(\psi_i) \end{cases} \quad (3.10)$$

from which the projection $P_\Phi u = \sum_{j=1}^m \hat{u}_j \psi_j$ may be obtained. This process defines a function $P_\Phi u = \text{GALERKINSOLVE}(\{\psi_1, \dots, \psi_m\}, a, L)$ which returns the projection of u onto the subspace $\Phi := \text{span}\{\psi_1, \dots, \psi_m\}$ with respect to the variational equation defined by the bilinear form a and data L . The routine GALERKINSOLVE will play a key role in Section 3.2.3.

3.2.2 Construction of the Galerkin Subspace

Suppose $u_0 \in X$ is any given approximation to the solution of (3.4); even $u_0 = 0 \in X$ is allowed. Can we improve the accuracy of the approximation $u_0 \approx u$ by adding a suitable correction to u_0 from the set V_n^σ ?

We can measure how well u_0 approximates the true solution u by examining the residual $L(v) - a(u_0, v)$, $v \in X$ in (3.4), which vanishes if and only if $u_0 = u$. In general,

the residual will be nonzero and defines a functional $r(u_0) : X \rightarrow \mathbb{R}$ given by the rule $\langle r(u_0), v \rangle = L(v) - a(u_0, v)$, where $\langle r(u_0), v \rangle$ denotes the duality pairing on X .

The following result establishes the relationship between the residual $r(u_0)$ and the error $u - u_0$ in the approximation $u \approx u_0$.

Proposition 3.2.1. *Let $u_0 \in X$ be given and set $\varphi_1 := (u - u_0)/|||u - u_0|||$. Then $\langle r(u_0), \varphi_1 \rangle = \max_{v \in B} \langle r(u_0), v \rangle$, where B is the closed unit ball in X .*

Proof. Let $u_0 \in X$ be given. We use (3.4) to obtain

$$\langle r(u_0), v \rangle = L(v) - a(u_0, v) = a(u, v) - a(u_0, v) = a(u - u_0, v), \quad \forall v \in X.$$

By the Cauchy-Schwarz inequality,

$$\sup_{v \in B} \langle r(u_0), v \rangle = \sup_{v \in B} a(u - u_0, v) \leq \sup_{v \in B} |||u - u_0||| \cdot |||v||| = |||u - u_0|||.$$

On the other hand, choosing $v = \varphi_1 = (u - u_0)/|||u - u_0||| \in B$ yields

$$a(u_0 - u, v) = \frac{|||u - u_0|||^2}{|||u - u_0|||} = |||u - u_0|||,$$

and hence, $\max_{v \in B} \langle r(u_0), v \rangle = |||u - u_0||| = \langle r(u_0), \varphi_1 \rangle$. □

The elementary proof of Proposition 3.2.1 is illuminating in that it shows that the maximizer $\varphi_1 \in B$ of the residual is proportional to the error $e = u - u_0$. The optimal correction to u_0 is the true error e since $u_0 + e = u_0 + (u - u_0) = u$ yields the true solution. The fact that the maximizer φ_1 is proportional to the error means that we should take the correction to u_0 to be a multiple of φ_1 .

Unfortunately, we will not be able to compute the maximizer φ_1 explicitly which, as mentioned above, would be equivalent to solving the variational problem (3.4) exactly.

Instead, we compute an approximation $\varphi_1^{NN} \approx \varphi_1$ by seeking

$$\varphi_1^{NN} \in V_{n,C}^\sigma : \langle r(u_0), \varphi_1^{NN} \rangle = \max_{v \in V_{n,C}^\sigma \cap B} \langle r(u_0), v \rangle. \quad (3.11)$$

Remark 3.2.2. A function φ_1^{NN} as defined in (3.11) exists. Observe that

$$\sup_{v \in V_{n,C}^\sigma \cap B} \langle r(u_0), v \rangle \leq \sup_{v \in B} \langle r(u_0), v \rangle < \infty.$$

Moreover, $V_{n,C}^\sigma$ is compact in $H^s(\Omega)$ by [103, Proposition 3.5] and thus, $V_{n,C}^\sigma \cap B$ is compact in X . The supremum is therefore achieved over $V_{n,C}^\sigma \cap B$ and, as a consequence, the function φ_1^{NN} appearing in (3.11) is well-defined.

We then form the subspace $S_1 = \text{span}\{\varphi_0^{NN}, \varphi_1^{NN}\}$ where $\varphi_0^{NN} := u_0 / |||u_0|||$ and compute a Galerkin approximation to u by seeking

$$u_1 \in S_1 : a(u_1, v) = L(v) \quad \forall v \in S_1. \quad (3.12)$$

The following result demonstrates that if the network width n and the bound C are sufficiently large, then any such function $\varphi_1^{NN} \in V_{n,C}^\sigma$ as defined in (3.11) is a good approximation to φ_1 .

Proposition 3.2.2. Let $0 < \tau < 1$ be given. Then there exist $n(\tau, u_0) \in \mathbb{N}$ and $C(\tau, u_0) > 0$ such that if $n \geq n(\tau, u_0)$ and $C \geq C(\tau, u_0)$, then $\varphi_1^{NN} \in V_{n,C}^\sigma \cap B$ given by (3.11) satisfies $|||\varphi_1 - \varphi_1^{NN}||| < 2\tau/(1 - \tau)$.

Proof. We first recall from Proposition 3.2.1 that

$$\varphi_1 = \arg \max_{v \in B} \langle r(u_0), v \rangle = \arg \max_{v \in B} a(u - u_0, v).$$

By the universal approximation property, there exists $n(\tau, u_0) \in \mathbb{N}$ and $\tilde{\varphi} \in V_n^\sigma$ such that

$|||\varphi_1 - \tilde{\varphi}||| < \tau$. Furthermore, by the triangle inequality, we have $1 - \tau < |||\tilde{\varphi}||| < 1 + \tau$ or, equally well, $|||\tilde{\varphi}||| - 1 = \xi$ for some $\xi \in (-\tau, \tau)$. Let $\hat{\varphi} := \tilde{\varphi}/|||\tilde{\varphi}||| \in V_n^\sigma \cap B$. Then,

$$\begin{aligned}
|||\varphi_1 - \hat{\varphi}||| &\leq |||\varphi_1 - \tilde{\varphi}||| + |||\tilde{\varphi} - \frac{1}{|||\tilde{\varphi}|||}\tilde{\varphi}||| \\
&< \tau + \frac{1}{|||\tilde{\varphi}|||} |||\tilde{\varphi}||| \cdot |||\tilde{\varphi} - \tilde{\varphi}||| \\
&< \tau + \frac{1}{1 - \tau} |||(1 + \xi)\tilde{\varphi} - \tilde{\varphi}||| \\
&= \tau + \frac{\xi}{1 - \tau} |||\tilde{\varphi}||| \\
&< \tau + \frac{\tau}{1 - \tau} (1 + \tau) = \frac{2\tau}{1 - \tau}.
\end{aligned}$$

Set $C(\tau, u_0) := |||\hat{\theta}|||$, where $\hat{\theta}$ denotes the parameters corresponding to $\hat{\varphi} \in V_n^\sigma$.

Now, $\langle r(u_0), v \rangle = a(u - u_0, v) = \gamma a(\varphi_1, v)$ where $\gamma = |||u - u_0|||$. Let $n \geq n(\tau, u_0)$ and $C \geq C(\tau, u_0)$. The solution of (3.11) satisfies

$$\langle r(u_0), \varphi_1^{NN} \rangle = \max_{v \in V_{n,C}^\sigma \cap B} \langle r(u_0), v \rangle = \max_{v \in V_{n,C}^\sigma \cap B} \gamma a(\varphi_1, v).$$

Moreover, since $\gamma a(\varphi_1, v) \leq \gamma a(\varphi_1, \varphi_1^{NN})$ for all $v \in V_{n,C}^\sigma \cap B$,

$$\begin{aligned}
|||\varphi_1 - \varphi_1^{NN}|||^2 &= |||\varphi_1|||^2 - 2a(\varphi_1, \varphi_1^{NN}) + |||\varphi_1^{NN}|||^2 \\
&= 2 - 2\gamma^{-1} \cdot \gamma a(\varphi_1, \varphi_1^{NN}) \\
&\leq 2 - 2\gamma^{-1} \cdot \gamma a(\varphi_1, \hat{\varphi}) \\
&= |||\varphi_1 - \hat{\varphi}|||^2 < 4\tau^2/(1 - \tau)^2
\end{aligned}$$

as required. □

Proposition 3.2.2 shows that $\varphi_1^{NN} \in V_{n,C}^\sigma$ tends to the true maximizer φ_1 as the network width n increases. The next result shows that the corresponding approximation u_1 tends

to the true solution u as n and C increase.

Proposition 3.2.3. *Let $0 < \tau < 1$, $n(\tau, u_0) \in \mathbb{N}$, and $C(\tau, u_0)$ be as in Proposition 3.2.2, and u_1 be defined as in (3.12). Then $|||u - u_1||| \leq |||u - u_0||| \cdot \min\{1, 2\tau/(1 - \tau)\}$.*

Proof. Choosing $v_S = u_0 \in S_0$ in (3.7) yields $|||u - u_1||| \leq |||u - u_0|||$. Conversely, choosing $v_S = |||u_0||| \cdot \varphi_0^{NN} + |||u - u_0||| \cdot \varphi_1^{NN} \in S_0$ and recalling that $\varphi_1 = (u - u_0)/|||u - u_0|||$ yields

$$\begin{aligned} |||u - u_1||| &\leq |||(u - u_0) - |||u - u_0||| \cdot \varphi_1^{NN}||| \\ &= |||\varphi_1 \cdot |||u - u_0||| - |||u - u_0||| \cdot \varphi_1^{NN}||| \\ &= |||u - u_0||| \cdot |||\varphi_1 - \varphi_1^{NN}||| \\ &< |||u - u_0||| \cdot \frac{2\tau}{1 - \tau}, \end{aligned}$$

where the final step follows thanks to Proposition 3.2.2. $\square$

Proposition 3.2.3 gives a bound on the accuracy of the approximation $u \approx u_1$ in terms of τ and the (unknown) error in $u \approx u_0$. Interestingly, Proposition 3.2.1 shows that $|||u - u_0||| = \langle r(u_0), \varphi_1 \rangle$ which, were φ_1 available, would mean the actual error could be computed explicitly. Does replacing φ_1 by the approximation $\varphi_1 \approx \varphi_1^{NN}$ then result in a good estimator for the true error? To be precise, we define the quantity

$$\eta(u_0, v) := \langle r(u_0), v \rangle / |||v||| \quad (3.13)$$

and observe that $\eta(u_0, \varphi_1^{NN})$ can be evaluated explicitly once φ_1^{NN} and u_0 are in hand. The following result shows that, provided $\tau < 1/3$, the numerical value of $\eta(u_0, \varphi_1^{NN})$ is indeed equivalent to the true error $|||u - u_0|||$.

Corollary 3.2.1. *Let $\eta(u_0, v)$ be as defined in (3.13). If $0 < \tau < 1/3$, then*

$$\frac{1-\tau}{1+\tau}\eta(u_0, \varphi_1^{NN}) \leq |||u - u_0||| \leq \frac{1-\tau}{1-3\tau}\eta(u_0, \varphi_1^{NN}). \quad (3.14)$$

Proof. By Proposition 3.2.1, we have $|||u - u_0||| = \langle r(u_0), \varphi_1 \rangle$. Hence,

$$|\eta(u_0, \varphi_1^{NN}) - |||u - u_0||| | = |\langle r(u_0), \varphi_1^{NN} \rangle - \langle r(u_0), \varphi_1 \rangle| = |a(u - u_0, \varphi_1^{NN} - \varphi_1)|,$$

and applying Proposition 3.2.2 yields

$$|\eta(u_0, \varphi_1^{NN}) - |||u - u_0||| | \leq |||u - u_0||| \cdot |||\varphi_1^{NN} - \varphi_1||| < |||u - u_0||| \cdot \frac{2\tau}{1-\tau}.$$

If $\tau \in (0, 1/3)$ then $2\tau/(1-\tau) \in (0, 1)$, and hence

$$\eta(u_0, \varphi_1^{NN}) \leq |||u - u_0||| \cdot \left(\frac{2\tau}{1-\tau} + 1 \right) = |||u - u_0||| \cdot \frac{1+\tau}{1-\tau}$$

and

$$\eta(u_0, \varphi_1^{NN}) \geq |||u - u_0||| \cdot \left(-\frac{2\tau}{1-\tau} + 1 \right) = |||u - u_0||| \cdot \frac{1-3\tau}{1-\tau}.$$

Combining these estimates gives (3.14). □

In summary, once the solution φ_1^{NN} of (3.11) is in hand, we can (a) obtain an improved approximation to $u \approx u_1$ by solving (3.12) and (b) obtain a numerical estimate $\eta(u_0, \varphi_1^{NN}) \approx |||u - u_0|||$ for the error. The foregoing arguments show that if $\tau \rightarrow 0$, or, equally well, the network width $n \rightarrow \infty$ and bound $C \rightarrow \infty$, then the resulting approximation $u_1 \rightarrow u$ (in X) and $\eta(u_0, \varphi_1^{NN}) \rightarrow |||u - u_0|||$ (in $\mathbb{R}$).

3.2.3 Galerkin Subspace Augmentation

The procedure outlined in the preceding section gives an improved approximation $u_1 = P_{S_1}u$ to u from the two-dimensional subspace $S_1 = \text{span}\{\varphi_0^{NN}, \varphi_1^{NN}\}$. How should we proceed if the approximation $u \approx u_1$ is lacking? A natural extension of this approach would be to apply the same procedure that was used to improve the approximation $u \approx u_0$ to improve u_1 . Namely, we compute an approximate maximizer φ_2^{NN} of the new residual $\langle r(u_1), v \rangle$ and use it to further augment the Galerkin subspace by setting $S_2 = \text{span}\{\varphi_0^{NN}, \varphi_1^{NN}, \varphi_2^{NN}\}$. Proceeding recursively, we construct the finite-dimensional subspace $S_j = \text{span}\{\varphi_0^{NN}, \varphi_1^{NN}, \varphi_2^{NN}, \dots, \varphi_j^{NN}\}$ with $\varphi_i^{NN} \in V_{n_i, C_i}^{\sigma_i}$.

Here, we have allowed for the possibility to vary both the network width n_i and activation function σ_i . Proceeding as in Section 3.2.2, we define $\varphi_i = (u - u_{i-1}) / |||u - u_{i-1}|||$ and compute an approximation to φ_i by seeking

$$\varphi_i^{NN} \in V_{n_i, C_i}^{\sigma_i} : \langle r(u_{i-1}), \varphi_i^{NN} \rangle = \max_{v \in V_{n_i, C_i}^{\sigma_i} \cap B} \langle r(u_{i-1}), v \rangle \quad (3.15)$$

and then computing a new approximation $u \approx u_i$ as follows:

$$u_i \in S_i : a(u_i, v) = L(v) \quad \forall v \in S_i, \quad (3.16)$$

or, equally well, $u_i = P_{S_i}u$.

We have the following analogs to Propositions 3.2.1, 3.2.2, and 3.2.3 and Corollary 3.2.1.

Corollary 3.2.2. *For each $i > 0$, let $u_i = P_{S_i}u$ be given and define $\varphi_i := (u - u_{i-1}) / |||u - u_{i-1}|||$. Then $\langle r(u_{i-1}), \varphi_i \rangle = \max_{v \in B} \langle r(u_{i-1}), v \rangle$.*

Corollary 3.2.3. *For each $i > 0$, let $0 < \tau_i < 1$ be given. Then there exists $n(\tau_i, u_{i-1}) \in \mathbb{N}$ and $C(\tau_i, u_{i-1}) > 0$ such that if $n \geq n(\tau_i, u_{i-1})$ and $C \geq C(\tau_i, u_{i-1})$, then the function*

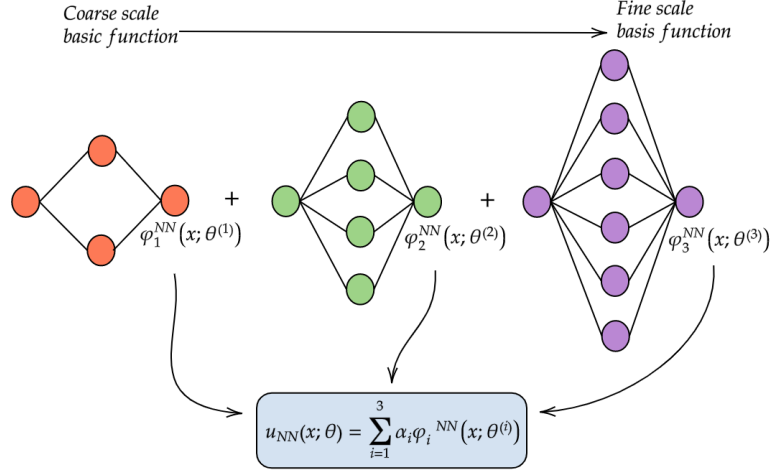


Figure 3.1: A visualization of the Galerkin Neural Network framework using three shallow networks consisting of two, four, and six neurons each.

$\varphi_i^{NN} \in V_{n,C}^{\sigma_i}$ as defined in (3.15) satisfies $|||\varphi_i - \varphi_i^{NN}||| < 2\tau_i/(1 - \tau_i)$. Moreover, if $\tau_i < 1/3$, then

$$\frac{1 - \tau_i}{1 + \tau_i} \eta(u_{i-1}, \varphi_i^{NN}) \leq |||u - u_{i-1}||| \leq \frac{1 - \tau_i}{1 - 3\tau_i} \eta(u_{i-1}, \varphi_i^{NN}).$$

In particular, Corollary 3.2.3 provides a computable estimate of the error $|||u - u_{i-1}|||$ that can be used as a stopping criterion for the recursive solution procedure as follows: Given a tolerance $\text{tol} > 0$, if $\eta(u_{i-1}, \varphi_i^{NN}) > \text{tol}$, then we augment the subspace S_{i-1} to $S_i = S_{i-1} \oplus \{\varphi_i^{NN}\}$ and compute an updated approximation $u_i \approx u$; otherwise, the approximation u_{i-1} is deemed satisfactory, and the procedure terminates. Algorithm 1 summarizes this recursive approach, where the function AUGMENTBASIS is the routine which computes the maximizer φ_i^{NN} in the subspace augmentation process and which will be described in Section 3.2.4.

The next result addresses the convergence, or otherwise, of the recursive procedure.

Corollary 3.2.4. *Let $j \geq 1$. If $\{\tau_i\}_{i=1}^j$, $\{n(\tau_i, u_{i-1})\}_{i=1}^j$, and $\{C(\tau_i, u_{i-1})\}_{i=1}^j$ are chosen*

Algorithm 1: Adaptive Construction of Subspace.

Input: Data L , bilinear operator a , network widths $\{n_i\}$, initial approximation $u_0 \in X$, tolerance $\text{tol} > 0$.

Output: Stopping iteration $N \in \mathbb{N}$, numerical approximation u_N to the variational problem: $u \in X$ such that $a(u, v) = \langle f, v \rangle$ for all $v \in X$; basis functions $\{\varphi_i^{NN}\}_{i=0}^N$.

```

1 Require:
2    $a$  symmetric positive-definite, continuous, and coercive.

3 Set  $i = 1$  and  $\varphi_0^{NN} = u_0 / |||u_0|||$ .
4  $(\varphi_1^{NN}, \eta) \leftarrow \text{AUGMENTBASIS}(u_0)$ .
5 while  $\eta > \text{tol}$  do
6    $u_i \leftarrow \text{GALERKINSOLVE}(\{\varphi_0^{NN}, \varphi_1^{NN}, \dots, \varphi_i^{NN}\}, a, L)$ 
7    $(\varphi_{i+1}^{NN}, \eta) \leftarrow \text{AUGMENTBASIS}(u_i)$ .
8   Set  $i \leftarrow i + 1$ .
9 end
10 Return  $N = i - 1$ ,  $u_N$ , and  $\{\varphi_j^{NN}\}_{j=0}^N$ .
```

as in Corollary 3.2.3, then

$$|||u - u_j||| \leq |||u - u_0||| \cdot \prod_{i=1}^j \min\{1, \frac{2\tau_i}{1 - \tau_i}\}.$$

Proof. Proposition 3.2.3 implies that for each i : $|||u - u_i||| \leq |||u - u_{i-1}||| \cdot \min\{1, 2\tau_i/(1 - \tau_i)\}$. Bootstrapping this result gives

$$\begin{aligned}
|||u - u_j||| &\leq |||u - u_{j-1}||| \cdot \min\{1, \frac{2\tau_j}{1 - \tau_j}\} \\
&\vdots \\
&\leq |||u - u_0||| \cdot \prod_{i=1}^j \min\{1, \frac{2\tau_i}{1 - \tau_i}\}
\end{aligned}$$

as claimed. □

Lastly, we discuss the conditioning of the linear system $K^{(j)}\hat{u} = F^{(j)}$ associated with the approximation $u \approx u_j$ in (3.16). This linear system follows the framework described

by (3.10). The matrix $K^{(j)}$ has entries

$$K_{k\ell}^{(j)} = a(\varphi_{k-1}^{NN}, \varphi_{\ell-1}^{NN}),$$

where the basis functions $\{\varphi_i^{NN}\}_{i=0}^j$ are obtained according to Algorithm 1. We observe that if $k = \ell$, then $K_{kk}^{(j)} = 1$ since $\varphi_k^{NN} \in B$.

Proposition 3.2.4. *Let $\{\varphi_i^{NN}\}_{i=0}^N$ be the set of basis functions constructed by Algorithm 1. Then for $i > 1$, $\varphi_i^{NN} \notin S_{i-1}$. Moreover, if $\gamma = \sum_{k=1}^N 2\tau_k/(1 - \tau_k) < 1$, then*

$$\text{cond}(K^{(N)}) < \frac{1 + \gamma}{1 - \gamma}. \quad (3.17)$$

Proof. Suppose, to the contrary, that $\varphi_i^{NN} \in S_{i-1}$. Then (3.16) implies $L(\varphi_i^{NN}) - a(u_{i-1}, \varphi_i^{NN}) = \eta(u_{i-1}, \varphi_i^{NN}) = 0$. But the stopping criterion in Algorithm 1 (see Line 5) ensures that each φ_i^{NN} satisfies $\eta(u_{i-1}, \varphi_i^{NN}) > \text{tol}$, a contradiction. Thus, $\varphi_i^{NN} \notin S_{i-1}$ and, in particular, the functions $\{\varphi_i^{NN}\}_{i=0}^N$ are linearly independent.

For the second assertion, if $k > \ell$, then we have $a(\varphi_k, \varphi_\ell^{NN}) \propto a(u - u_{k-1}, \varphi_\ell^{NN}) = 0$ by (3.16), and applying Corollary 3.2.3 yields

$$\begin{aligned} |a(\varphi_k^{NN}, \varphi_\ell^{NN})| &= |a(\varphi_k^{NN}, \varphi_\ell^{NN}) - a(\varphi_k, \varphi_\ell^{NN})| = |a(\varphi_k^{NN} - \varphi_k, \varphi_\ell^{NN})| \\ &\leq |||\varphi_k^{NN} - \varphi_k||| \cdot |||\varphi_\ell^{NN}||| < \frac{2\tau_k}{1 - \tau_k}. \end{aligned}$$

Thus, $|K_{\ell k}^{(N)}| = |K_{k\ell}^{(N)}| < 2\tau_\ell/(1 - \tau_\ell)$ for $k \neq \ell$ thanks to the symmetry of $a(\cdot, \cdot)$.

By Gershgorin's Theorem [50], for each eigenvalue λ of $K^{(N)}$ there exists ℓ such that

$$|\lambda - K_{\ell\ell}^{(N)}| = |\lambda - 1| < \sum_{k \neq \ell} |K_{k\ell}^{(N)}| < \sum_{k \neq \ell} \frac{2\tau_k}{1 - \tau_k}.$$

Then $|\lambda - 1| < \gamma$ for each λ , and

$$\text{cond}(K^{(N)}) = \frac{|\max \lambda|}{|\min \lambda|} < \frac{1 + \gamma}{1 - \gamma}$$

as claimed. $\square$

In summary, the procedure given by Algorithm 1 adaptively constructs a subspace using the basis functions $\varphi_i^{NN} \in V_{n_i, C_i}^{\sigma_i}$, $i = 1, \dots, N$. By Corollary 3.2.4, the resulting Galerkin approximation u_N converges to u provided that the widths n_i and bounds C_i are sufficiently large. Moreover, the off-diagonal entries of $K^{(N)}$ are small provided that n_i and C_i are sufficiently large. In particular, $\text{cond}(K^{(N)}) \approx 1$.

3.2.3.1 Selection of Network Widths

Algorithm 1 allows for the flexibility to vary both the width n_i of the network and the activation function σ_i as i is increased. Surprisingly, we can fix n and set $n_i \equiv n$ and yet still hope to achieve convergence as $i \rightarrow \infty$:

Proposition 3.2.5. *Let $n \in \mathbb{N}$ be fixed and $\Omega \subset \mathbb{R}^d$ compact. Suppose $f \in H^s(\Omega)$, $s \in \mathbb{R}_{\geq 0}$. For $\tau > 0$, there exists $r \in \mathbb{N}$ and $\{u_j\}_{j=1}^r \in V_n^\sigma$ such that $\|f - \sum_{j=1}^r u_j\|_{H^s} < \tau$.*

Proof. By the universal approximation property, there exists $n(\tau, f) \in \mathbb{N}$ and $\hat{f} \in V_{n(\tau, f)}^\sigma$ such that $\|f - \hat{f}\|_{H^s} < \tau$. Choose r such that $rn > n(\tau, f)$. Then there exists $\tilde{f} \in V_{nr}^\sigma$ such that $\|f - \tilde{f}\|_{H^s} < \tau$, where $\tilde{f}(x) = \sum_{k=1}^{nr} c_k \sigma(x \cdot W_k + b_k) = \sum_{j=1}^r u_j(x)$, with $u_j(x) = \sum_{k > (j-1)n}^{jn} c_k \sigma(x \cdot W_k + b_k)$. $\square$

Proposition 3.2.5 shows that any $f \in H^s(\Omega)$ can be approximated by a sum of functions in V_n^σ to arbitrary accuracy. In principle, this means fixing $n_i \equiv n$ and carrying out sufficiently many iterations of Algorithm 1 should give convergence. However, in practice the rate of convergence will become arbitrarily slow.

For instance, let us again consider the scenario described in Remark 3.2.1, where it was found that a ReLU network of width $n(\tau, f) = \sqrt{C\|f''\|_{L^2}/\tau}$ was sufficient to approximate a function $f \in H^2(0, 1)$ to a tolerance τ in $L^2(0, 1)$. The proof of Proposition 3.2.5 shows that a sum of $r \geq n^{-1}\sqrt{C\|f''\|_{L^2}/\tau}$ functions in V_n^σ achieves a tolerance of τ . In other words, $\mathcal{O}(\tau^{-1/2})$ iterations of Algorithm 1 are needed to achieve tolerance τ . This means that, on average, the error is reduced by a factor of $\tau^{\sqrt{\tau}}$ per iteration of the algorithm during this phase. However, in the limit as $\tau \rightarrow 0$, the average reduction factor tends to 1, i.e. the algorithm converges arbitrarily slowly.

A similar conclusion is reached if one considers Corollary 3.2.4 with $n_i = n$ is fixed. Specifically, the sequence $\{\tau_i\}_{i=1}^j$ will generally be increasing due to the decreased ability of a fixed-width network $\varphi_i^{NN} \in V_{n,C}^\sigma$ to capture higher resolution features of the error as i increases. In other words, there exists $i^* \in \mathbb{N}$ such that $\min\{1, 2\tau_i/(1 - \tau_i)\} = 1$ for all $i \geq i^*$, meaning that the convergence stagnates eventually.

On the other hand, Corollary 3.2.4 also suggests how we might avoid this stagnation. Observe that if $0 < \tau < 1/3$ is fixed and $\tau_i = \tau$ for all i , then it is expected that the sequence $\{n(\tau, u_i)\}_{i=0}^j$ is nondecreasing. In particular, the error in u_i is a factor of $2\tau/(1 - \tau)$ smaller than the error in u_{i-1} . In order for $\varphi_i^{NN} \in V_{n(\tau, u_{i-1}), C(\tau, u_{i-1})}^{\sigma_i}$ to capture the higher resolution features of $(u - u_{i-1})/|||u - u_{i-1}|||$ (as compared to the relatively lower resolution features of $(u - u_{i-2})/|||u - u_{i-2}|||$), we necessarily require that $n(\tau, u_{i-1}) > n(\tau, u_{i-2})$. Hence, by increasing the width n_i of the networks as i increases, we have $|||u - u_j||| \leq |||u - u_0||| \cdot (2\tau/(1 - \tau))^j$.

3.2.4 Neural Network Approximation of Augmented Basis Function

In order to compute the augmented basis function $\varphi_i^{NN} \in V_{n_i, C_i}^{\sigma_i} \cap B$ according to (3.15), we first note that functions $v \in V_{n_i, C_i}^{\sigma_i}$ do not necessarily satisfy $|||v||| = 1$. We thus

normalize by forming $v/|||v|||$:

$$\max_{v \in V_{n_i, C_i}^{\sigma_i} \cap B} \langle r(u_{i-1}), v \rangle = \max_{v \in V_{n_i, C_i}^{\sigma_i} \setminus \{0\}} \langle r(u_{i-1}), \frac{v}{|||v|||} \rangle = \max_{v \in V_{n_i, C_i}^{\sigma_i} \setminus \{0\}} \eta(u_{i-1}, v), \quad (3.18)$$

where $\eta(u_{i-1}, v)$ is as defined in (3.13). Next, we note that any $v \in V_{n_i, C_i}^{\sigma_i}$ is fully characterized by the set of parameters $\theta^{(i)} = (W^{(i)}, b^{(i)}, c^{(i)})$. Thus, (3.18) becomes

$$\max_{v \in V_{n_i, C_i}^{\sigma_i} \setminus \{0\}} \eta(u_{i-1}, v) = \max_{\substack{\theta^{(i)} \\ \theta \neq (0,0,0) \\ ||\theta^{(i)}|| < C_i}} \eta(u_{i-1}, v(\theta^{(i)})). \quad (3.19)$$

where $\eta(u_{i-1}, v)$ is as defined in (3.13). Next, we note that any $v \in V_{n_i, C_i}^{\sigma_i}$ is fully characterized by the set of parameters $\theta^{(i)} = (W^{(i)}, b^{(i)}, c^{(i)})$. Thus, (3.18) becomes

$$\max_{v \in V_{n_i, C_i}^{\sigma_i} \setminus \{0\}} \eta(u_{i-1}, v) = \max_{\substack{\theta^{(i)} \\ \theta \neq (0,0,0) \\ ||\theta^{(i)}|| < C_i}} \eta(u_{i-1}, v(\theta^{(i)})). \quad (3.20)$$

In practice, we do not enforce boundedness of $\theta^{(i)}$. While boundedness can be enforced by selecting a large value of C_i and clipping the network parameters should they exceed this value during training, we find that for the examples in Section 3.3, the network parameters are uniformly bounded without explicitly enforcing $||\theta^{(i)}|| < C_i$.

The standard approach to computing optima of objective functions associated with neural networks is to apply a gradient-based optimizer to all of the parameters in θ , for example [18, 42, 72, 77, 106, 134, 136]. In order to solve (3.15), we instead propose a training procedure in the spirit of [35] as follows:

1. On the first training iteration, the hidden parameters $(W^{(i)}, b^{(i)})$ are initialized appropriately, which we describe on a case-by-case basis in Section 3.3. Otherwise, the hidden parameters are updated via a gradient-based optimizer. For instance,

Algorithm 2: Basis Function Generation and Error Estimation.

Input: Data L , bilinear operator a , network width n_i , activation function σ_i , initial approximation $u_{i-1} \in X$.

Output: Numerical approximation φ_i^{NN} to the maximizer $\arg \max_{v \in B} \langle r(u_i), v \rangle = (u - u_{i-1}) / |||u - u_{i-1}|||$, and error estimator $\eta(u_{i-1}, \varphi_i^{NN}) \approx (u - u_{i-1}) / |||u - u_{i-1}|||$

```

1 Function AugmentBasis( $u_j$ ):
2   Initialize hidden parameters  $(W^{(i)}, b^{(i)}) \in \mathbb{R}^{d \times n_i} \times \mathbb{R}^{n_i}$ 1.
3   Compute corresponding activation coefficients:
       $c^{(i)} \leftarrow \text{GALERKINLSQ}(W^{(i)}, b^{(i)}, \sigma_i, a, L(\cdot) - a(u_{i-1}, \cdot))$ .
4   for each training epoch do
5     Compute the update rule for  $(W^{(i)}, b^{(i)})$ :
          
$$W^{(i)} \leftarrow W^{(i)} + \alpha \nabla_{W^{(i)}} [\eta(u_{i-1}, v(\theta^{(i)}))]$$

          
$$b^{(i)} \leftarrow b^{(i)} + \alpha \nabla_{b^{(i)}} [\eta(u_{i-1}, v(\theta^{(i)}))]$$

          Compute corresponding activation coefficients:
           $c^{(i)} \leftarrow \text{GALERKINLSQ}(W^{(i)}, b^{(i)}, \sigma_i, a, L(\cdot) - a(u_{i-1}, \cdot))$ .
6   end
7   Return  $\varphi_i^{NN} = v(\theta^{(i)}) / |||v(\theta^{(i)})|||$  and  $\eta(u_{i-1}, \varphi_i^{NN})$ .
8 EndFunction

```

with gradient descent, the update is described by

$$W^{(i)} \leftarrow W^{(i)} + \alpha \nabla_{W^{(i)}} [\eta(u_{i-1}, v(\theta^{(i)}))] \quad (3.21)$$

$$b^{(i)} \leftarrow b^{(i)} + \alpha \nabla_{b^{(i)}} [\eta(u_{i-1}, v(\theta^{(i)}))] \quad (3.22)$$

with learning rate α .

2. We form $\Phi := \text{span}\{\sigma(x \cdot W_j + b_j)\}_{j=1}^n$ and take the projection $v(\theta^{(i)}) \leftarrow P_\Phi(u - u_{i-1})$ as described by $\text{GALERKINLSQ}(W^{(i)}, b^{(i)}, \sigma_i, a, L(\cdot) - a(u_{i-1}, \cdot))$.

The solution of the linear system (3.9) with the data $L(\cdot) - a(u_{i-1}, \cdot)$ provides the expansion coefficients of the projection of the error $u - u_{i-1}$ onto Φ with respect to a . Specifically, by (3.7), $P_\Phi(u - u_{i-1})$ is the best approximation to $u - u_{i-1}$ from Φ .

The training procedure for φ_i^{NN} is summarized in Algorithm 2.

3.2.4.1 Implementation Details

Most problems in which the bilinear operator a is associated with the weak formulation of a PDE require the computation of L^2 inner products in Ω . For instance, consider the case with $a(u, v) := (u, v)_{L^2(\Omega)}$ and $L(v) := (f, v)_{L^2(\Omega)}$ where $f \in L^2(\Omega)$. In order to compute the objective function $\eta(u_0, v(\theta)) = \langle r(u_0), v(\theta) \rangle / |||v(\theta)|||$, we approximate the associated integrals by employing a quadrature rule consisting of nodes $\{x_i\}_{i=1}^{n_G} \subset \Omega$ and weights $\{w_i\}_{i=1}^{n_G}$:

$$\int_{\Omega} v(x) \, dx \approx \sum_{i=1}^{n_G} w_i v(x_i).$$

We refer the reader to [36] for further details on the derivation, accuracy, and convergence of such quadrature rules.

The objective function in the above example is approximated by

$$\frac{L(v(\theta)) - a(u_0, v(\theta))}{|||v(\theta)|||} \approx \frac{\sum_{i=1}^{n_G} w_i [f(x_i)v(x_i; \theta) - u_0(x_i)v(x_i; \theta)]}{\sqrt{\sum_{i=1}^{n_G} w_i v(x_i; \theta)^2}}.$$

In all examples in Section 3.3, a single step of the Adam optimizer [73] with full batch is employed to update the hidden parameters $\{W, b\}$ at each training epoch following Algorithm 1. In other words, the training data essentially consists of the set of n_G quadrature nodes which are used to evaluate the loss function.

For most numerical examples, the activation function σ_i at Galerkin iteration i is adaptive in the sense that we utilize a scaling factor as a hyperparameter [68]. Given a fixed nonpolynomial activation function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$, we define $\sigma_i(t) = \sigma(\beta_i t)$, where $\{\beta_i\}$ is a nondecreasing sequence in $\mathbb{R}_+$. As β_i grows larger, the gradient of σ_i generally increases, which allows for better resolution of the fine-scale error features as we shall demonstrate in Section 3.3. Finally, in most of the proceeding examples, the network widths $\{n_i\}$ follow a geometric progression of the form $n_i = N \cdot r^{i-1}$, where N is the initial

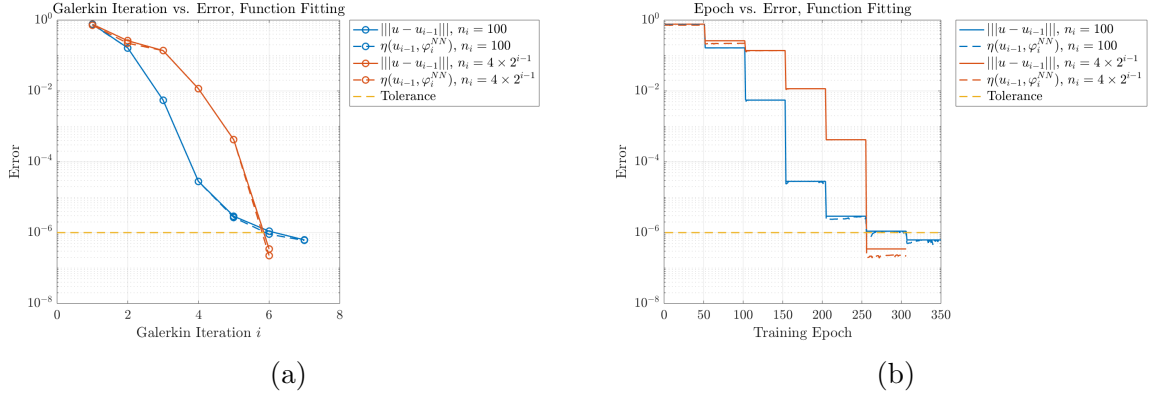


Figure 3.2: (a) Estimated and true L^2 error at each Galerkin iteration. (b) The piecewise constant segments (solid lines) denote the true errors for each Galerkin iteration i , while the dashed red line (resp. blue line) denotes the progress of the loss function in approximating the true errors within each Galerkin iteration for increasing network widths (resp. constant network widths). The x -axis thus denotes the cumulative training epoch over all Galerkin iterations.

width for the first Galerkin iteration r is the geometric ratio. The learning rate is fixed for each Galerkin iteration i and follows an exponential decay of the form $\alpha_i = A \cdot \rho^{-(i-1)}$, where A is the initial learning rate for the first Galerkin iteration and ρ is the rate of decay. Standard neural network approaches where one trains only a single fixed-architecture network commonly utilize a decreasing learning rate schedule as the training epochs progress. In our approach, we only decrease the learning rate *between* each Galerkin iteration to compensate for the increasing widths n_i as described in Section 3.2.3.1.

3.3 Numerical Examples

We apply Algorithm 1 to a number of problems in one and two dimensions, demonstrating the robustness of our algorithm for several different classes of problems as well as the robustness of the error indicator $\eta(u_{i-1}, \varphi_i^{NN})$.

3.3.1 Function Fitting

We consider the following illuminating example in order to investigate whether stagnation of the error $\|u - u_i\|$ occurs when $n_i \equiv n$ is fixed: given $f \in L^2(\Omega)$, $\Omega = (0, 1)$,

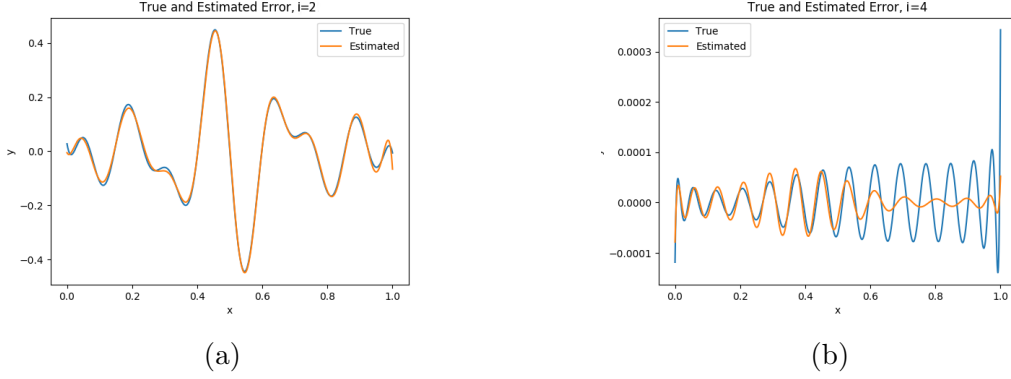


Figure 3.3: (a) True error $u - u_{i-1}$ and approximate error φ_j^{NN} for $i = 2$. (b) True error $u - u_{i-1}$ and approximate error φ_i^{NN} for $i = 4$ with fixed width $n_i = 100$.

let $a(u, v) := (u, v)_{L^2(\Omega)}$ and $L(v) := (f, v)_{L^2(\Omega)}$ with f being the first four terms of the Fourier series for the square wave:

$$f(x) = \sin(x) + \frac{1}{3} \sin(3\pi x) + \frac{1}{5} \sin(5\pi x) + \frac{1}{7} \sin(7\pi x).$$

This is a simple function fitting problem whose exact solution is $u = f$. For this and all subsequent examples, we set $u_0 = 0$. The loss function is approximated by

$$\frac{L(v(\theta)) - a(u_0, v(\theta))}{|||v(\theta)|||} \approx \frac{\sum_{i=1}^{n_G} w_i [f(x_i)v(x_i; \theta) - u_0(x_i)v(x_i; \theta)]}{\sqrt{\sum_{i=1}^{n_G} w_i v(x_i; \theta)^2}}.$$

We use the hyperbolic tangent activation function $\sigma_i(t) = \tanh(\beta_i t)$, $\beta_i = 1 + 3(i - 1)$

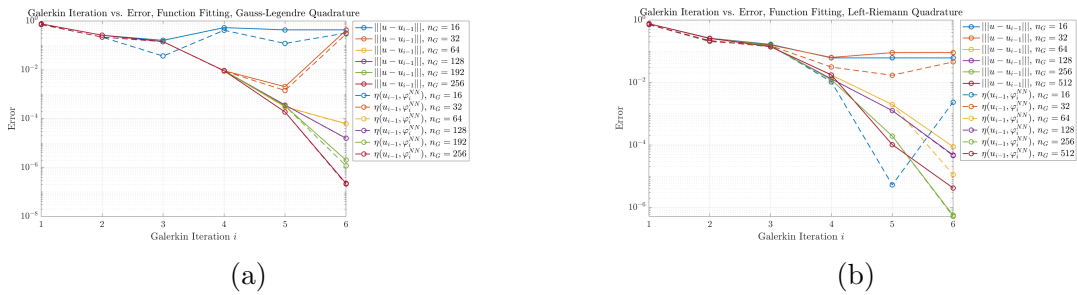


Figure 3.4: (a) Error at each Galerkin iteration i using a Gauss-Legendre quadrature rule with varying number of nodes (b) the analogous results using a left Riemann-sum with varying number of nodes.

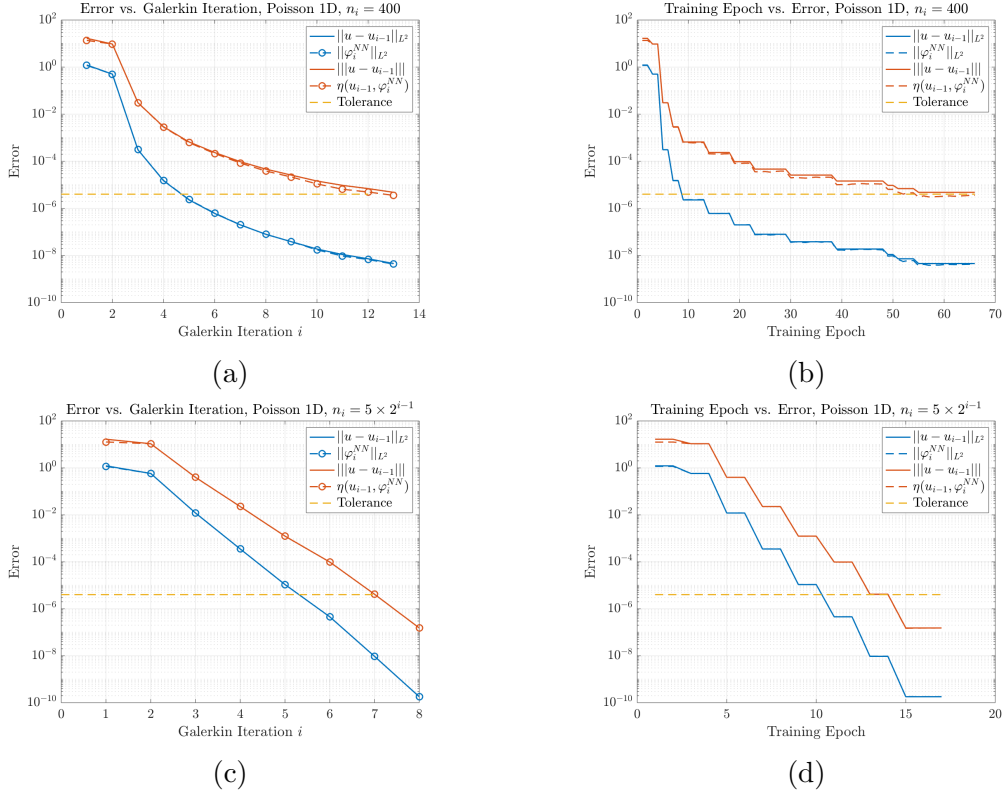


Figure 3.5: Displacement of a string (Poisson equation in 1D). (a) Estimated and true error in the L^2 and energy norms at each Galerkin iteration, $n_i = 400$ for all i . (b) The piecewise constant segments (solid lines) denote the true errors for each Galerkin iteration i , while the dashed red line (resp. blue line) denotes the progress of the loss function (resp. L^2 error estimator $\|\varphi_i^{NN}\|_{L^2}$) in approximating the true errors within the each Galerkin iteration with $n_i = 400$. The x -axis thus denotes the cumulative training epoch over all Galerkin iterations. (c) and (d) The analogous results for $n_i = 5 \times 2^{i-1}$.

at each Galerkin iteration and consider both the case when successive networks have fixed widths $n_i = 100$ for all i and the case when successive networks have increasing widths $n_i = 4 \times 2^{i-1}$. We set $\text{tol} = 10^{-6}$ and adaptively construct a Galerkin subspace that achieves an approximation with $\eta(u_{i-1}, \varphi_i^{NN}) < \text{tol}$. The hidden parameters are initialized such that $W_j^{(i)} = 1$ and $b_j^{(i)} = -jh$, $h = 1/n_i$ i.e. the biases are spaced uniformly throughout Ω . To evaluate the objective function and train the network, we employ a fixed Gauss-Legendre quadrature rule with 512 nodes to approximate all inner products. A separate Gauss-Legendre quadrature rule with 1000 nodes is used for validation, i.e., to compute the true errors.

Figure 3.2a shows the true L^2 error at each Galerkin iteration along with the estimator $\eta(u_{i-1}, \varphi_i^{NN})$ while Figure 3.2 shows the true and estimated L^2 error at each epoch. Figure 3.3 shows the true error $u - u_{i-1}$ and estimated error φ_i^{NN} for $i = 2$ and $i = 4$ with $n_i = 100$. It is apparent that initially, the low frequency components of the error are learned, with later iterations learning the high frequency error components. For $i = 2$, the low frequency error components are sufficiently learned by the network. However, for $i = 4$, the network lacks sufficient capacity to learn the high frequency error components, thus leading to inadequate training of the network and stagnation in the error. On the other hand, we observe from Figure 3.2 that there is no stagnation of the error in the later Galerkin iterations if n_i is increased with i . Moreover, the case when n_i is increased at each Galerkin iteration eventually achieves the same accuracy as the case when n_i is fixed at each Galerkin iteration while also achieving better computational efficiency due to utilizing initial networks with smaller widths.

Of course, the quadrature rule used to approximate the loss function must be chosen sufficiently accurately. Figure 3.4 shows the convergence results obtained as the number of Gauss-Legendre quadrature points increases as well as a comparison with a left-Riemann sum quadrature rule. In each case, we observe that if the number of nodes is too small, the error either stagnates or increases eventually due to the inability of the quadrature rule to adequately evaluate the inner products in the loss function and (3.9) and (3.10).

3.3.2 Second Order Problems

3.3.2.1 Displacement of a String

We now illustrate the approach for a simple linear ODE:

$$\begin{cases} -u'' = f, & \text{in } \Omega = (0, 1) \\ u + \varepsilon \partial_n u = 0, & \text{on } \partial\Omega, \end{cases}$$

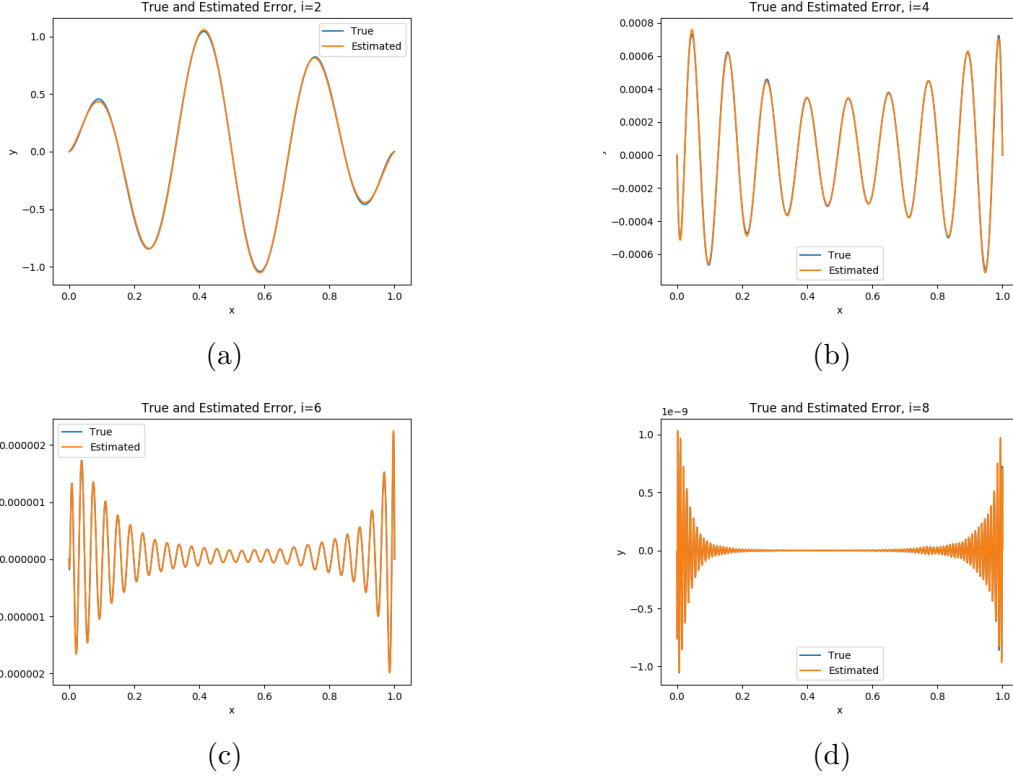


Figure 3.6: Displacement of a string (Poisson equation in 1D). True error $u - u_{i-1}$ and estimated error φ_i^{NN} for $i = 2, 4, 6, 8$.

where $\varepsilon > 0$ and $f(x) = (2\pi)^2 \sin(2\pi x) + (4\pi)^2 \sin(4\pi x) + (6\pi)^2 \sin(6\pi x)$, with f inducing a linear functional given by the rule $L(v) := (f, v)_\Omega$. The variational formulation is posed on the Hilbert space $X = H^1(\Omega)$ corresponding to the bilinear operator $a(u, v) = (u', v')_\Omega + \varepsilon^{-1}(u(0)v(0) + u(1)v(1))$. One can easily verify that a is continuous and coercive with respect to $\|\cdot\|_{H^1(\Omega)}$. Observe that as $\varepsilon \rightarrow 0$, the derivative terms in the boundary condition vanish and we obtain $u(0) \rightarrow 0$ and $u(1) \rightarrow 0$, recovering the standard Poisson equation with homogeneous Dirichlet boundary conditions. The parameter ε may thus be regarded as a penalization term which approximately enforces the homogeneous boundary condition $u(0) = u(1) = 0$. Alternatively, as $\varepsilon \rightarrow \infty$ we obtain $\partial_n u \rightarrow 0$ on $\partial\Omega$, and ε may be regarded as approximately imposing a homogeneous Neumann boundary condition. The true solution is

$$u(x) = \sin(2\pi x) + \sin(4\pi x) + \sin(6\pi x) + \frac{1}{1 + 2\varepsilon} (-24\pi\varepsilon x + 12\pi\varepsilon).$$

i	1	2	3	4	5	6	7	8	9	10
$\text{cond}(K^{(i)})$	1.00	1.00	1.00	1.00	1.02	1.03	1.07	1.08	1.09	1.18

Table 3.1: Displacement of a string. Condition number for the Galerkin matrix at each Galerkin iteration i .

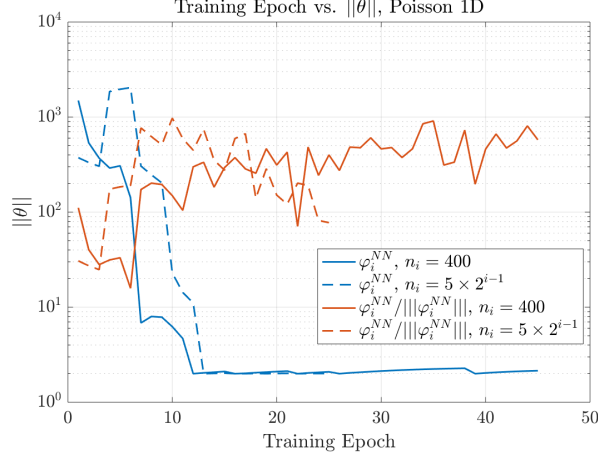


Figure 3.7: Displacement of a string. The total norm of the network parameters for φ_i^{NN} and $\varphi_i^{NN} / ||\varphi_i^{NN}||$.

The corresponding loss function is approximated by

$$\frac{L(v(\theta)) - a(u_0, v(\theta))}{|||v(\theta)|||} \approx \frac{L_1 - L_2 - L_3}{\sqrt{\sum_{i=1}^{n_G} w_i v'(x_i; \theta)^2 + \varepsilon^{-1}(v(0; \theta)^2 + v(1; \theta)^2)}}$$

where $L_1 = \sum_{i=1}^{n_G} w_i f(x_i) v(x_i; \theta)$, $L_2 = \sum_{i=1}^{n_G} w_i u'_0(x_i) v'(x_i; \theta)$, and $L_3 = \varepsilon^{-1}[u_0(0) \cdot v(0; \theta) + u_0(1) v(1; \theta)]$.

The tolerance is set to $\text{tol} = 2 \times 10^{-6}$, and $\varepsilon = 10^{-4}$. The network architecture for Galerkin iteration i is as follows: $n_i = 400$, $\sigma_i(t) = \tanh(\beta_i t)$, $\beta_i = i$. The learning rate is $\alpha = 2 \times 10^{-2}$. The hidden parameters are initialized in the same manner as in Section 3.3.1. For comparison, we shall also consider the set of network architectures given by $n_i = 5 \times 2^{i-1}$ and learning rate $\alpha = (2 \times 10^{-2})/1.1^{i-1}$. To evaluate the objective function and train the network, we employ a fixed Gauss-Legendre quadrature rule with 512 nodes to approximate all inner products. A separate Gauss-Legendre quadrature rule with 1000 nodes is used for validation.

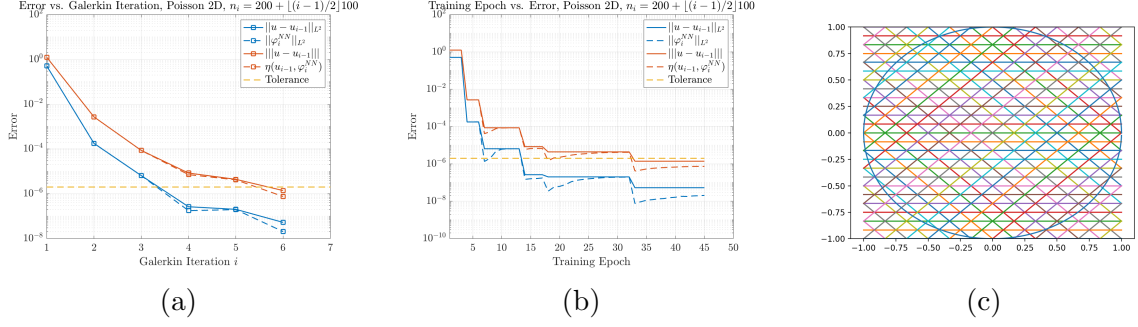


Figure 3.8: Displacement of a membrane (Poisson equation in 2D). Estimated and exact error in the L^2 and energy norms at each Galerkin iteration (a) and epoch (b). Initial hyperplanes (c).

Figure 3.5 shows the true errors $\|u - u_{i-1}\|_{L^2}$ and $\|u - u_{i-1}\|$ at the end of each Galerkin iteration as well as the corresponding error estimates $\|\varphi_i^{NN}\|_{L^2}$ and $\eta(u_{i-1}, \varphi_i^{NN})$. We also provide the analogous results after each training epoch. Figure 3.6 shows the exact error $u - u_{i-1}$ as well as the maximizer φ_i^{NN} at several stages of the algorithm. We observe first that the initial Galerkin iterations reduce the error substantially, with further decreases beginning to stagnate as i grows larger whenever n_i is constant; stagnation is not an issue when n_i is increased with i . Moreover, both estimators $\eta(u_{i-1}, \varphi_i^{NN})$ and $\|\varphi_i^{NN}\|_{L^2}$ provide accurate estimates of the error, with negligible discrepancies beginning to show in the final Galerkin iterations. Figure 3.7 shows the total norm of the network parameters of φ_i^{NN} and $\varphi_i^{NN}/\|\varphi_i^{NN}\|$ at each training epoch. We observe that in both the normalized and unnormalized case, the network parameters are well-bounded and do not exhibit explosive behavior. Lastly, Table 3.1 shows the condition number of the matrix $K^{(j)}$ at each Galerkin iteration i . We observe that the condition number is approximately unity as expected.

i	1	2	3	4	5	6	7
$\text{cond}(K^{(i)})$	1.00	1.00	1.00	1.00	1.01	1.36	1.37

Table 3.2: Displacement of a membrane. Condition number for the Galerkin matrix at each Galerkin iteration i . We observe that $\text{cond}(A^{(i)}) = \mathcal{O}(1)$.

3.3.2.2 Displacement of a Membrane

We consider the Poisson equation in the unit disk:

$$\begin{cases} -\Delta u = f, & \text{in } \Omega = \{(x, y) : x^2 + y^2 < 1\} \\ u + \varepsilon \partial_n u = 0, & \text{on } \partial\Omega, \end{cases}$$

where, $f(x, y) = 2$. The variational formulation is posed on the Hilbert space $X = H^1(\Omega)$ corresponding to the bilinear operator $a(u, v) = (\nabla u, \nabla v)_\Omega + \varepsilon^{-1}(u, v)_{\partial\Omega}$ with data $L(v) := (f, v)_\Omega$. One can easily verify that a is continuous and coercive with respect to $\|\cdot\|_{H^1(\Omega)}$. The true solution is

$$u(x, y) = -\frac{x^2 + y^2}{2} + \varepsilon + \frac{1}{2}.$$

The corresponding loss function is approximated by

$$\frac{L_1 - L_2 - L_3}{\sqrt{\sum_{i=1}^{n_\Omega} w_i |\nabla v(x_i^\Omega; \theta)|^2 + \varepsilon^{-1} \sum_{i=1}^{n_{\partial\Omega}} w_i^{\partial\Omega} v(x_i^{\partial\Omega}; \theta)^2}} \quad (3.23)$$

where $L_1 = \sum_{i=1}^{n_\Omega} w_i^\Omega f(x_i) v(x_i; \theta)$, $L_2 = \sum_{i=1}^{n_\Omega} w_i^\Omega \nabla u_0(x_i^\Omega) \cdot \nabla v(x_i^\Omega; \theta)$, and also $L_3 = \varepsilon^{-1} \sum_{i=1}^{n_{\partial\Omega}} w_i^{\partial\Omega} u_0(x_i^{\partial\Omega}) v(x_i^{\partial\Omega}; \theta)$. Here, $\{w^\Omega, x^\Omega\}$ is the quadrature rule in Ω while $\{w^{\partial\Omega}, x^{\partial\Omega}\}$ is the quadrature rule on $\partial\Omega$.

The tolerance is set to $\text{tol} = 2 \times 10^{-6}$, and $\varepsilon = 10^{-4}$. The network architecture for Galerkin iteration i is as follows: $n_i = 200 + \lfloor \frac{i-1}{2} \rfloor \cdot 100$, $\sigma_i(t) = \tanh(t)$. The learning rate is $\alpha = (1 \times 10^{-2})/1.1^{i-1}$. The hidden parameters are initialized such that the hyperplanes lie parallel to $y = 0$, $x = 0$, $y = x$, or $y = -x$ as visualized in Figure 3.8 (right). To

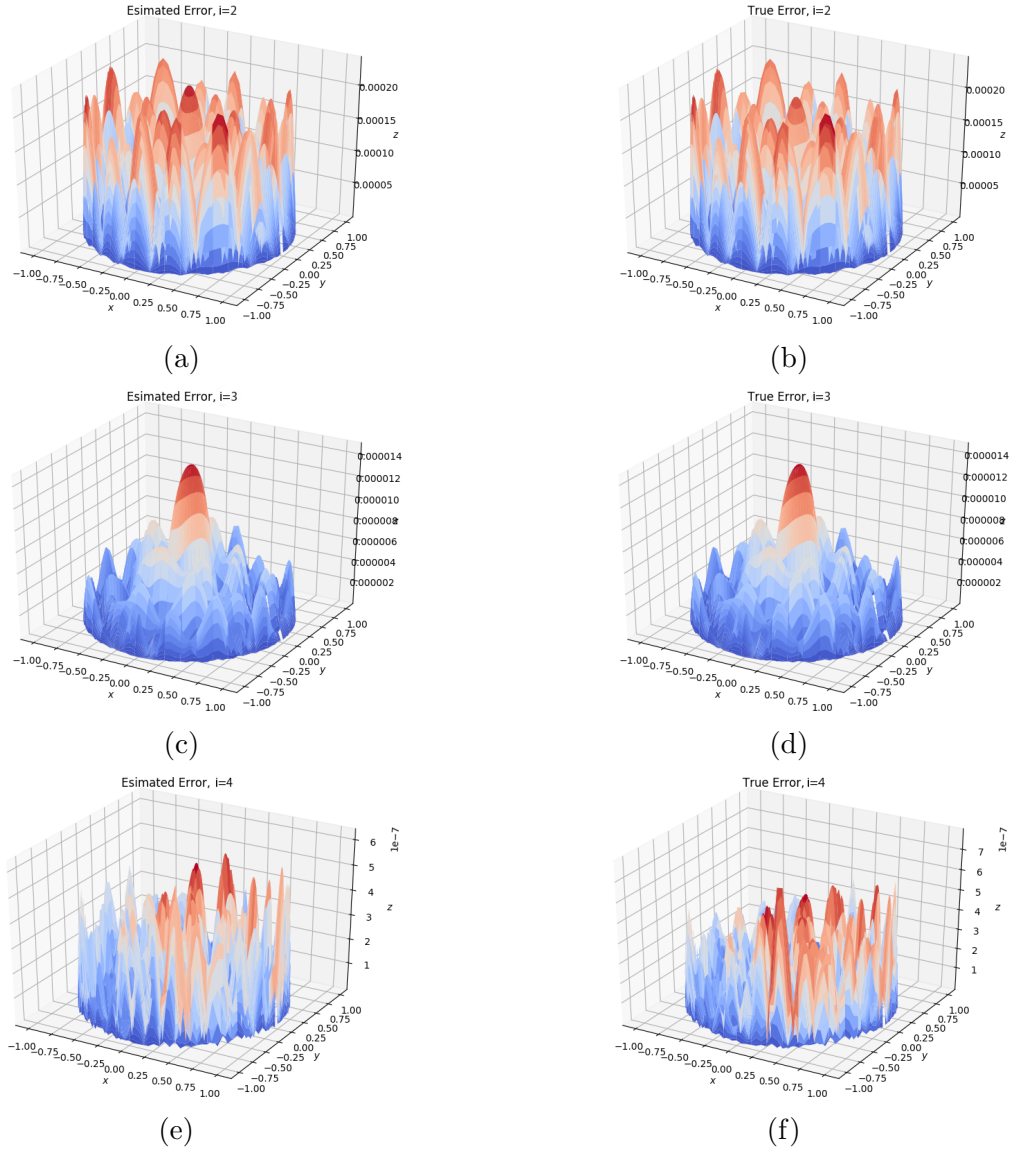


Figure 3.9: Displacement of a membrane (Poisson equation in 2D). Exact error $u - u_{i-1}$ (right column) and approximate error φ_i^{NN} (left column) for $i = 2, 3, 4$.

evaluate the loss function and train the network, we employ a fixed Gauss-Legendre quadrature rule with 128×128 nodes to approximate all inner products in the interior of the domain. A quadrature rule with 256 equally-spaced nodes is employed to approximate inner products on the boundary of the domain.

Figure 3.8 (left) shows the exact errors $\|u - u_{i-1}\|_{L^2}$ and $\|u - u_{i-1}\|$ at the end of each epoch along with the estimated errors $\|\varphi_i^{NN}\|_{L^2}$ and $\eta(u_{i-1}, \varphi_i^{NN})$ for our algorithm. Figure 3.9 shows the exact error $u - u_{i-1}$ (right column) as well as the computed maximizer φ_i^{NN}

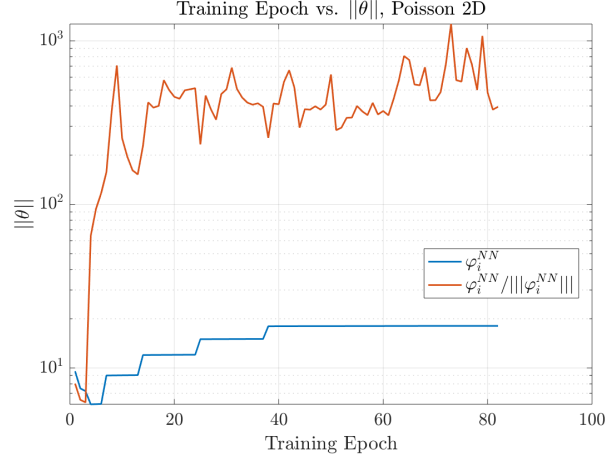


Figure 3.10: Displacement of a membrane. The total norm of the network parameters for φ_i^{NN} and $\varphi_i^{NN} / ||\varphi_i^{NN}||$.

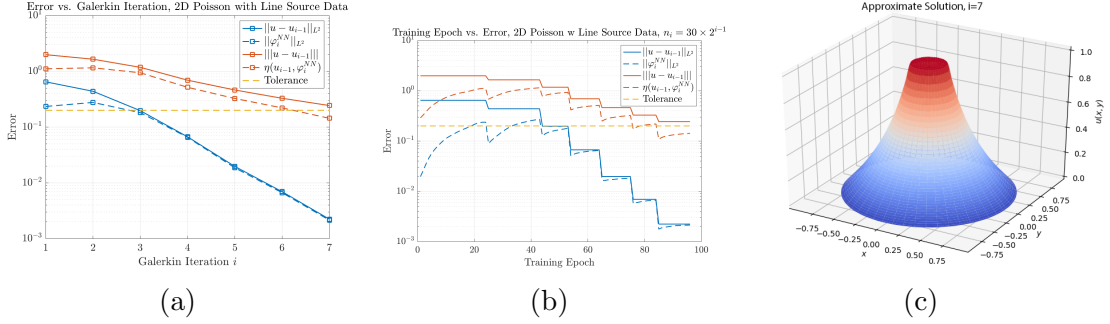


Figure 3.11: Poisson equation with line source data, $R_e = 1 - 1/\pi^2$ and $R_0 = 1/\sqrt{29}$. (a) true and estimated errors after each Galerkin iteration (b) true and estimated errors after each training epoch (c) approximate solution u_i for $i = 7$.

(left column) at several stages of our algorithm. We again observe good performance of the error estimator, with minor discrepancies beginning to show in the final Galerkin iteration.

Figure 3.10 shows the total norm of the network parameters of φ_i^{NN} and $\varphi_i^{NN} / ||\varphi_i^{NN}||$ at each training epoch, and we again observe boundedness of the parameters. Table 3.2 shows the condition number of the Galerkin matrix $K^{(i)}$ at Galerkin iteration i . We observe again that $\text{cond}(K^{(i)})$ is approximately unity.

3.3.2.3 Poisson Equation with Line Source Data

We once more consider the Poisson equation in two dimensions, this time in the domain $\Omega = \{(x, y) : x^2 + y^2 < R_e^2\}$ with data $L : H^1(\Omega) \rightarrow \mathbb{R}$ defined by

$$L(v) := \int_{\Gamma} v \, ds, \quad \Gamma = \{(x, y) : x^2 + y^2 = R_0^2\}, \quad R_0 < R_e.$$

The exact variational solution is constant in the inner disk $\{(x, y) : x^2 + y^2 \leq R_0^2\}$ and logarithmic in the annulus $\Omega \setminus \{(x, y) : x^2 + y^2 \leq R_0^2\}$:

$$u(r, \theta) = -\frac{\varepsilon}{R_0 \log(R_0/R_e)} + \begin{cases} 1, & r \leq R_0 \\ \frac{1}{\log(R_0/R_e)} \log(r/R_e), & R_0 < r \leq R_e. \end{cases}$$

We consider two separate cases:

1. $R_e = 1 - 1/\pi^2$ and $R_0 = 1/\sqrt{29}$,
2. $R_e = 1 - 1/\pi^2$ and $R_0 = 7/(6\sqrt{2})$.

In the first case, the logarithmic behavior dominates with a sharp cusp in the solution at $r = R_0$. In the second case, the solution still has a sharp cusp at $r = R_0$, but the logarithmic behavior is limited to the small annulus $R_0 \leq r \leq R_e$, which resembles a boundary layer. The corresponding loss function is approximated by

$$\frac{L_1 - L_2 - L_3}{\sqrt{\sum_{i=1}^{n_{\Omega}} w_i |\nabla v(x_i^{\Omega}; \theta)|^2 + \varepsilon^{-1} \sum_{i=1}^{n_{\partial\Omega}} w_i^{\partial\Omega} v(x_i^{\partial\Omega}; \theta)^2}}$$

where $L_1 = \sum_{i=1}^{n_{\Gamma}} w_i^{\Gamma} v(x_i^{\Gamma}; \theta)$, $L_2 = \sum_{i=1}^{n_{\Omega}} w_i^{\Omega} \nabla u_0(x_i^{\Omega}) \cdot \nabla v(x_i^{\Omega}; \theta)$, and also $L_3 = \varepsilon^{-1} \cdot \sum_{i=1}^{n_{\partial\Omega}} w_i^{\partial\Omega} u_0(x_i^{\partial\Omega}) v(x_i^{\partial\Omega}; \theta)$. Here, $\{w^{\Omega}, x^{\Omega}\}$ is the quadrature rule in Ω , $\{w^{\partial\Omega}, x^{\partial\Omega}\}$ is the quadrature rule on $\partial\Omega$, and $\{w^{\Gamma}, x^{\Gamma}\}$ is the quadrature rule on Γ .

The tolerance is set to $\text{tol} = 2 \times 10^{-1}$ in the first case and $\text{tol} = 1.0$ in the second

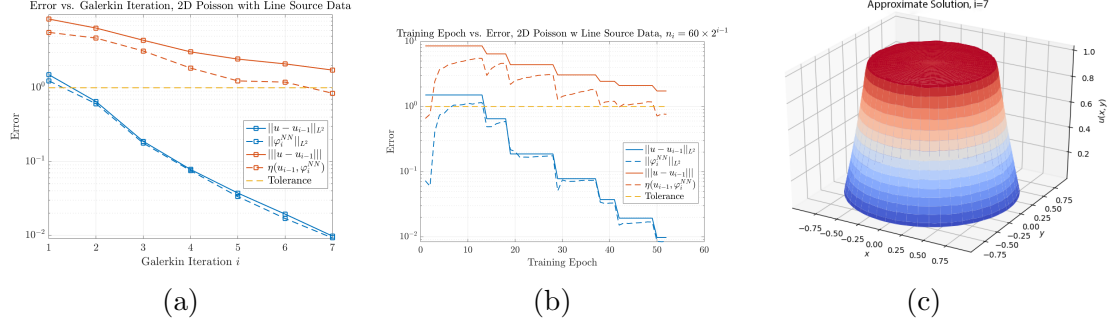


Figure 3.12: Poisson equation with line source data, $R_e = 1 - 1/\pi^2$ and $R_0 = 7/(6\sqrt{2})$. (a) true and estimated errors after each Galerkin iteration (b) true and estimated errors after each training epoch (c) approximate solution u_i for $i = 7$.

case, and $\varepsilon = 10^{-3}$. The network architecture for iteration i is as follows: $n_i = 30 \times 2^{i-1}$, $\sigma_i(t) = \tanh(\beta_i t)$, $\beta_i = i$. The learning rate for each basis function φ_i^{NN} is $\alpha_i = \frac{2 \times 10^{-2}}{1.1^{i-1}}$. The hidden parameters are initialized according to the box initialization detailed by Cyr, et al. [34]. We employ fixed tensor-product Gauss-Legendre quadrature rule with 128×128 nodes in order to approximate inner products in the interior of the domain. We approximate the inner products on $\partial\Omega$ and Γ using 512 equally-spaced nodes on the circles of radii R_e and R_0 , respectively.

The true and estimated errors after each refinement and epoch are provided in Figure 3.11 along with a plot of the neural network approximation for case 1. The analogous results for case 2 are provided in Figure 3.12. Interestingly, the cusp along $r = R_0$ is resolved well in both cases without any encoding of the line source into the training data, albeit with a lower rate of convergence as compared with the examples with smooth solutions.

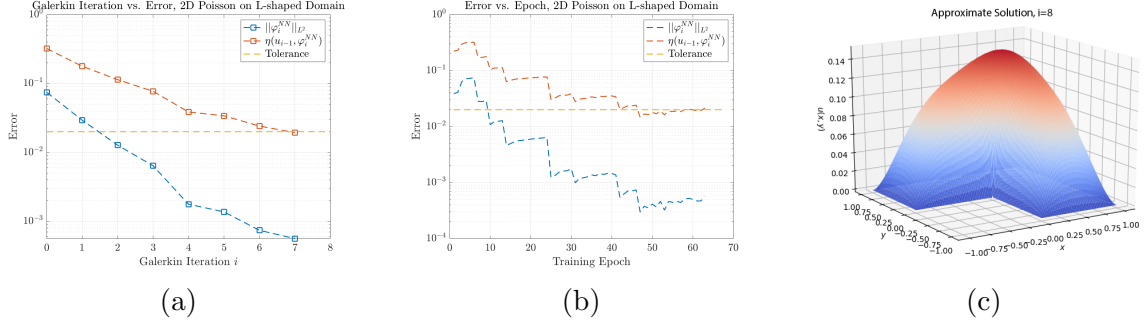


Figure 3.13: Poisson equation in L-shaped domain. Estimated errors after each Galerkin iteration (a) and epoch (b) as well as approximate solution u_i for $i = 8$ (c).

3.3.2.4 L-shaped Domain

In this example, we consider the Poisson equation on an L-shaped domain in which the solution contains a corner singularity:

$$\begin{cases} -\Delta u = 1, & \text{in } \Omega = (-1, 1)^2 \setminus (-1, 0]^2 \\ u + \varepsilon \partial_n u = 0, & \text{on } \partial\Omega. \end{cases}$$

The loss function is given by (3.23) with $f \equiv 1$ and is approximated using the same methodology as in 3.3.2.2.

The tolerance is set to $\text{tol} = 2 \times 10^{-2}$, and $\varepsilon = 10^{-4}$. The network architecture for iteration i is as follows: $n_i = 20 \times 2^{i-1}$, $\sigma_i(t) = \tanh(t)$. The learning rate for each basis function φ_i^{NN} is $\alpha_i = \frac{2 \times 10^{-2}}{1.1^{i-1}}$. The hidden parameters are again initialized according to the box initialization in $(-1, 1)^2$. We employ fixed Gauss-Legendre quadrature rules on each square subdomain of 128×128 nodes each in order to approximate inner products in the interior of the domain. A Gauss-Lobatto quadrature rule is employed on $[-1, 0] \times \{0\}$ and $\{0\} \times [-1, 0]$ to approximate the boundary containing the singularity, while Gauss-Legendre quadrature rules are employed on the remaining boundary edges.

Figure 3.13a shows the estimated errors $\|\varphi_i^{NN}\|_{L^2}$ and $\eta(u_{i-1}, \varphi_i^{NN})$ after each Galerkin iteration. Figure 3.13b shows the the estimated errors at each training iteration, while Figure 3.13c shows the approximate solution to the variational problem after eight Galerkin

iterations. As expected, without a priori knowledge of the location of the corner singularity, we observe a reduced rate of convergence as compared to prior examples with smooth solutions. Nevertheless, our algorithm is capable of resolving the corner singularity.

3.3.3 Fourth Order Problems

3.3.3.1 Displacement of a Beam

We consider the fourth-order ODE

$$\begin{cases} u^{(4)}(x) = f(x), & \text{in } \Omega = (0, 1) \\ u - \varepsilon_1 \partial_n(u'') = 0, & \text{on } \partial\Omega \\ u' + \varepsilon_2 \partial_n(u') = 0, & \text{on } \partial\Omega \end{cases}$$

with $f(x) = (2\pi)^4 \sin(2\pi x)$. The variational formulation is posed on the Hilbert space $X = H^2(\Omega)$ with corresponding bilinear operator $a(u, v) = (u'', v'')_\Omega + \varepsilon_1^{-1}(u(0)v(0) + u(1)v(1)) + \varepsilon_2^{-1}(u'(0)v'(0) + u'(1)v'(1))$ and data $L(v) = (f, v)_\Omega$. The exact solution is given by

$$u(x) = \sin(2\pi x) - 2\pi(2x - 1) \frac{4\varepsilon_1(\pi^2(6\varepsilon_2 - 2x^2 + 2x + 1) + 3) + x(x - 1)}{24\varepsilon_1 + 6\varepsilon_2 + 1}.$$

The corresponding loss function is approximated by

$$\frac{L_1 - L_2 - L_3 - L_4}{\sqrt{\sum_{i=1}^{n_G} w_i v''(x_i; \theta)^2 + \varepsilon_1^{-1}(v(0; \theta)^2 + v(1; \theta)^2) + \varepsilon_2^{-1}(v'(0; \theta)^2 + v'(1; \theta)^2)}}$$

where $L_1 = \sum_{i=1}^{n_G} w_i f(x_i) v(x_i; \theta)$, $L_2 = \sum_{i=1}^{n_G} w_i u_0''(x_i) v''(x_i; \theta)$, $L_3 = \varepsilon_1^{-1}[u_0(0) \cdot v(0; \theta) + u_0(1)v(1; \theta)]$, and $L_4 = \varepsilon_2^{-1}[u_0'(0)v'(0; \theta) + u_0'(1)v'(1; \theta)]$.

The tolerance is set to $\text{tol} = 3 \times 10^{-5}$, and $\varepsilon_1 = \varepsilon_2 = 10^{-4}$. The network architecture for iteration i is as follows: $n_i = 30 \times 2^{i-1}$, $\sigma_i(t) = \tanh(\beta_i t)$, $\beta_j = 1 + 3(i - 1)$. The learning

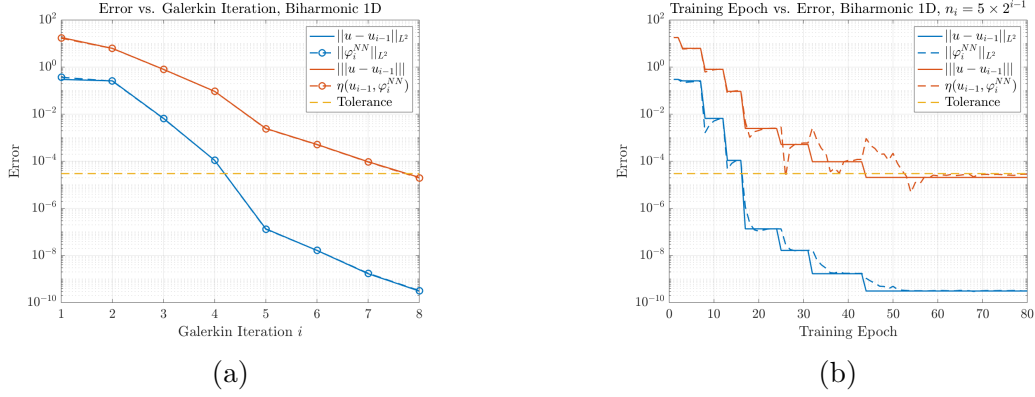


Figure 3.14: The biharmonic equation in 1D (displacement of a beam). True and estimated error in the L^2 and energy norm at the end of each Galerkin iteration (a) and epoch (b).

rate for each basis function φ_i^{NN} is $\alpha_i = \frac{2 \times 10^{-2}}{1.1^{i-1}}$. The hidden parameters are initialized as in Section 3.3.1. To evaluate the loss function and train the network, we employ a fixed Gauss-Legendre quadrature rule with 512 nodes to approximate all inner products. A separate Gauss-Legendre quadrature rule with 1000 nodes is used for validation, i.e. to compute the exact errors in the L^2 and energy norms.

Figure 3.14a shows the exact errors $\|u - u_{i-1}\|_{L^2}$ and $\|u - u_{i-1}\|$ at the end of each Galerkin iteration along with the estimated errors $\|\varphi_i^{NN}\|_{L^2}$ and $\eta(u_{i-1}, \varphi_i^{NN})$ for our algorithm. Figure 3.14b shows the true and estimated errors at each training epoch, while Figure 3.15 shows the exact error $u - u_{i-1}$ as well as the maximizer φ_i^{NN} at several stages of refinement. We observe no difficulties in approximating both the true error and the true solution.

3.3.3.2 Beam with Applied Couple

We consider the biharmonic equation with point moment source term:

$$\begin{cases} u^{(4)} = \delta'_{\{x=1/2\}}, & \text{in } \Omega = (0, 1) \\ u - \varepsilon_1 \partial_n(u'') = 0, & \text{on } \partial\Omega \\ u' + \varepsilon_2 \partial_n(u') = 0, & \text{on } \partial\Omega. \end{cases}$$

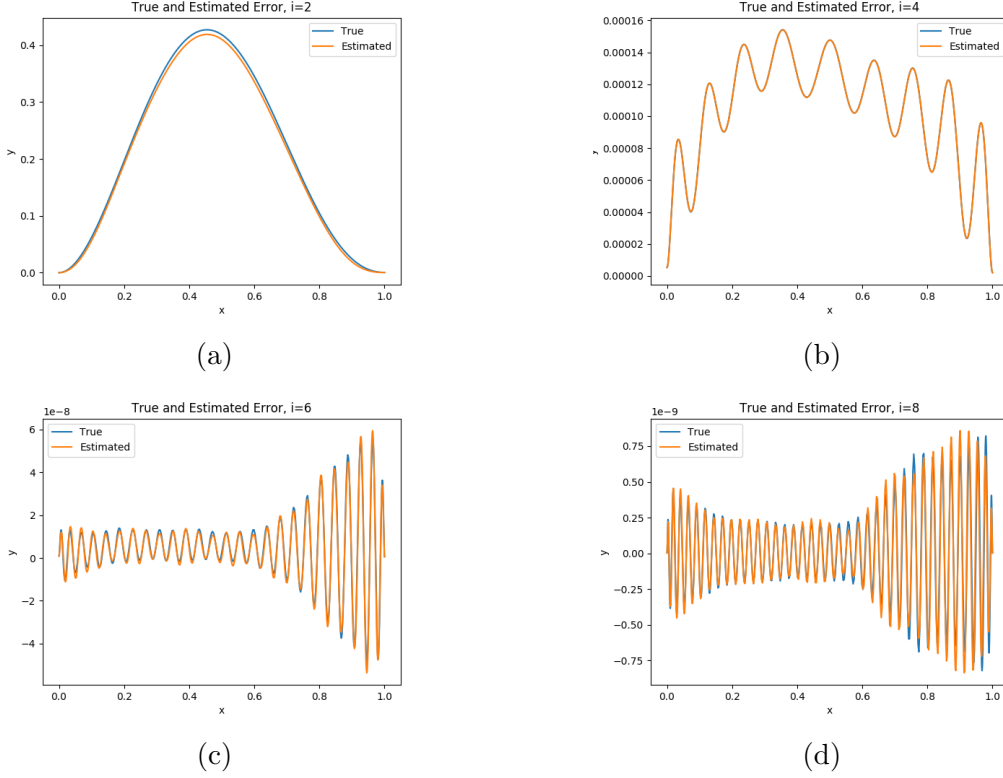


Figure 3.15: Displacement of a beam. Exact error $u - u_{i-1}$ and approximate error φ_i^{NN} for $i = 2, 4, 6, 8$.

The bilinear operator a is as defined in Section 3.3.3.1 and $L(v) = -v'(1/2)$. The exact variational solution corresponding to $\varepsilon_1 = \varepsilon_2 = 0$ is

$$u(x) = \frac{1}{4}(x - 1/2)|x - 1/2| - \frac{1}{4}x^3 + \frac{3}{8}x^2 - \frac{1}{4}x + \frac{1}{16}.$$

The corresponding loss function is approximated by

$$\frac{L_1 - L_2 - L_3 - L_4}{\sqrt{\sum_{i=1}^{n_G} w_i v''(x_i; \theta)^2 + \varepsilon_1^{-1}(v(0; \theta)^2 + v(1; \theta)^2) + \varepsilon_2^{-1}(v'(0; \theta)^2 + v'(1; \theta)^2)}}$$

where $L_1 = -v'(1/2; \theta)$, $L_2 = \sum_{i=1}^{n_G} w_i u_0''(x_i) v''(x_i; \theta)$, $L_3 = \varepsilon_1^{-1}[u_0(0)v(0; \theta) + u_0(1)v(1; \theta)]$, and $L_4 = \varepsilon_2^{-1}[u_0'(0)v'(0; \theta) + u_0'(1)v'(1; \theta)]$.

The tolerance is set to $\text{tol} = 4 \times 10^{-3}$, and $\varepsilon_1 = \varepsilon_2 = 10^{-5}$. The network architecture

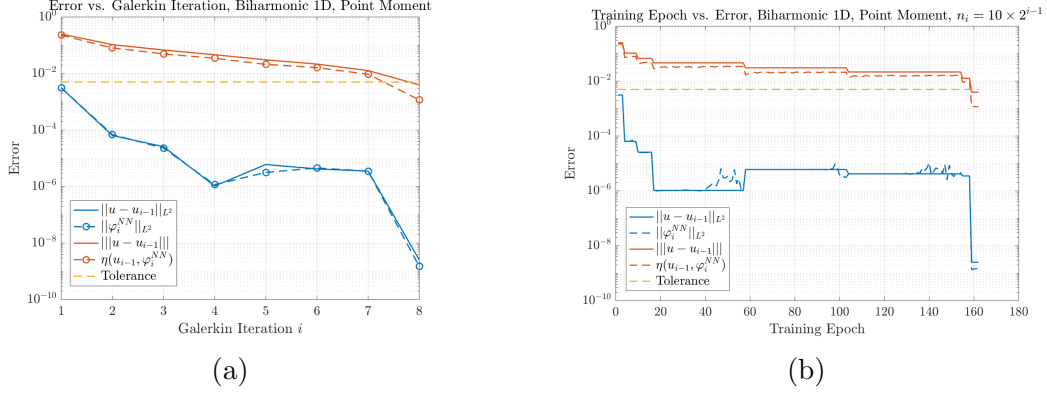


Figure 3.16: The biharmonic equation in 1D with point moment (beam with applied couple). True and estimated error in the L^2 and energy norm at the end of each Galerkin iteration (a) and epoch (b).

for iteration i is as follows: $n_i = 10 \times 2^{i-1}$, $\sigma_i(t) = \tanh(\beta_i t)$, $\beta_i = 1 + 3 \cdot 2^{i-1}$. The learning rate for each basis function φ_i^{NN} is $\alpha_i = \frac{1 \times 10^{-2}}{1.4^{i-1}}$. The hidden parameters are initialized as in Section 3.3.1. To evaluate the loss function and train the network, we employ a fixed Gauss-Legendre quadrature rule with 1024 nodes to approximate all inner products. A separate Gauss-Legendre quadrature rule with 1000 nodes is used for validation, i.e. to compute the exact errors in the L^2 and energy norms.

Figure 3.16a shows the true errors $\|u - u_{i-1}\|_{L^2}$ and $\|u - u_{i-1}\|$ at the end of each Galerkin iteration along with the estimated errors $\|\varphi_i^{NN}\|_{L^2}$ and $\eta(u_{i-1}, \varphi_i^{NN})$ for our algorithm. Figure 3.16b shows the true and estimated errors at each training epoch, while Figure 3.17 shows the exact error $u - u_{i-1}$ as well as the maximizer φ_i^{NN} at several stages of refinement along with both the numerical solution u_i and its second derivative u_i'' . We observe that initially, low frequency oscillations in the error are learned followed later by suppression of the high frequency error components which are particularly notable near the discontinuity in the second derivative of u_i .

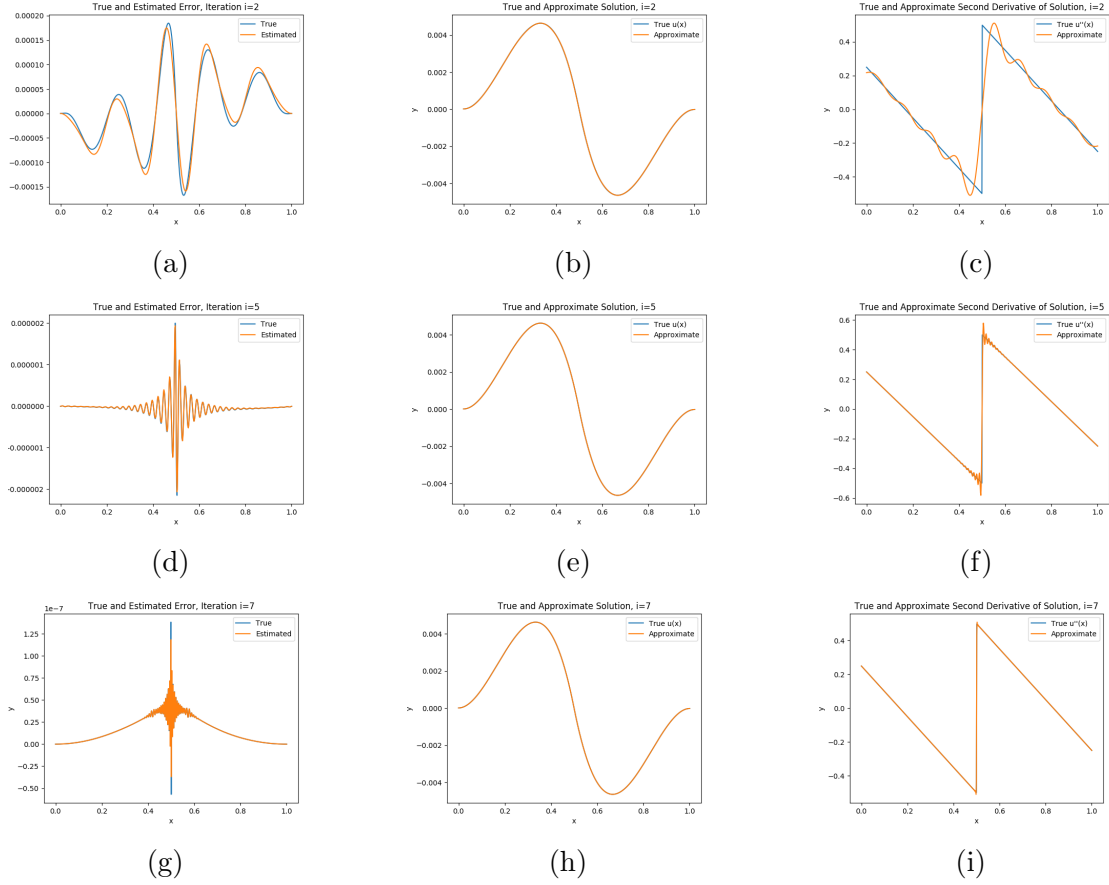


Figure 3.17: Biharmonic equation with point moment in 1D (beam with applied couple). Exact error $u - u_{i-1}$ and approximate error φ_i^{NN} as well as numerical solution u_i and its derivative u_i'' for $i = 2$ (top row), $i = 5$ (middle row), and $i = 7$ (bottom row).

3.3.3.3 Circular Plate with Point Load

We consider the two-dimensional biharmonic equation with point load source term in the unit disk:

$$\begin{cases} \Delta^2 u = \delta_{\{(0,0)\}} & \text{in } \Omega = \{(x, y) : x^2 + y^2 < 1\} \\ u - \varepsilon_1 \partial_n (\Delta u) = 0 & \text{on } \partial\Omega \\ \Delta u + \varepsilon_2 \partial_n u = 0 & \text{on } \partial\Omega. \end{cases}$$

The variational formulation is again posed on $X = H^2(\Omega)$ with bilinear operator $a(u, v) = (\Delta u, \Delta v)_\Omega + \varepsilon_1^{-1}(u, v)_{\partial\Omega} + \varepsilon_2(\partial_n u, \partial_n v)_{\partial\Omega}$ and data $L(v) = v(0, 0)$. The exact variational

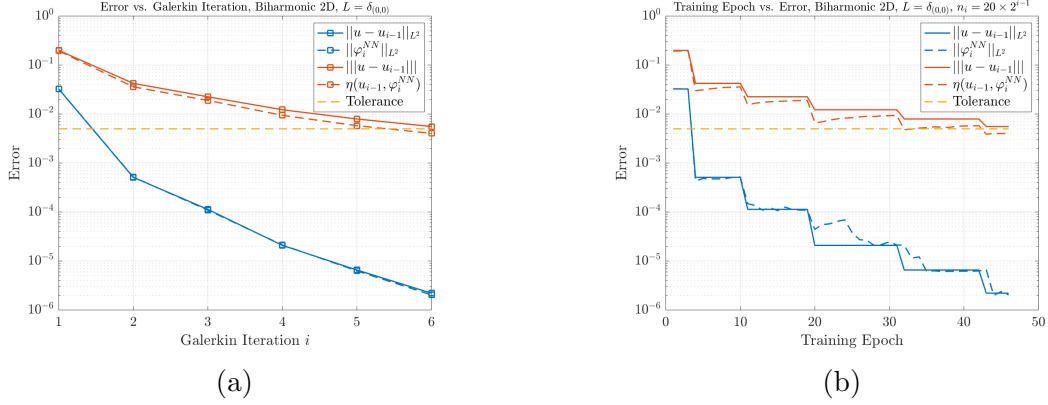


Figure 3.18: The biharmonic equation in 2D. True and estimated error in the L^2 and energy norm at the end of each Galerkin iteration (a) and epoch (b).

solution is given by

$$u(r, \theta) = \frac{r^2}{8\pi} \log(r) + c_1 r^2 + c_2, \quad c_1 = \frac{1}{4 + 2\varepsilon_2} \cdot \left(-\frac{\varepsilon_2}{2\pi} - \frac{1}{8\pi}\right), \quad c_2 = -c_1 + \frac{\varepsilon_1}{2\pi}.$$

The corresponding loss function is approximated by

$$\frac{L_1 - L_2 - L_3 - L_4}{\sqrt{\sum_{i=1}^{n_\Omega} w_i |\Delta v(x_i^\Omega; \theta)|^2 + \varepsilon^{-1} \sum_{i=1}^{n_{\partial\Omega}} w_i^{\partial\Omega} v(x_i^{\partial\Omega}; \theta)^2 + \varepsilon_2 \sum_{i=1}^{n_{\partial\Omega}} w_i^{\partial\Omega} \partial_n v(x_i^{\partial\Omega}; \theta)^2}}$$

where $L_1 = v(0; \theta)$, $L_2 = \sum_{i=1}^{n_\Omega} w_i^\Omega \Delta u_0(x_i^\Omega) \Delta v(x_i^\Omega; \theta)$, $L_3 = \varepsilon_1^{-1} \sum_{i=1}^{n_{\partial\Omega}} w_i^{\partial\Omega} u_0(x_i^{\partial\Omega}) \cdot v(x_i^{\partial\Omega}; \theta)$, and $L_4 = \varepsilon_2 \sum_{i=1}^{n_{\partial\Omega}} w_i^{\partial\Omega} \partial_n u_0(x_i^{\partial\Omega}) \partial_n v(x_i^{\partial\Omega}; \theta)$. Here, $\{w^\Omega, x^\Omega\}$ is the quadrature rule in Ω while $\{w^{\partial\Omega}, x^{\partial\Omega}\}$ is the quadrature rule on $\partial\Omega$.

3.4 Conclusions

We have introduced a framework for adaptively generating basis functions for a finite-dimensional subspace which may be used in a standard Galerkin method for approximating variational problems. We have demonstrated that each basis function φ_i^{NN} may be expressed as the realization of a neural network $V_{n_i, C_i}^{\sigma_i}$ with a single hidden layer by learning the dual representation of the weak residual $r(u_{i-1}) : X \rightarrow \mathbb{R}$. Furthermore,

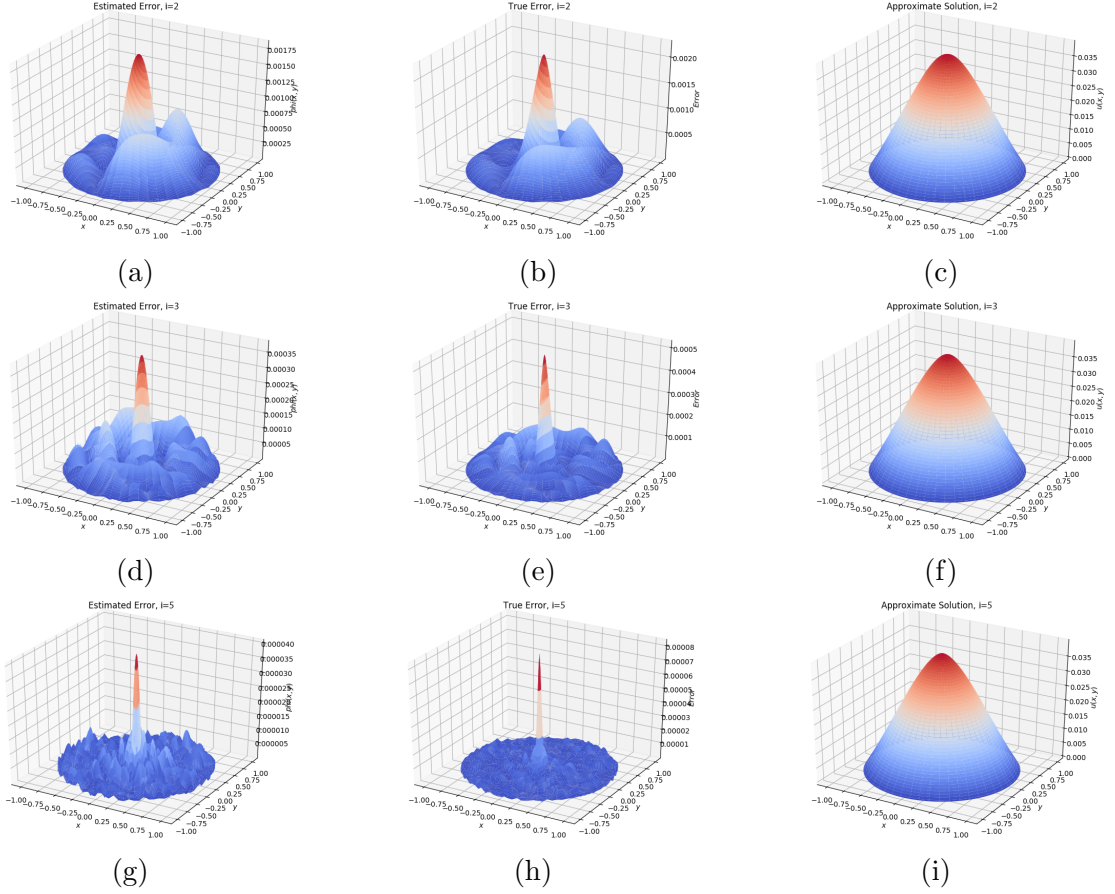


Figure 3.19: Biharmonic equation in 2D with point load source. True error $u - u_{i-1}$ (middle column) and estimated error φ_i^{NN} (left column) as well as numerical solution u_i (right column) for $i = 2, 3, 5$.

we have shown that $\eta(u_{i-1}, \varphi_i^{NN})$ is an equivalent measure of the error $\|u - u_{i-1}\|$ and constitutes a robust a posteriori error estimate. We emphasize that our algorithm may be applied seamlessly to the broad class of problems in this section with no major modifications. This is in contrast to e.g. the finite element method which requires a tailored approach depending on the underlying Sobolev space and spatial dimension of the variational problem.

3.5 Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1644760 as well as PhILMS grant DE-

SC0019453.

CHAPTER 4

Application to Reissner-Mindlin plates

4.1 Introduction

Neural networks offer an interesting alternative to traditional numerical methods for partial differential equations (PDEs) such as finite elements, finite differences, and finite volumes, and have been used to approximate various linear and nonlinear elliptic, parabolic, and hyperbolic PDEs [18, 69, 105, 106, 115, 130]. We have seen in the previous chapter that Galerkin neural networks, in contrast to many other works in the literature thus far – even those that are based on variational principles – allow one to leverage the approximation properties of Galerkin methods while still enjoying the rich nonlinear expressivity of neural networks. The numerical examples considered thus far demonstrate that Galerkin neural networks can be applied broadly to many different classes of problems without any significant modification (cf. finite element methods which must be modified significantly when moving from the H^1 formulations in Section 3.3.2 to the H^2 formulations in Section 3.3.3). Moreover, the variational approach taken with Galerkin neural networks allows for straightforward incorporation of singular data such as point and line sources. However,

the examples shown in Chapter 3 have been relatively simple and can be approximated using any number of traditional techniques to high orders of accuracy.

All of this begs the question of what advantages (if any) are offered by neural networks over traditional numerical methods? Traditional methods have the benefit of being refined over many decades and are often capable of delivering high-fidelity approximations efficiently while the potential benefits of a neural network-based approach for such problems are unclear. It is nevertheless the case that there remain many classes of problem that pose difficulties for traditional numerical methods. Examples include parameter-dependent problems that arise in linear elasticity and plate theory which exhibit locking [32]. Locking means that the numerical approximation deteriorates for small parameter values even though the solution itself may not be sensitive to this parameter. Additionally, in some models such as the Reissner-Mindlin plate model, a characteristic feature of solutions is the presence of boundary layers. In conjunction with the possibility of locking, the robust and accurate approximation of such problems still poses a serious challenge, which has led to the development of a whole gamut of sophisticated techniques in an attempt to obtain schemes capable of delivering robust and high-resolution approximations [6, 12, 25, 28, 31, 44, 66, 117, 118].

In this work, we consider the question of whether a neural network approach is capable of approximating Reissner-Mindlin plates uniformly in the plate thickness while also resolving multiscale features such as boundary layers. The universal approximation property of neural networks should, in theory, mean that neural networks have the capability to resolve multiscale features. While results exist quantifying how the accuracy of a neural network approximation varies with respect to the width and depth of the network [39, 45, 63, 99, 113] for certain classes of functions, the capability of neural networks to uniformly approximate parameter-dependent functions exhibiting boundary layers remains open. The relationship between ReLU networks and piecewise linear approximations means that one can expect univariate ReLU networks to be capable of

delivering uniformly accurate approximations of functions exhibiting boundary layers. The results in Figure 4.1 confirm this expectation. While such a result shows that neural networks are capable of delivering uniformly accurate approximations for direct approximation of functions exhibiting boundary layers in the univariate case, it remains to be seen whether similar results can be achieved in higher dimensions and when the function is implicitly defined to be the solution of a singularly perturbed system of elliptic PDEs.

Accordingly, we investigate the neural network approximation of Reissner-Mindlin plates. As alluded to earlier, the nonlinear nature of neural networks should mean that they are naturally adaptive to multiscale features without the need for specialized grids or elements. In addition, it is easy to increase the smoothness of the neural network approximation by choosing appropriate activation functions, which gives the flexibility to consider natural variational formulations posed on smoother Sobolev spaces than might otherwise be practical were standard finite elements to be used. We introduce and analyze several new, least squares-type, variational formulations for the Reissner-Mindlin plate and use them in conjunction with the Galerkin Neural Network approach developed in

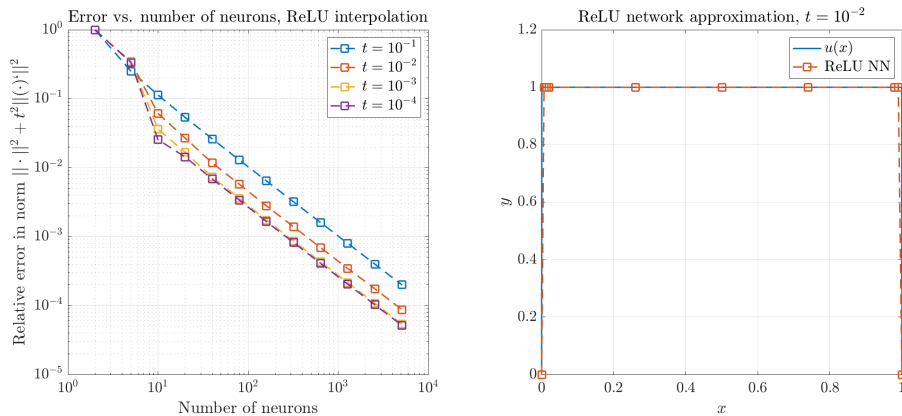


Figure 4.1: Left: Numerical approximation rate of the function $u(x) = (1 - e^{(x-1)/t} - e^{-x/t} + e^{-1/t}) / (1 + e^{-1/t})$ for various t using a ReLU network with one hidden layer. Right: The function $u(x)$ alongside the ReLU approximation. The hidden weights are set to 1, the biases are graded to account for the boundary layers, and the activation coefficients are chosen so that the network interpolates the function u at the points where the ReLU basis functions activate.

Chapter 3 [3]. The relative performance of the various formulations and, in particular, whether they exhibit locking, together with their ability to resolve multiscale features is illustrated for a class of new benchmark (closed form) solutions of the Reissner-Mindlin problem which exhibits boundary layers.

The rest of this work is structured as follows. In Section 4.2, we briefly summarize the relevant properties of the Galerkin Neural Network framework and its underlying theory which will be used to approximate the model problem. In Section 4.3, we introduce a new realistic benchmark problem which exhibits multiscale features and also consider the capacity of neural networks to approximate such features. In Section 4.4, we apply the Galerkin Neural Network method to several new variational formulations for the Reissner-Mindlin plate and demonstrate the robustness of our approach with respect to the plate thickness. Conclusions follow in Section 4.5.

4.2 Review of the Galerkin Neural Network Framework

In this section, we briefly summarize the main features of the Galerkin Neural Network approach developed in Chapter 3 [3] that we will later use. Let

$$V_n^\sigma := \{v : v(x) = \sum_{i=1}^n c_i \sigma(x \cdot W_i + b_i), \ b_i, c_i \in \mathbb{R}, \ W_i \in \mathbb{R}^d, \ x \in \Omega\} \quad (4.1)$$

be the set of all functions which are the realizations of a feedforward neural network consisting of a single hidden layer of n neurons and nonlinear, continuous activation function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$. Here, W and b are the weights and biases, respectively, of the hidden layer, and c are the activation coefficients of the network. Additionally, we let $V_{n,C}^\sigma$ be the

subset consisting of realizations with bounded parameters:

$$V_{n,C}^\sigma := \{v \in V_n^\sigma : \|(W, b, c)\| < C\}, \quad (4.2)$$

where $\|(W, b, c)\| := \max_{ij} |W_{ij}| + \max_i |b_i| + \max_i |c_i|$.

We will consider the basic setting of the following variational problem:

$$u \in X : a(u, v) = L(v) \quad \forall v \in X, \quad (4.3)$$

where $X \subset H^s(\Omega)$, $L(\cdot)$ is a bounded linear operator on $H^s(\Omega)$, and $a(\cdot, \cdot)$ is a symmetric, positive-definite bilinear operator on $H^s(\Omega)$ which is continuous and coercive with respect to $H^s(\Omega)$, i.e., there exist constants $M, \alpha > 0$ such that $|a(u, v)| \leq M \|u\|_{H^s(\Omega)} \|v\|_{H^s(\Omega)}$ and $\alpha \|v\|_{H^s(\Omega)}^2 \leq a(v, v)$ for all $u, v \in X$. The bilinear form $a(\cdot, \cdot)$ induces a norm denoted by $\|\cdot\|_a := \sqrt{a(\cdot, \cdot)}$.

In Chapter 3 [3], we used neural networks to iteratively construct a sequence of basis functions $\{\varphi_i^{NN} \in V_{n_i}^{\sigma_i}, i \in \mathbb{N}\}$ that were used to define a Galerkin scheme for (4.3) based on an initial approximation $u_0 \in X$ and the functions φ_i^{NN} as follows:

$$u_i \in S_i := \text{span}\{u_0, \varphi_1^{NN}, \dots, \varphi_i^{NN}\} : a(u_i, v) = L(v) \quad \forall v \in S_i. \quad (4.4)$$

Céa's Lemma [27] provides the following error estimate for u_i :

$$\|u - u_i\|_a \leq \sqrt{\frac{M}{\alpha}} \|u - v\|_a \quad \forall v \in S_i, \quad (4.5)$$

which means that the approximation u_i is, up to a constant, the best possible approximation from the subspace S_i .

The basis functions φ_i^{NN} are constructed iteratively as follows. Given u_{i-1} , $i \geq 1$, the

weak residual $r(u_{i-1}) : X \rightarrow \mathbb{R}$ is defined by

$$\langle r(u_{i-1}), v \rangle = L(v) - a(u_{i-1}, v), \quad v \in X, \quad (4.6)$$

where $\langle r(u_{i-1}), v \rangle$ denotes the duality pairing on X . Using the weak residual as the loss function, the i th basis function φ_i^{NN} is constructed by training a neural network to find the maximizer as follows:

$$\varphi_i^{NN} \in V_{n_i, C_i}^\sigma : \langle r(u_{i-1}), \varphi_i^{NN} \rangle = \max_{v \in V_{n_i, C_i}^\sigma \cap B} \langle r(u_{i-1}), v \rangle, \quad (4.7)$$

where B is the closed unit ball in X . Any standard training procedure may be used to solve (4.7); we use the approach given in Algorithm 2 from Chapter 3.

The residual may be rewritten using (4.3) as

$$\langle r(u_{i-1}), v \rangle = a(u, v) - a(u_{i-1}, v) = a(u - u_{i-1}, v), \quad v \in X$$

which, thanks to the Cauchy-Schwarz inequality, is maximized when $v \propto u - u_{i-1}$. This means that the basis function $\varphi_i^{NN} \in V_{n_i, C_i}^\sigma \cap B$ is an approximation to the normalized error $(u - u_{i-1}) / \|u - u_{i-1}\|$. Consequently, we may view $\{\varphi_i^{NN}\}$ as a sequence of increasingly fine scale corrections to the initial approximation u_0 .

The basic properties of the Galerkin Neural Network scheme are summarized in the following theorem [3].

Theorem 4.2.1. *For $i \geq 1$, let $\tau_i \in (0, 1)$ and u_i be defined as in (4.4). Then there exist network widths $n_i = n(\tau_i, u_{i-1})$ and bounds $C_i = C(\tau_i, u_{i-1})$ depending on τ_i and u_{i-1} such that*

$$\|u - u_i\|_a \leq \|u - u_0\|_a \cdot \prod_{j=1}^i 2\tau_j / (1 - \tau_j) \quad (4.8)$$

Moreover, if $0 < \tau_i < 1/3$ then

$$\frac{1 - \tau_i}{1 + \tau_i} \eta_i \leq \|u - u_{i-1}\|_a \leq \frac{1 - \tau_i}{1 - 3\tau_i} \eta_i \quad (4.9)$$

where $\eta_i := \langle r(u_{i-1}), \varphi_i^{NN} \rangle$.

The estimate (4.9) shows that the (fully computable) quantity η_i provides an a posteriori error estimator [5, 125] for the true error $\|u - u_{i-1}\|_a$ which can be used as a stopping criterion (as in Algorithm 1).

4.3 Boundary Layer Resolution using Neural Networks

Physical problems involving small (or large) parameters often exhibit localized features on a length scale defined by the parameter. One such example is the Reissner-Mindlin plate model which describes the bending of a thin plate while taking into account shear deformation. Let $\boldsymbol{\beta}$ be the rotation of the fibers normal to the midplane of the plate and let ω be the transverse displacement of the midplane itself. The Reissner-Mindlin model takes the form of a system of elliptic PDEs:

$$\begin{cases} -\Delta \boldsymbol{\beta} + t^{-2}(\boldsymbol{\beta} - \nabla \omega) = 0 & \text{in } \Omega \\ t^{-2} \operatorname{div} (\boldsymbol{\beta} - \nabla \omega) = g, & \text{in } \Omega \\ \boldsymbol{\beta} = \mathbf{0}, \omega = 0 & \text{on } \partial\Omega. \end{cases} \quad (4.10)$$

Here, $\Omega \subset \mathbb{R}^2$ is the midplane of the plate, $t > 0$ is the plate thickness, g is the applied transverse load, and, for ease of exposition, we use a simplified stress-displacement law which nevertheless retains the essential character of the Reissner-Mindlin model. The boundary conditions correspond to a hard, simple support [11], which means that, in

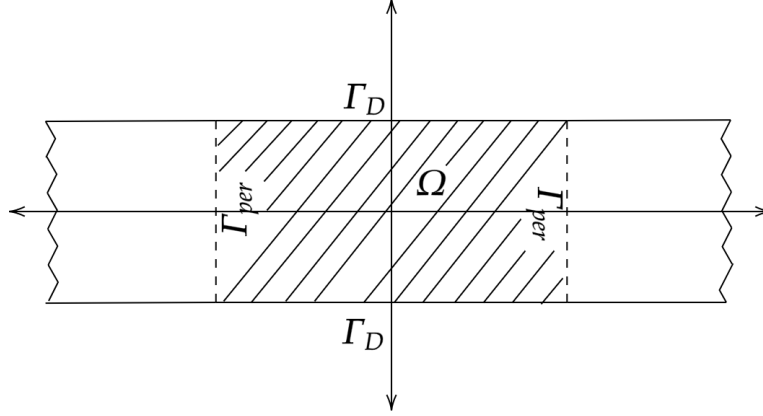


Figure 4.2: Domain for the test problem in Section 4.3.

the limit where the plate thickness t tends to 0, the Reissner-Mindlin model gives rise to boundary layers which manifest most strongly in the shear stress [13] defined by

$$\boldsymbol{\sigma} := t^{-2}(\boldsymbol{\beta} - \nabla\omega). \quad (4.11)$$

4.3.1 Benchmark Solution of Reissner-Mindlin Plate Exhibiting Boundary Layers

In order to illustrate the above points more concretely, we derive a class of closed form solutions of the Reissner-Mindlin plate model which will later be used to benchmark the performance of numerical schemes. Consider an infinite strip along the x -axis subject to a univariate 2π -periodic transverse load g . Due to periodicity, it suffices to consider a single period $\Omega = (-\pi, \pi) \times (-1, 1)$ as shown in Figure 4.2. We partition the boundary of Ω into disjoint sets $\Gamma_D = (-\pi, \pi) \times \{-1\} \cup \{1\}$ and $\Gamma_{\text{sym}} = \{-\pi\} \cup \{\pi\} \times (-1, 1)$ with homogeneous Dirichlet boundary conditions applied on Γ_D and periodic boundary conditions applied on Γ_{per} . The periodic transverse load $g(x)$ is written as a Fourier series given by

$$g(x) = \frac{1}{2}g_0 + \sum_{n=1}^{\infty} g_n \cos(nx) + \sum_{n=1}^{\infty} \tilde{g}_n \sin(nx).$$

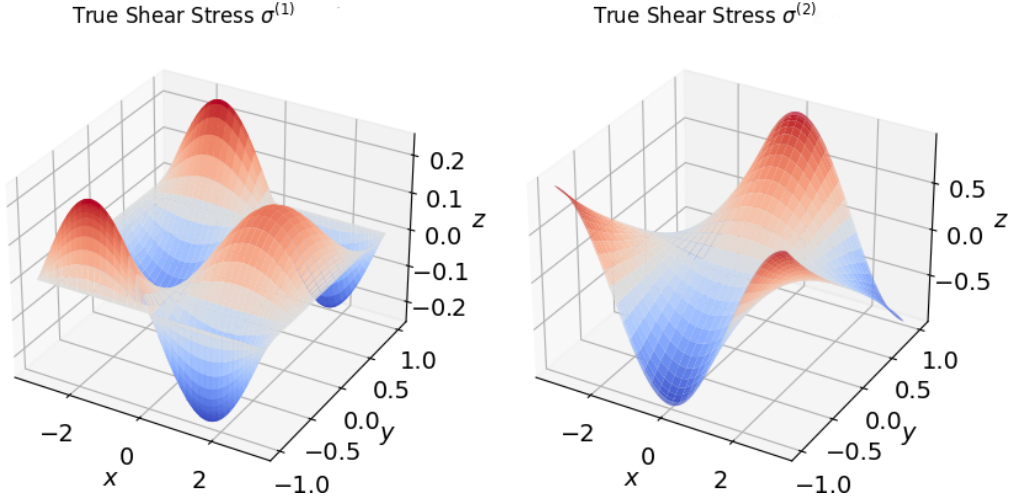


Figure 4.3: Left: True x -component of shear stress with $n = 1$ and $t = 10^{-2}$. Right: True y -component of the shear stress with $n = 1$ and $t = 10^{-2}$.

For $n \in \mathbb{N}_0$, let (β_n, ω_n) be given by

$$\begin{aligned} \beta_n(x, y) &= \begin{bmatrix} -(\Phi'_n(y, t) + n \cdot \Psi_n(y, t) - n \cdot \Upsilon_n(y, t)) \sin(nx) \\ (n \cdot \Phi_n(y, t) + \Psi'_n(y, t) - \Upsilon'_n(y, t)) \cos(nx) \end{bmatrix} \\ \omega_n(x, y) &= [(1 + t^2)\Psi_n(y, t) - \Upsilon_n(y, t)] \cos(nx), \end{aligned} \quad (4.12)$$

where Φ_n , Ψ_n , and Υ_n are given by

$$\begin{aligned} \Phi_n(y, t) &= \frac{A_n(t)t \sinh(\lambda_n y)}{\sinh(\lambda_n)}, \quad \Psi_n(y, t) = \begin{cases} -y^2/2 + D_0(t), & n = 0 \\ n^{-2} - D_n(t) \cdot \frac{\cosh(ny)}{\cosh(n)}, & n > 0 \end{cases} \\ \Upsilon_n(y, t) &= \begin{cases} -y^2/2 - y^4/24 + B_0(t)y^2, & n = 0 \\ \frac{B_n(t)y \cdot \sinh(ny)}{\sinh n} - (n^{-4} - n^{-2}) + \frac{C_n(t) \cdot \cosh(ny)}{\cosh n}, & n > 0, \end{cases} \end{aligned} \quad (4.13)$$

with $A_n(t)$, $B_n(t)$, $C_n(t)$, and $D_n(t)$ coefficients depending on n and t , and $\lambda_n := (n^2 + 1/t^2)^{1/2}$. The values of $A_n(t)$, $B_n(t)$, $C_n(t)$, and $D_n(t)$ are determined by the boundary

conditions and have series expansions valid for small t given by

$$\begin{aligned}
A_n(t) &= \left(\frac{2}{n} - 4\xi_n \right) t^2 + \mathcal{O}(t^3) \\
B_n(t) &= \xi_n \frac{\tanh(n)}{n} \left[-\frac{1}{n^2} + 2(-1 + 2\xi_n)t^2 + \mathcal{O}(t^3) \right] \\
C_n(t) &= \frac{\xi_n}{n^2} \left[\frac{n \coth(n) - 2n^2 + 1}{n^2} + (2(\coth(n) - 2n^2 + 1) - 2\xi_n \tanh(n)) t^2 + \mathcal{O}(t^3) \right] \\
D_n(t) &= \xi_n \left[\frac{\tanh(n)}{n^2} + 4(1 - 2\xi_n)t^2 + \mathcal{O}(t^3) \right],
\end{aligned} \tag{4.14}$$

where $\xi_n = \sinh(2n)/(2n + \sinh(2n))$. Full details leading to (4.14) will be found in the Appendix.

Straightforward manipulation reveals that the functions defined in (4.12) satisfy

$$\begin{cases} -\Delta \boldsymbol{\beta}_n + t^{-2}(\boldsymbol{\beta}_n - \nabla \omega_n) = 0 & \text{in } \Omega \\ t^{-2} \operatorname{div} (\boldsymbol{\beta}_n - \nabla \omega_n) = \cos(nx), & \text{in } \Omega \\ \boldsymbol{\beta}_n = \mathbf{0}, \omega_n = 0 & \text{on } \Gamma_D \\ \boldsymbol{\beta}_n, \omega_n \text{ periodic} & \text{on } \Gamma_{\text{per}}. \end{cases} \tag{4.15}$$

In turn, thanks to linearity and the translation identity $\sin(nx) = \cos(nx - \pi/2)$, the solution to (4.10) is given by

$$\begin{aligned}
\boldsymbol{\beta}(x, y) &= \frac{1}{2} g_0 \boldsymbol{\beta}_0(x, y) + \sum_{n=1}^{\infty} g_n \boldsymbol{\beta}_n(x, y) + \sum_{n=1}^{\infty} \tilde{g}_n \boldsymbol{\beta}_n(x - \frac{\pi}{2n}, y) \\
\omega(x, y) &= \frac{1}{2} g_0 \omega_0(x, y) + \sum_{n=1}^{\infty} g_n \omega_n(x, y) + \sum_{n=1}^{\infty} \tilde{g}_n \omega_n(x - \frac{\pi}{2n}, y).
\end{aligned}$$

This solution was given in [6] in the particular case when $n = 1$; here, we generalize to any $n \in \mathbb{N}_0$. The shear stress $\boldsymbol{\sigma}_n = t^{-2}(\boldsymbol{\beta}_n - \nabla \omega_n)$ associated with the n th Fourier mode

is given by

$$\boldsymbol{\sigma}_n(x, y) = \begin{bmatrix} [-t^{-2}\Phi'_n(y, t) + n \cdot \Psi_n(y, t)] \sin(nx) \\ [nt^{-2} \cdot \Phi_n(y, t) - \Psi'_n(y, t)] \cos(nx) \end{bmatrix}, \quad (4.16)$$

which, for $t \ll 1$, exhibits a boundary layer on the upper and lower edges of the plate:

$$\boldsymbol{\sigma}_n(x, y) \sim \frac{A_n(t)}{t^2 \sinh \lambda_n} \begin{bmatrix} -\lambda_n \cosh(\lambda_n y) \sin(nx) \\ n \sinh(\lambda_n y) \cos(nx) \end{bmatrix},$$

or, thanks to (4.14):

$$\boldsymbol{\sigma}_n(x, y) \sim \left(\frac{1}{n} - \frac{2 \sinh(2n)}{2n + \sinh(2n)} + \mathcal{O}(t^3) \right) \frac{2}{\sinh(\lambda_n)} \begin{bmatrix} -\lambda_n \cosh(\lambda_n y) \sin(nx) \\ n \sinh(\lambda_n y) \cos(nx) \end{bmatrix}.$$

Figure 4.3 shows the shear stress in the case $n = 1$ and $t = 10^{-2}$. The factors $\cosh(\lambda_n y)$ and $\sinh(\lambda_n y)$ each give rise to boundary layers in both components of the shear stress, which is more pronounced in the x -component of $\boldsymbol{\sigma}_n$ due to the presence of the factor $\lambda_n \gg n$ when $t \ll 1$.

4.3.2 Function Fitting of Reissner-Mindlin Shear Stress

We are interested in the approximability of the boundary layer in the shear stress $\boldsymbol{\sigma}$ using neural networks. In order to isolate the issue of approximability, we utilize Algorithm 1 in order to “learn” the x -component of the shear stress, $\boldsymbol{\sigma}^{(1)}$, with $n = 1$. The bilinear operator is given by $a(u, v) = (u, v)_\Omega + (\nabla u, \nabla v)_\Omega + \varepsilon^{-1}(u, v)_{\partial\Omega}$ and the data is given by $L(v) = (\boldsymbol{\sigma}^{(1)}, v)_\Omega + (\nabla \boldsymbol{\sigma}^{(1)}, \nabla v)_\Omega$. The training data consists of 128×128 Gauss-Legendre quadrature points. The width of the network for each basis function is $n_i = 20 \cdot 2^{i-1}$ while the activation function for each basis function is $\sigma_i(z) = \tanh((1 + 0.25i)z)$. The hyperplanes of the network for each basis function are initialized so that they are parallel

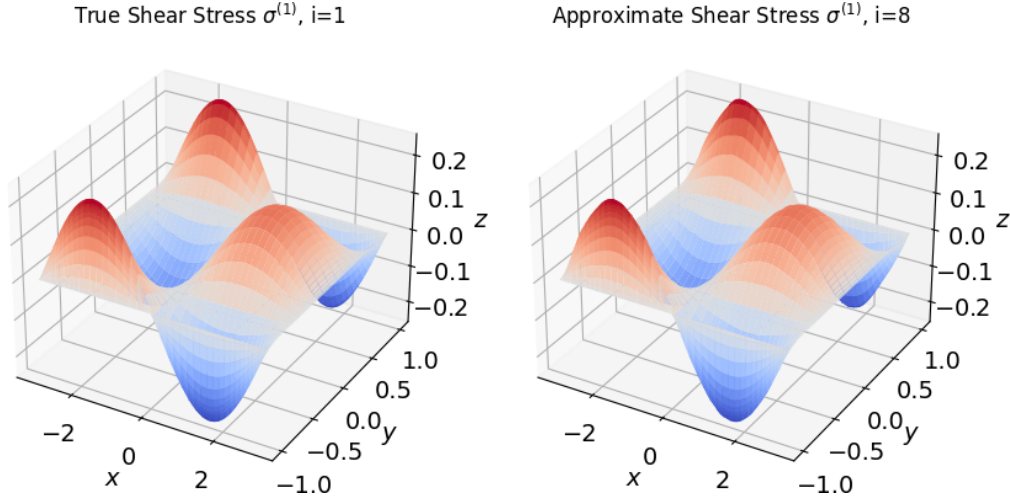


Figure 4.4: Left: True x -component of shear stress with $n = 1$ and $t = 10^{-2}$. Right: Neural network approximation of x -component of shear stress with $n = 1$ and $t = 10^{-2}$.

to the x -axis, the y -axis, $y = x$, or $y = -x$ (see [3], §3.2.2).

Figure 4.4 shows the neural network approximation to the shear stress, from which it is apparent that the sequence of networks is capable of approximating the boundary layer. These results demonstrate that the Galerkin Neural Network procedure is capable of resolving boundary layers without issue when applied to a simple function fitting of the shear stress σ_n .

4.4 Variational Formulations of the Reissner-Mindlin Model

Encouraged by the results of Section 4.3 showing the capability of neural networks to approximate the shear stress, we now turn to the question of which choice of variational formulation for the Reissner-Mindlin plate model to use in conjunction with neural networks. Ideally, we seek a variational formulation for which the associated bilinear form is continuous and coercive with constants M and α (see [27]) that are independent of t which, thanks to Céa's Lemma, means that by establishing control of the ratio M/α , we have a quasi-optimal solution. To this end, we present and analyze several variational

formulations and provide numerical results demonstrating their efficacy or lack thereof.

In all numerical examples that follow, unless otherwise stated, the neural network architecture for the i th Galerkin Neural Network basis function consists of a single hidden layer of width $n_i = 20 \cdot 2^{i-1}$. The activation function for the i th basis function is $\sigma_i(z) = \tanh((1 + 0.25i)z)$. The weights and biases are initialized so that the hyperplanes $x \cdot W_j + b_j$ of the hidden layer are either parallel to the x -axis, y -axis, $y = x$, or $y = -x$ as described in [3]. The training data consists of 128×128 Gauss-Legendre quadrature nodes, while the validation data used to compute the loss function and true errors where such a computation is possible consists of 150×150 Gauss-Legendre quadrature nodes.

4.4.1 Natural Variational Formulation

The natural variational formulation of (4.10) is to seek $(\boldsymbol{\beta}, \omega) \in X := \mathbf{H}_0^1(\Omega) \times H_0^1(\Omega)$, where X is equipped with the norm $\|(\boldsymbol{\beta}, \omega)\|_X := (\|\boldsymbol{\beta}\|_{H^1(\Omega)}^2 + \|\omega\|_{H^1(\Omega)}^2)^{1/2}$, such that

$$\mathfrak{B}_0((\boldsymbol{\beta}, \omega); (\boldsymbol{\varphi}, v)) := (\nabla \boldsymbol{\beta}, \nabla \boldsymbol{\varphi})_\Omega + t^{-2}(\boldsymbol{\beta} - \nabla \omega, \boldsymbol{\varphi} - \nabla v)_\Omega = (g, v) =: \mathfrak{L}_0(\boldsymbol{\varphi}, v) \quad (4.17)$$

for all $(\boldsymbol{\varphi}, v) \in X$. The operator $\mathfrak{B}_0$ is symmetric and positive-definite as well as continuous and coercive according to the following result:

Proposition 4.4.1. *Let $(\boldsymbol{\beta}, \omega) \in X$ and $0 < t \leq 1$. There exist constants $C_1 > 0$ and $C_2 > 0$ such that*

$$\begin{aligned} \mathfrak{B}_0((\boldsymbol{\beta}, \omega); (\boldsymbol{\varphi}, v)) &\leq C_1 t^{-2} \|(\boldsymbol{\beta}, \omega)\|_X \cdot \|(\boldsymbol{\varphi}, v)\|_X \quad \forall (\boldsymbol{\beta}, \omega), (\boldsymbol{\varphi}, v) \in X \\ C_2 \|(\boldsymbol{\beta}, \omega)\|_X^2 &\leq \mathfrak{B}_0((\boldsymbol{\beta}, \omega); (\boldsymbol{\beta}, \omega)) \quad \forall (\boldsymbol{\beta}, \omega) \in X. \end{aligned}$$

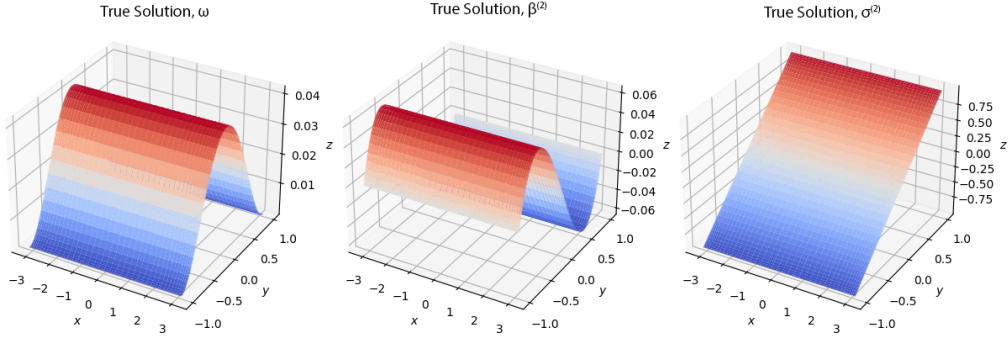


Figure 4.5: True solution ω (left), β (middle), and σ (right) with $t = 10^{-6}$ and $n = 0$.

Proof. First, we apply the Cauchy-Schwarz inequality with respect to L^2 to obtain

$$\begin{aligned} \mathfrak{B}_0((\beta, \omega); (\varphi, v)) &\leq \|\nabla \beta\|_{\Omega} \|\nabla \varphi\|_{\Omega} + t^{-2} \|\beta - \nabla \omega\|_{\Omega} \|\varphi - \nabla v\|_{\Omega} \\ &\leq \|\nabla \beta\|_{\Omega} \|\nabla \varphi\|_{\Omega} + t^{-2} (\|\beta\|_{H^1(\Omega)} + \|\omega\|_{H^1(\Omega)}) (\|\varphi\|_{H^1(\Omega)} + \|v\|_{H^1(\Omega)}). \end{aligned}$$

Applying the Cauchy-Schwarz inequality with respect to ℓ^2 as well as Young's inequality yields

$$\begin{aligned} \mathfrak{B}_0((\beta, \omega); (\varphi, v)) &\leq [\|\beta\|_{H^1(\Omega)}^2 + 2t^{-2} \|\beta\|_{H^1(\Omega)}^2 + 2t^{-2} \|\omega\|_{H^1(\Omega)}^2]^{1/2} \\ &\quad \cdot [\|\varphi\|_{H^1(\Omega)}^2 + 2t^{-2} \|\varphi\|_{H^1(\Omega)}^2 + 2t^{-2} \|v\|_{H^1(\Omega)}^2]^{1/2} \\ &\leq C_1 t^{-2} (\|\beta\|_{H^1(\Omega)}^2 + \|\omega\|_{H^1(\Omega)}^2)^{1/2} (\|\varphi\|_{H^1(\Omega)}^2 + \|v\|_{H^1(\Omega)}^2)^{1/2} \end{aligned}$$

where $C_1 = 3$. Similarly, applying Poincaré's inequality to β and ω and the triangle inequality yields

$$\begin{aligned} \|\beta\|_{H^1(\Omega)}^2 + \|\omega\|_{H^1(\Omega)}^2 &\leq C_p \|\nabla \beta\|_{\Omega}^2 + C_p \|\nabla \omega\|_{\Omega}^2 \leq C_2 (\|\nabla \beta\|_{\Omega}^2 + \|\beta - \nabla \omega\|_{\Omega}^2) \\ &\leq C_2 (\|\nabla \beta\|_{\Omega}^2 + t^{-2} \|\beta - \nabla \omega\|_{\Omega}^2). \end{aligned}$$

□

Proposition 4.4.1 means that Algorithm 1 is applicable to (4.17). However, we note

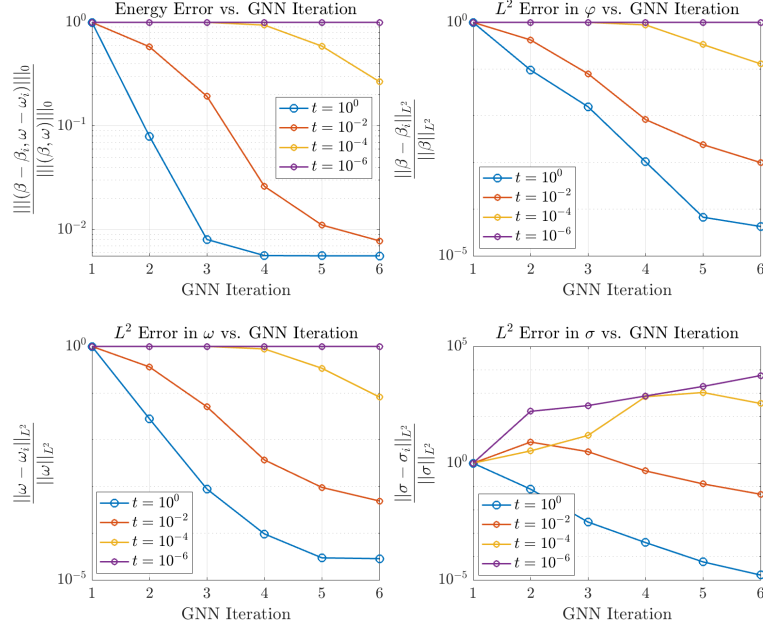


Figure 4.6: Relative error after each iteration of Algorithm 1 for the case when $n = 0$ with variational formulation on $\mathfrak{B}_0, \mathfrak{L}_0$.

that the ratio M/α is unbounded as $t \rightarrow 0$, which suggests that the approximation to the variational problem may not be quasi-optimal. In order to illustrate this, we begin by considering the simplest case of (4.15) in which the load $g(x, y) = 1$. This problem corresponds to (4.15) with $n = 0$. In this case, the true solution is univariate and does not vary in the x -coordinate, nor is it sensitive to the value of t . However, even though the solution is univariate in y , we do not explicitly enforce this in the neural network

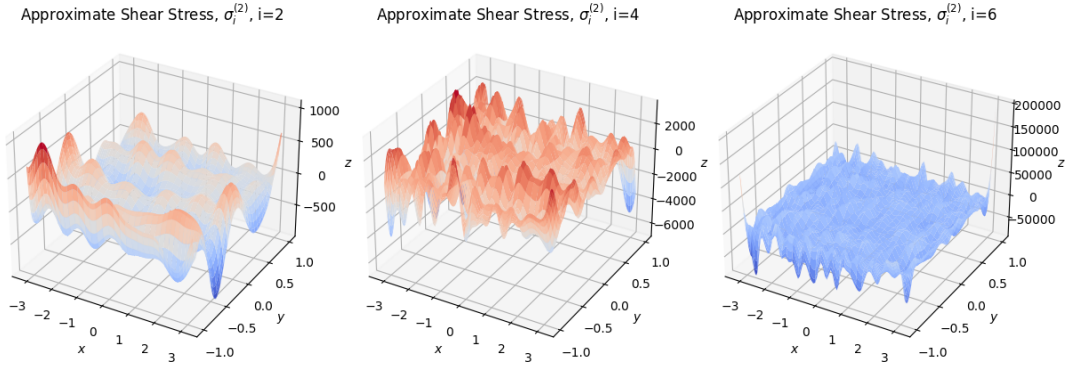


Figure 4.7: Approximate x -component of the shear stress for the 2nd, 4th, and 6th iterations of the Galerkin Neural Network algorithm for the univariate problem with variational formulation on $\mathfrak{B}_0, \mathfrak{L}_0$.

structure. Figure 4.5 shows the y -components of the true solutions ω and β , and σ when $t = 10^{-6}$.

Figure 4.6 shows a convergence plot of the error in the energy norm ($\mathfrak{B}_0$ -norm) after each iteration of Algorithm 1 as well as the L^2 error in the stress $\sigma = t^{-2}(\beta - \nabla\omega)$ and rotation and displacement β and ω for $t = 1, 10^{-2}, 10^{-4}, 10^{-6}$. We denote by φ_i^{NN} the basis functions for approximating β and by v_i^{NN} the basis functions for approximating ω . Figure 4.7 shows the approximate shear stress obtained by Algorithm 1 by computing $t^{-2}(\varphi_i - \nabla\omega_i)$ after the 2nd, 4th, 6th, and 8th iterations with $t = 10^{-6}$. The hyperparameters are as given at the beginning of Section 4.4.

Since the Galerkin Neural Network framework is not a direct Galerkin method, a natural question to ask is whether it exhibits locking. As the results show, it is immediately evident that as t decreases, the convergence rate of both the energy error and the L^2 errors are greatly reduced. Even worse, while the true shear stress is a simple linear function, the neural network approximation exhibits large spurious oscillations which dampen slowly as more iterations of Algorithm 1 are taken. This stalled convergence as t is decreased is reminiscent of the locking phenomenon observed in finite element approximation [15, 122] and suggests that the natural variational formulation (4.17), like with the finite element method, is not suitable for the Galerkin Neural Network framework.

4.4.2 Mixed Least Squares Variational Formulation

One common approach to help reduce, or even eliminate, the effects of locking consists of introducing the shear stress $\boldsymbol{\sigma}$ as a primal variable [10]. That is, the system

$$\begin{cases} -\Delta \boldsymbol{\beta} + \boldsymbol{\sigma} = \mathbf{0} & \text{in } \Omega \\ \operatorname{div} \boldsymbol{\sigma} = g & \text{in } \Omega \\ t^2 \boldsymbol{\sigma} - (\boldsymbol{\beta} - \nabla \omega) = \mathbf{0} & \text{in } \Omega \\ \boldsymbol{\beta} = \mathbf{0}, \omega = 0 & \text{on } \partial\Omega \end{cases} \quad (4.18)$$

is considered in lieu of (4.10). One disadvantage of this approach is that the resulting mixed variational formulation corresponds to the bilinear form

$$(\nabla \boldsymbol{\beta}, \nabla \boldsymbol{\varphi})_\Omega + (\boldsymbol{\sigma}, \boldsymbol{\varphi} - \nabla v)_\Omega + (\boldsymbol{\beta} - \nabla \omega, \boldsymbol{\tau})_\Omega - t^2 (\boldsymbol{\sigma}, \boldsymbol{\tau})_\Omega$$

which, although symmetric, is not positive definite (e.g. take $\boldsymbol{\beta} = \boldsymbol{\varphi} = \mathbf{0}$, $\omega = v = 0$, and $\boldsymbol{\sigma} = \boldsymbol{\tau} \neq \mathbf{0}$), so Algorithm 1 is not applicable. Instead, we shall consider an alternative mixed formulation based on least squares variational principles. Setting $X_{LS} := \mathbf{H}_0^1(\Delta; \Omega) \times H_0^1(\Omega) \times \mathbf{H}(\operatorname{div}; \Omega)$ where $\mathbf{H}_0^1(\Delta; \Omega) := \{\mathbf{v} \in \mathbf{H}_0^1(\Omega) : \Delta \mathbf{v} \in \mathbf{L}^2(\Omega)\}$ with norm

$$\|(\boldsymbol{\beta}, \omega, \boldsymbol{\sigma})\|_{LS} := (\|\boldsymbol{\beta}\|_{H^1(\Omega)}^2 + \|t\Delta \boldsymbol{\beta}\|_\Omega^2 + \|\omega\|_{H^1(\Omega)}^2 + \|t\boldsymbol{\sigma}\|_\Omega^2 + \|\nabla \cdot \boldsymbol{\sigma}\|_\Omega^2)^{1/2},$$

we define the bilinear operator $\mathfrak{B}_{LS} : X_{LS} \times X_{LS} \rightarrow \mathbb{R}$ given by

$$\begin{aligned} \mathfrak{B}_{LS}((\boldsymbol{\beta}, \omega, \boldsymbol{\sigma}); (\boldsymbol{\varphi}, v, \boldsymbol{\tau})) &:= t^2(-\Delta \boldsymbol{\beta} + \boldsymbol{\sigma}, -\Delta \boldsymbol{\varphi} + \boldsymbol{\tau})_\Omega + (\operatorname{div} \boldsymbol{\sigma}, \operatorname{div} \boldsymbol{\tau})_\Omega \\ &\quad + (t^2 \boldsymbol{\sigma} - (\boldsymbol{\beta} - \nabla \omega), t^2 \boldsymbol{\tau} - (\boldsymbol{\varphi} - \nabla v))_\Omega \end{aligned} \quad (4.19)$$

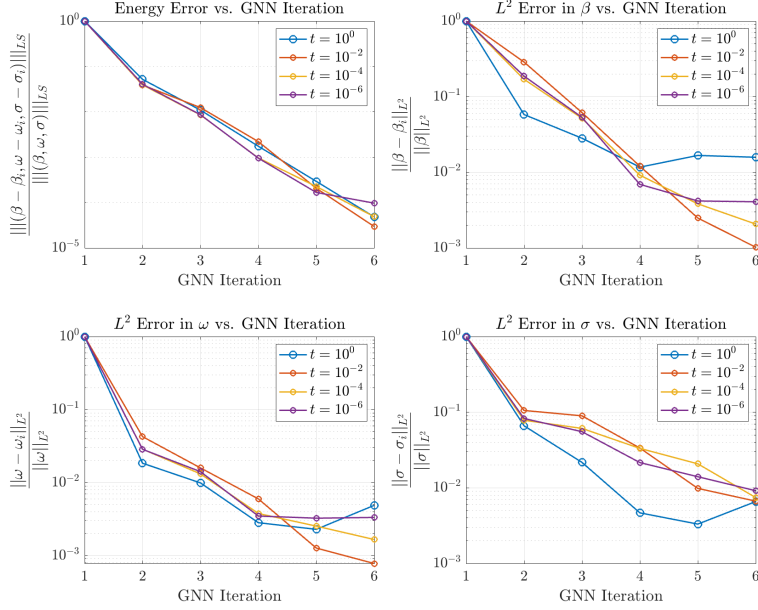


Figure 4.8: Relative errors after each iteration of Algorithm 1 for the univariate problem with variational formulation on $\mathfrak{B}_{LS}, \mathfrak{L}_{LS}$.

and the linear operator $\mathfrak{L}_{LS} : X_{LS} \rightarrow \mathbb{R}$ given by

$$\mathfrak{L}_{LS}(\varphi, v, \tau) := (g, \operatorname{div} \tau)_{\Omega}. \quad (4.20)$$

The bilinear operator $\mathfrak{B}_{LS}$ is both continuous and coercive, allowing us to apply Algorithm 1. However, we note that the coercivity estimate is again degenerate in t , which suggests we might expect deterioration of the numerical approximation as the plate thickness is reduced.

Proposition 4.4.2. *Let $(\beta, \omega, \sigma) \in X_{LS}$ and $0 < t \leq 1$. There exist constants $C_1 > 0$ and $C_2 > 0$ independent of t such that*

$$\mathfrak{B}_{LS}((\beta, \omega, \sigma); (\varphi, v, \tau)) \leq C_1 \|(\beta, \omega, \sigma)\|_{LS} \cdot \|(\varphi, v, \tau)\|_{LS} \quad \forall (\beta, \omega, \sigma), (\varphi, v, \tau) \in X_{LS}$$

$$C_2 t^2 \|(\beta, \omega, \sigma)\|_{LS}^2 \leq \mathfrak{B}_{LS}((\beta, \omega, \sigma); (\beta, \omega, \sigma)) \quad \forall (\beta, \omega, \sigma) \in X_{LS}.$$

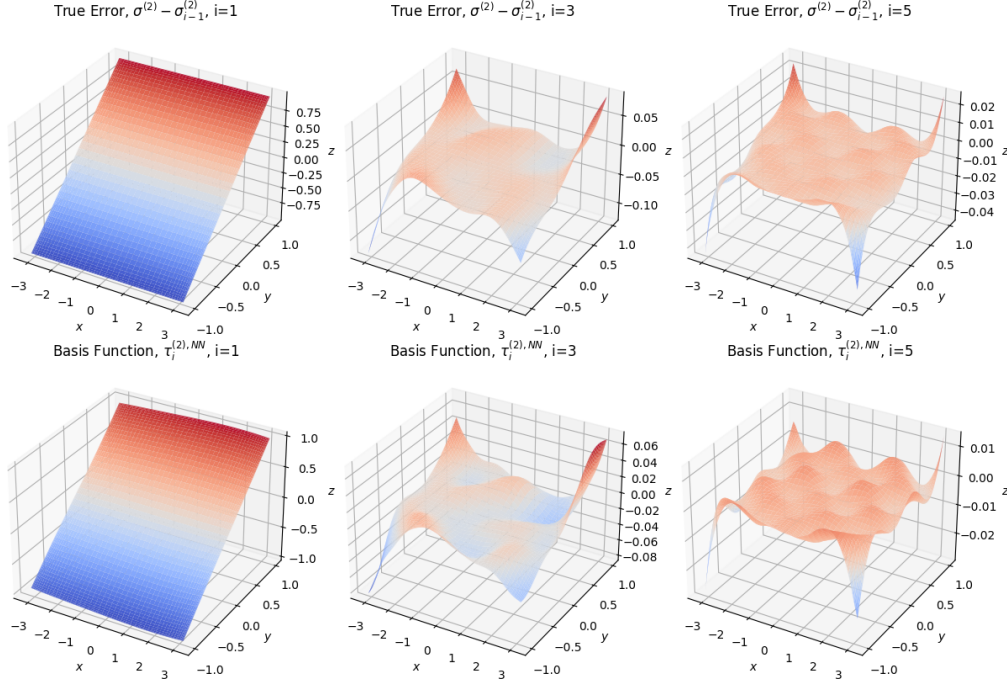


Figure 4.9: For $t = 10^{-6}$, the y -component of the true error $\sigma - \sigma_i$ (top row) and the y -component of the basis function τ_i^{NN} (bottom row) for the 1st, 3rd, and 5th iterations of Algorithm 1 for the univariate problem with variational formulation on $\mathfrak{B}_{LS}, \mathfrak{L}_{LS}$.

Proof. For the first inequality, we apply the Cauchy-Schwarz inequality with respect to L^2 to obtain

$$\begin{aligned}
\mathfrak{B}_{LS}((\beta, \omega, \sigma); (\varphi, v, \tau)) &\leq \|t(-\Delta\beta + \sigma)\|_{\Omega} \cdot \|t(-\Delta\varphi + \tau)\|_{\Omega} + \|\nabla \cdot \sigma\|_{\Omega} \cdot \|\nabla \cdot \tau\|_{\Omega} \\
&\quad + \|t^2\sigma - (\beta - \nabla\omega)\|_{\Omega} \cdot \|t^2\tau - (\varphi - \nabla v)\|_{\Omega} \\
&\leq (\|t\Delta\beta\|_{\Omega} + \|t\sigma\|_{\Omega}) \cdot (\|t\Delta\varphi\|_{\Omega} + \|t\tau\|_{\Omega}) + \|\nabla \cdot \sigma\|_{\Omega} \cdot \|\nabla \cdot \tau\|_{\Omega} \\
&\quad + (\|t\sigma\|_{\Omega} + \|\beta\|_{H^1(\Omega)} + \|\omega\|_{H^1(\Omega)}) \cdot \\
&\quad (\|t\tau\|_{\Omega} + \|\varphi\|_{H^1(\Omega)} + \|v\|_{H^1(\Omega)}).
\end{aligned}$$

An application of the Cauchy-Schwarz inequality with respect to ℓ^2 yields the result.

As for the second inequality, given $(\boldsymbol{\beta}, \omega, \boldsymbol{\sigma}) \in X_{LS}$, we form the system

$$\begin{cases} \mathbf{g}_1 := -\Delta \boldsymbol{\beta} + \boldsymbol{\sigma} & \text{in } \Omega \\ g_2 := \nabla \cdot \boldsymbol{\sigma} & \text{in } \Omega \\ t^2 \mathbf{g}_3 := t^2 \boldsymbol{\sigma} - (\boldsymbol{\beta} - \nabla \omega) & \text{in } \Omega \\ \boldsymbol{\beta} = \mathbf{0}, \omega = 0 & \text{on } \partial\Omega. \end{cases} \quad (4.21)$$

We can form the following variational formulation: $(\boldsymbol{\beta}, \omega, \boldsymbol{\sigma}) \in X_{LS}$ s.t.

$$\begin{cases} (\nabla \boldsymbol{\beta}, \nabla \boldsymbol{\varphi})_\Omega + (\boldsymbol{\sigma}, \boldsymbol{\varphi})_\Omega & = (\mathbf{g}_1, \boldsymbol{\varphi})_\Omega \quad \forall \boldsymbol{\varphi} \in X_\beta \\ (\nabla \cdot \boldsymbol{\sigma}, v)_\Omega & = (g_2, v)_\Omega \quad \forall v \in X_\omega \\ t^2(\boldsymbol{\sigma}, \boldsymbol{\tau})_\Omega - (\boldsymbol{\beta}, \boldsymbol{\tau})_\Omega + (\nabla \omega, \boldsymbol{\tau})_\Omega & = t^2(\mathbf{g}_3, \boldsymbol{\tau})_\Omega \quad \forall \boldsymbol{\tau} \in X_\sigma. \end{cases} \quad (4.22)$$

Setting $\boldsymbol{\varphi} = \boldsymbol{\beta}$, $v = \omega$, and $\boldsymbol{\tau} = \boldsymbol{\sigma}$, integrating the third equation of (4.22) by parts, and adding all three equations together yields

$$\|\nabla \boldsymbol{\beta}\|_\Omega^2 + t^2 \|\boldsymbol{\sigma}\|_\Omega^2 \leq \|\mathbf{g}_1\|_\Omega \cdot \|\boldsymbol{\beta}\|_\Omega + \|g_2\|_\Omega \cdot \|\omega\|_\Omega + t \|\mathbf{g}_3\|_\Omega \cdot t \|\boldsymbol{\sigma}\|_\Omega. \quad (4.23)$$

For the second term on the RHS of (4.23), we observe by the Poincare inequality that

$$\|\omega\|_\Omega \leq C \|\nabla \omega\|_\Omega \leq C \|t^2 \mathbf{g}_3 - t^2 \boldsymbol{\sigma} + \boldsymbol{\beta}\|_\Omega \leq C(t^2 \|\mathbf{g}_3\|_\Omega + t^2 \|\boldsymbol{\sigma}\|_\Omega + \|\boldsymbol{\beta}\|_\Omega),$$

from which we obtain

$$\begin{aligned} \|\nabla \boldsymbol{\beta}\|_\Omega^2 + t^2 \|\boldsymbol{\sigma}\|_\Omega^2 &\leq C(\|\mathbf{g}_1\|_\Omega + \|g_2\|_\Omega) \cdot \|\boldsymbol{\beta}\|_\Omega + C\|g_2\|_\Omega \cdot t^2 \|\mathbf{g}_3\|_\Omega \\ &\quad + C(t\|g_2\|_\Omega + t\|\mathbf{g}_3\|_\Omega) \cdot t \|\boldsymbol{\sigma}\|_\Omega. \end{aligned}$$

Now, the first term on the RHS can be dealt with using Poincare's inequality and the

ϵ -Young's inequality while the second and third terms on the RHS can be dealt with using the ϵ -Young's inequality:

$$\|\nabla \boldsymbol{\beta}\|_{\Omega}^2 + t^2 \|\boldsymbol{\sigma}\|_{\Omega}^2 + \|\nabla \omega\|_{\Omega}^2 \leq C(\|\mathbf{g}_1\|_{\Omega}^2 + \|g_2\|_{\Omega}^2 + \|t^2 \mathbf{g}_3\|_{\Omega}^2).$$

As for $\|\nabla \cdot \boldsymbol{\beta}\|_{\Omega}$, we have $\|\nabla \cdot \boldsymbol{\beta}\|_{\Omega} = \|g_2\|_{\Omega}$. Finally, for the second-order term, we consider

$$-t\Delta \boldsymbol{\beta} + t\boldsymbol{\sigma} = t\mathbf{g}_1$$

to obtain $\|t\Delta \boldsymbol{\beta}\|_{\Omega} \leq \|t\boldsymbol{\sigma}\|_{\Omega} + \|t\mathbf{g}_1\|_{\Omega}$. Altogether, we have

$$\begin{aligned} \|(\boldsymbol{\beta}, \omega, \boldsymbol{\sigma})\|_{LS}^2 &\leq C(\|\mathbf{g}_1\|_{\Omega}^2 + \|t\mathbf{g}_1\|_{\Omega}^2 + \|g_2\|_{\Omega}^2 + \|t^2 \mathbf{g}_3\|_{\Omega}^2) \\ &\leq Ct^{-2}(\|t\mathbf{g}_1\|_{\Omega}^2 + \|g_2\|_{\Omega}^2 + \|t^2 \mathbf{g}_3\|_{\Omega}^2) \leq Ct^{-2} \cdot \mathfrak{B}_{LS}((\boldsymbol{\beta}, \omega, \boldsymbol{\sigma}); (\boldsymbol{\beta}, \omega, \boldsymbol{\sigma})). \end{aligned}$$

□

While the least squares formulation based on (4.18) seems a natural choice, hitherto it has not been employed in practice. One reason is that a direct application of the least squares functional to second-order problems requires that the approximation space be a conforming subspace of H^2 , i.e., continuously differentiable elements are required when using finite element methods, which is often viewed as unattractive by finite element practitioners. To circumvent this issue, several approaches exist in the literature. The first is to reduce second order problems to first order problems by introducing auxiliary variables, which increases the size of the corresponding linear system. The second more sophisticated approach is to recast the least squares formulation in terms of a negative norm (i.e. H^{-1}) residual, which in the context of finite elements allows one to retain the advantages of least squares formulations while only requiring continuous basis functions,

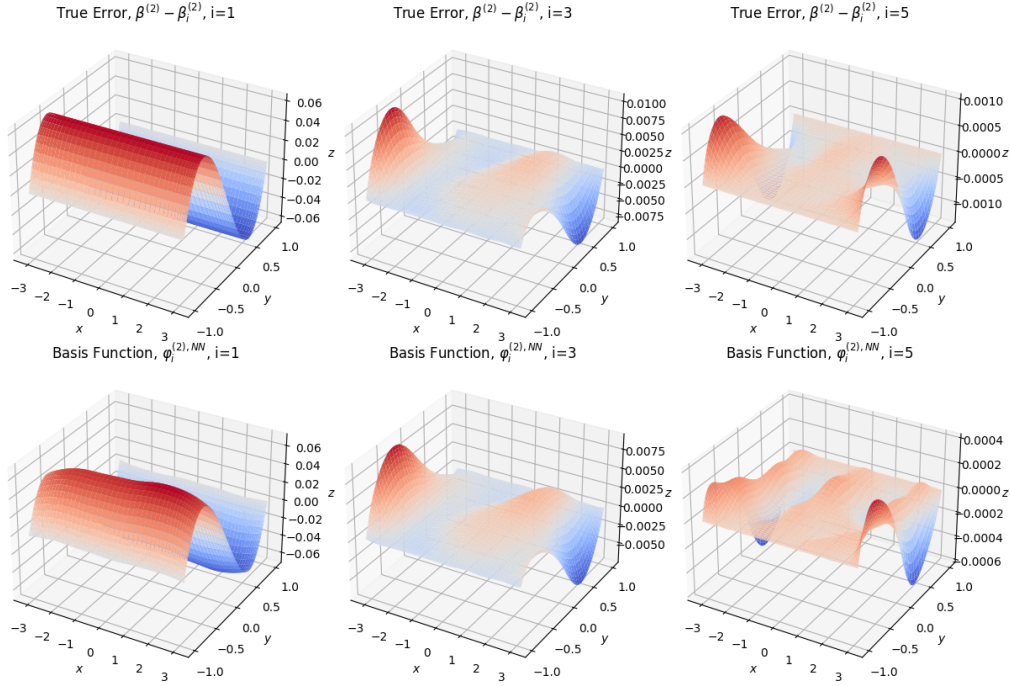


Figure 4.10: For $t = 10^{-6}$, the y -component of the true error $\beta - \beta_i$ (top row) and the y -component of the basis function φ_i^{NN} (bottom row) for the 1st, 3rd, and 5th iterations of Algorithm 1 for the univariate problem with variational formulation on $\mathfrak{B}_{LS}, \mathfrak{L}_{LS}$.

as in [26, 31].

One advantage of applying the Galerkin Neural Network framework to the H^2 least squares formulation is that the regularity and global nature of functions in the set $V_{n,C}^\sigma$ is determined solely by the regularity of the activation function σ . In other words, choosing $\sigma \in C^2(\bar{\Omega})$ is sufficient to ensure that the resulting neural network functions are H^2 -conforming.

4.4.2.1 Benchmark Problem with Constant Load

We return to the constant load univariate problem described in Section 4.4.1, this time applying Algorithm 1 to the variational formulation involving $\mathfrak{B}_{LS}$ and $\mathfrak{L}_{LS}$. The hyperparameters are chosen as described at the beginning of Section 4.4.

Figures 4.8-4.11 show the analogous results to Figures 4.6-4.7 for the least squares approach described by (4.19). We denote by φ_i^{NN} , v_i^{NN} , and τ_i^{NN} the basis functions

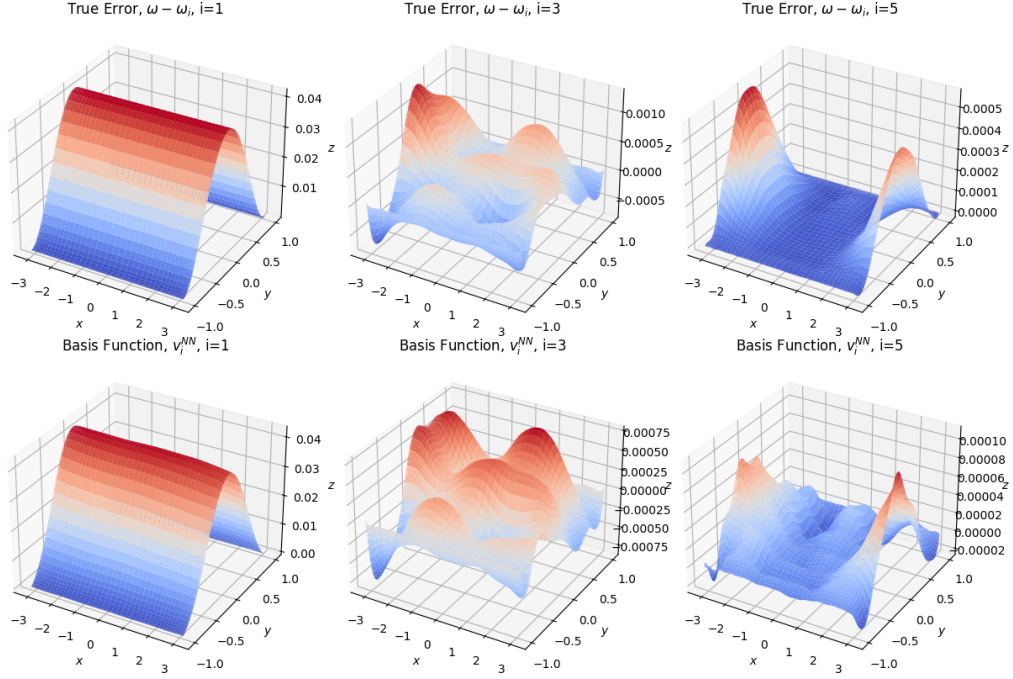


Figure 4.11: For $t = 10^{-6}$, the true error $\omega - \omega_i$ (top row) and the basis function v_i^{NN} (bottom row) for the 1st, 3rd, and 5th iterations of Algorithm 1 for the univariate problem with variational formulation on $\mathfrak{B}_{LS}, \mathfrak{L}_{LS}$.

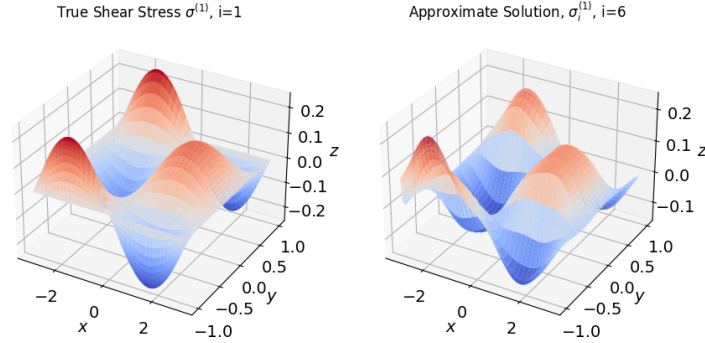


Figure 4.12: Left: True x -component of shear stress. Right: Neural network approximation of x -component of shear stress with variational formulation on $\mathfrak{B}_{LS}, \mathfrak{L}_{LS}$.

used to approximate β , ω , and σ , respectively. Figures 4.10-4.11 show comparisons of the basis functions φ_i^{NN} and v_i^{NN} with the errors in β and ω , respectively. We observe that no locking effect nor any spurious oscillations are present in the shear stress.

4.4.2.2 Benchmark Problem with Sinusoidal Load and Boundary Layer

We again consider the problem described in Section 4.3, this time with $n = 1$. In this case, the solution is fully two-dimensional and the shear stress contains a boundary

layer of width $\mathcal{O}(t^{-1})$. Figure 4.12 shows the true shear stress in the x -coordinate when $t = 10^{-2}$ as well as the neural network approximation to the shear stress when $\mathfrak{B}_{LS}$ and $\mathfrak{L}_{LS}$ are used in the variational formulation with Algorithm 1. Figure 4.13 shows the errors in the $\mathfrak{B}_{LS}$ -norm and L^2 -norm for each primal variable. The hyperparameters are chosen as described at the beginning of Section 4.4.

We observe that while the mixed least squares variational formulation on X_{LS} is capable of providing a uniformly accurate approximation in t when the solution is smooth, the same cannot be said for problems with boundary layers. In particular, we observe that most of the approximation error in the x -coordinate of the shear stress is encoded in its y -derivative $-\partial\sigma^{(1)}/\partial y$ – while the variational formulation posed on $\mathfrak{B}_{LS}$, $\mathfrak{L}_{LS}$ only contains information about $\partial\sigma^{(1)}/\partial x$.

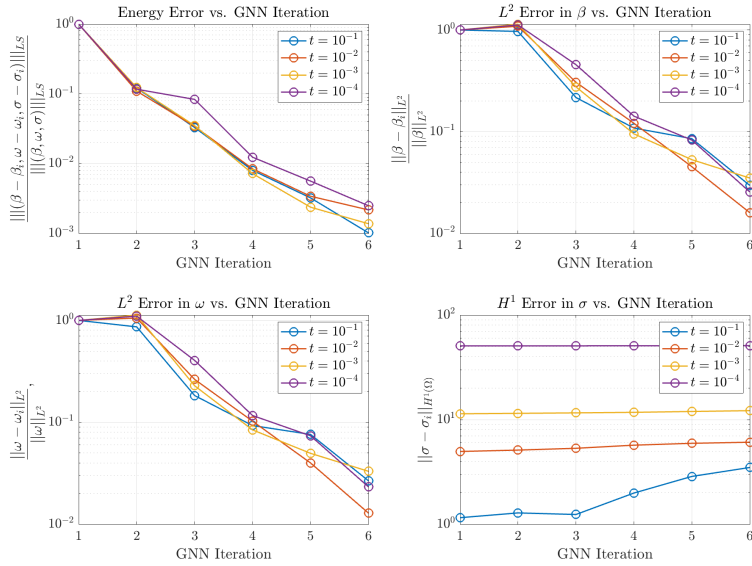


Figure 4.13: Relative energy error, relative L^2 error in β and ω , and H^1 error in σ after each iteration of Algorithm 1 for the Reissner-Mindlin plate with boundary layer and variational formulation on $\mathfrak{B}_{LS}$, $\mathfrak{L}_{LS}$.

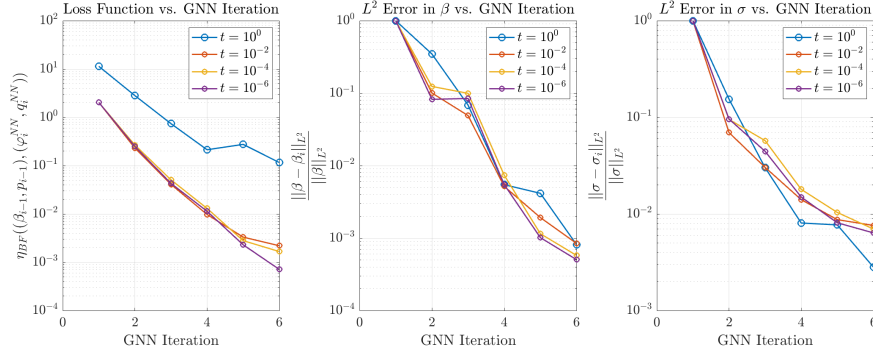


Figure 4.14: Errors after each iteration of Algorithm 1 for the univariate problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}, \tilde{\mathfrak{L}}_{BF}$.

4.4.3 Least Squares Based on Brezzi-Fortin Formulation

A third variational formulation of the Reissner-Mindlin problem starts by considering the Helmholtz decomposition of the shear stress which explicitly accounts for both the irrotational and solenoidal components of the shear stress. By writing the Helmholtz decomposition of the shear stress as $\boldsymbol{\sigma} = \nabla^\perp p - \nabla r$, Brezzi and Fortin arrived at the following equivalent formulation to (4.17) [29]: seek $(r, \boldsymbol{\beta}, p, \omega) \in H_0^1(\Omega) \times \mathbf{H}_0^1(\Omega) \times H^1(\Omega)/\mathbb{R} \times H_0^1(\Omega)$ such that

$$\begin{cases} (\nabla r, \nabla \mu)_\Omega = (g, \mu)_\Omega & \forall \mu \in H_0^1(\Omega) \\ (\nabla \boldsymbol{\beta}, \nabla \boldsymbol{\varphi})_\Omega + (\nabla^\perp p, \boldsymbol{\varphi})_\Omega = (\nabla r, \boldsymbol{\varphi})_\Omega & \forall \boldsymbol{\varphi} \in \mathbf{H}_0^1(\Omega) \\ (\boldsymbol{\beta}, \nabla^\perp q)_\Omega - t^2(\nabla^\perp p, \nabla^\perp q)_\Omega = 0 & \forall q \in H^1(\Omega)/\mathbb{R} \\ (\nabla \omega, \nabla v)_\Omega = (\boldsymbol{\beta} + t^2 \nabla r, \nabla v)_\Omega & \forall v \in H_0^1(\Omega). \end{cases} \quad (4.24)$$

Thus, the system (4.24) may be solved in three stages: first, a straightforward solution of the Poisson equation with data g to obtain r ; second, a solution of the perturbed, rotated Stokes problem with data $(\nabla r, 0)$ to obtain $\boldsymbol{\beta}$ and p ; and finally, another straightforward solution of the Poisson equation with data $-\nabla \cdot (\boldsymbol{\beta} + t^2 \nabla r)$ to obtain ω .

In this work, we shall apply the Galerkin Neural Network algorithm based on a least squares formulation of each of the equations of (4.24). We shall focus the brunt of our

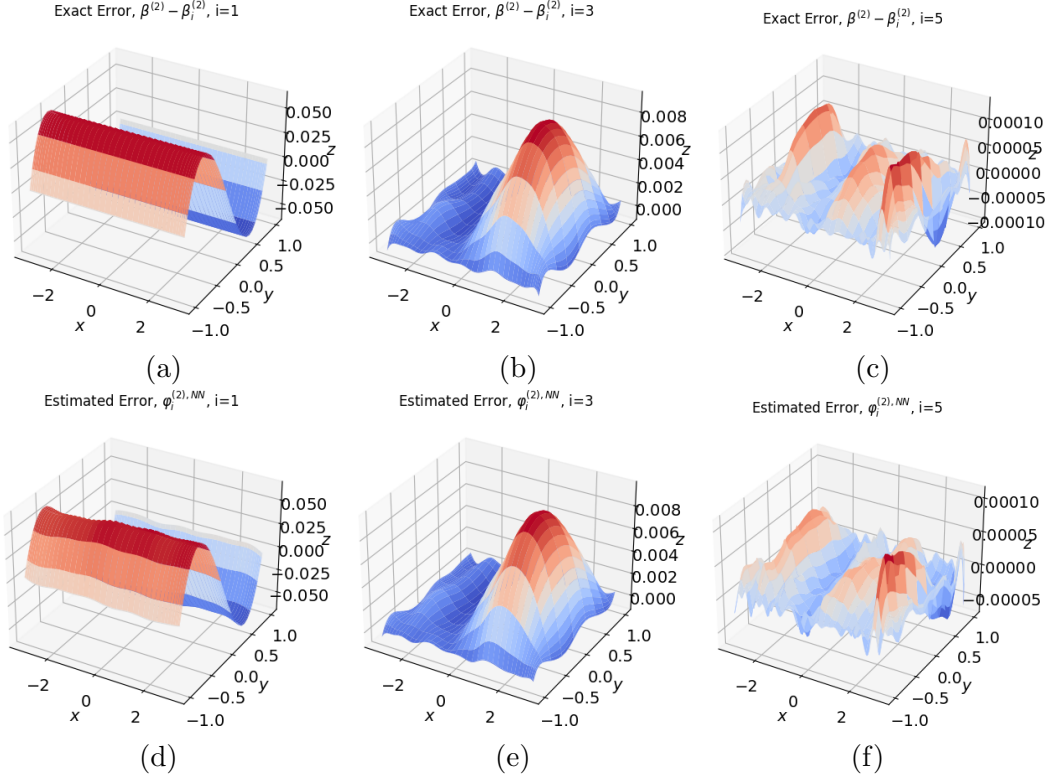


Figure 4.15: For $t = 10^{-6}$ in the univariate problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: (a)-(c) True error $\beta^{(2)} - \beta_i^{(2)}$. (d)-(f) Basis function $\varphi_i^{(2), NN}$.

attention on the inner perturbed Stokes problem. As stated in (4.24), the inner product described by

$$((\beta, p), (\beta, p)) \mapsto (\nabla \beta, \nabla \varphi)_{\Omega} + (\nabla^{\perp} p, \varphi)_{\Omega} + (\beta, \nabla^{\perp} q)_{\Omega} - t^2 (\nabla^{\perp} p, \nabla^{\perp} q)_{\Omega}$$

is not positive definite. In particular, the choice $\beta = \varphi = (1, 1)^T$ and $p = q = x + y$ yields $((\beta, p), (\beta, p)) \mapsto -2t^2|\Omega| < 0$. Instead, we consider a least squares formulation of the perturbed Stokes problem, which consists of the variational problem posed on $X_{BF} := (\mathbf{H}^2(\Omega) \cap \mathbf{H}_0^1(\Omega)) \times (H^1(\Delta; \Omega)/\mathbb{R})$ where $H^1(\Delta; \Omega) := \{v \in H^1(\Omega) : \Delta v \in L^2(\Omega)\}$ and X_{BF} is endowed with the norm

$$\|(\varphi, q)\|_{BF} := (\|\varphi\|_{H^2(\Omega)}^2 + \|\frac{1}{t}(\nabla \times \beta)\|_{\Omega}^2 + \|q\|_{H^1(\Omega)}^2 + \|t\Delta q\|_{L^2(\Omega)}^2 + \|t\partial_n q\|_{H^{1/2}(\partial\Omega)}^2)^{1/2}$$

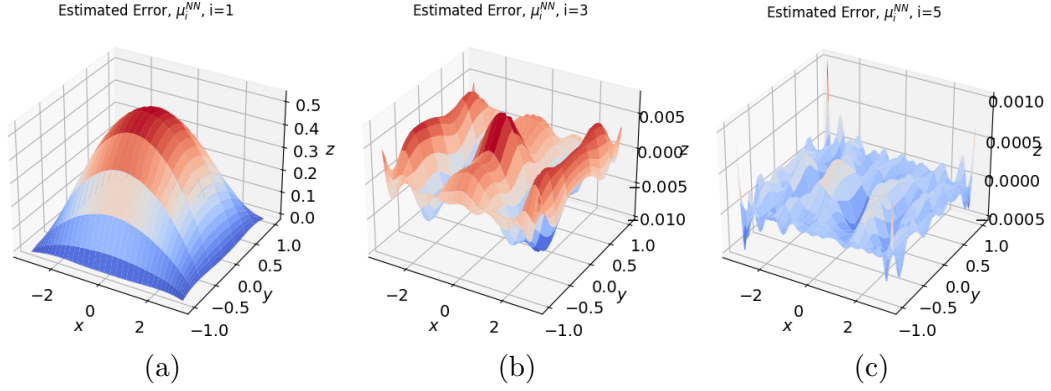


Figure 4.16: For $t = 10^{-6}$ in the univariate problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: basis function μ_i^{NN} .

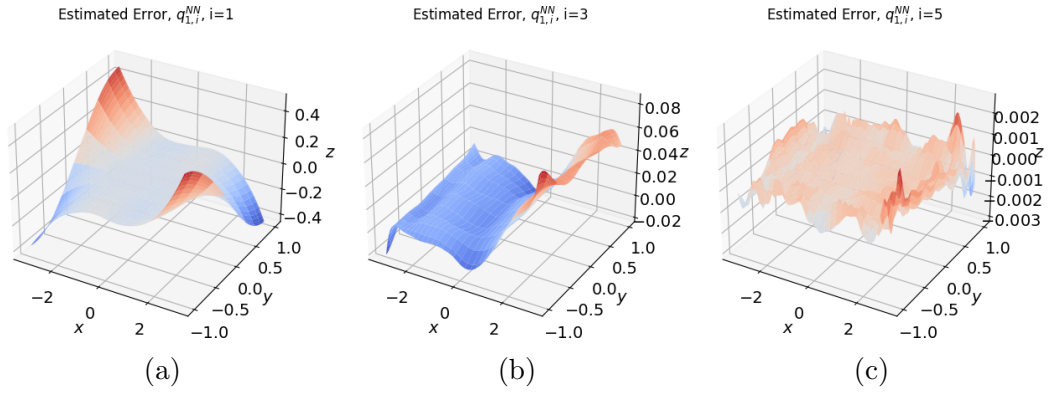


Figure 4.17: For $t = 10^{-6}$ in the univariate problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: basis function $q_i^{(2),NN}$.

with

$$\begin{aligned} \mathfrak{B}_{BF}((\boldsymbol{\beta}, p); (\boldsymbol{\varphi}, q)) &:= (-\Delta \boldsymbol{\beta} + \nabla^\perp p, -\Delta \boldsymbol{\varphi} + \nabla^\perp q)_\Omega + t^{-2}(\nabla \times \boldsymbol{\beta} + t^2 \Delta p, \nabla \times \boldsymbol{\varphi} + t^2 \Delta q)_\Omega \\ &\quad + t^2(\partial_n p, \partial_n q)_{H^{1/2}(\partial\Omega)} \\ \mathfrak{L}_{BF}((\boldsymbol{\varphi}, q)) &:= (\nabla r, -\Delta \boldsymbol{\varphi} + \nabla^\perp q)_\Omega. \end{aligned}$$

Here, $H^{1/2}(\Omega)$ is the Sobolev-Slobodeckij space [41] and $(\cdot, \cdot)_{H^{1/2}(\Omega)}$ is its associated inner product.

The following result demonstrates that the Helmholtz least squares variational formulation for $\boldsymbol{\beta}$ and p is both continuous and coercive, thus allowing us to apply Algorithm

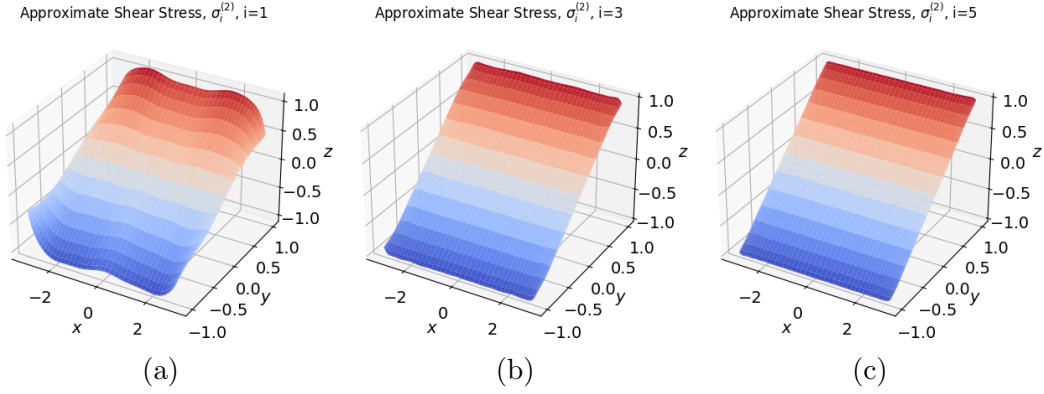


Figure 4.18: For $t = 10^{-2}$ in the univariate problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: approximation $\sigma_i^{(2)} = -\partial p_i / \partial x - \partial r_i / \partial y$.

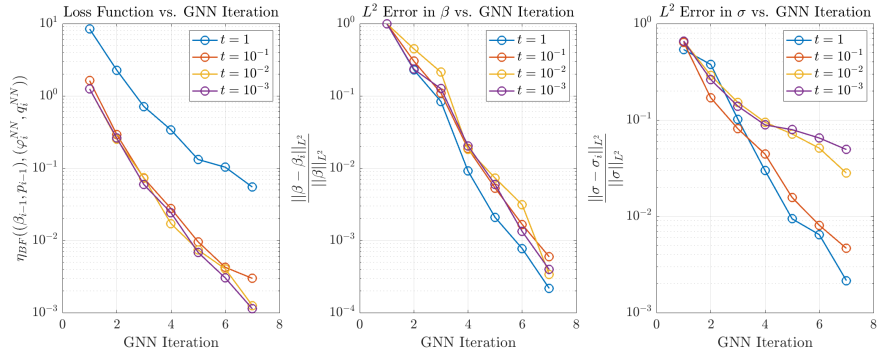


Figure 4.19: Energy error and L^2 error in β and ω for the problem with sinusoidal load with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$.

1.

Proposition 4.4.3. *Let $(\beta, p) \in X_{BF}$ and suppose $0 < t \leq 1$. There exist constants $C_1 > 0$ and $C_2 > 0$ independent of t such that*

$$\begin{aligned} \mathfrak{B}_{BF}(\beta, p; \varphi, q) &\leq C_1 \|(\beta, p)\|_{BF} \|(\varphi, q)\|_{X_{BF}} & \forall (\beta, p), (\varphi, q) \in X_{BF} \\ C_2 \|(\beta, p)\|_{BF}^2 &\leq \mathfrak{B}_{BF}(\beta, p; \beta, p) & \forall (\beta, p) \in X_{BF}. \end{aligned}$$

Proof. We begin by applying the Cauchy-Schwarz inequality with respect to L^2 and $H^{-1/2}$:

$$\begin{aligned}
\mathfrak{B}_{BF}((\boldsymbol{\beta}, p); (\boldsymbol{\varphi}, q)) &\leq \| -\Delta \boldsymbol{\beta} + \nabla^\perp p \|_\Omega \cdot \| -\Delta \boldsymbol{\varphi} + \nabla^\perp q \|_\Omega \\
&\quad + \left\| \frac{1}{t} (\nabla \times \boldsymbol{\beta} + t^2 \Delta p) \right\|_\Omega \cdot \left\| \frac{1}{t} (\nabla \times \boldsymbol{\varphi} + t^2 \Delta q) \right\|_\Omega \\
&\quad + t^2 \| \partial_n p \|_{H^{1/2}(\partial\Omega)} \cdot \| \partial_n q \|_{H^{1/2}(\partial\Omega)} \\
&\leq (\| \boldsymbol{\beta} \|_{H^2(\Omega)} + \| p \|_{H^1(\Omega)}) (\| \boldsymbol{\varphi} \|_{H^2(\Omega)} + \| q \|_{H^1(\Omega)}) \\
&\quad + \left(\left\| \frac{1}{t} (\nabla \times \boldsymbol{\beta}) \right\|_\Omega + \| t p \|_{H^2(\Omega)} \right) \cdot \left(\left\| \frac{1}{t} (\nabla \times \boldsymbol{\varphi}) \right\|_\Omega + \| t q \|_{H^2(\Omega)} \right) \\
&\quad + t^2 \| \partial_n p \|_{H^{1/2}(\partial\Omega)} \cdot \| \partial_n q \|_{H^{1/2}(\partial\Omega)}.
\end{aligned}$$

An application of the Cauchy-Schwarz inequality with respect to ℓ^2 yields the first result.

As for the second inequality, given $(\boldsymbol{\beta}, p) \in X_{BF}$, we form the system

$$\begin{cases} \mathbf{g}_1 := -\Delta \boldsymbol{\beta} + \nabla^\perp p & \text{in } \Omega \\ t \cdot g_2 := \nabla \times \boldsymbol{\beta} + t^2 \Delta p & \text{in } \Omega \\ \boldsymbol{\beta} = \mathbf{0} & \text{on } \partial\Omega \\ g_3 := t \cdot \partial_n p & \text{on } \partial\Omega. \end{cases} \quad (4.25)$$

From [30], we have the a priori estimate

$$\| \boldsymbol{\beta} \|_{H^2(\Omega)}^2 + \| p \|_{H^1(\Omega)}^2 + \| t \Delta p \|_\Omega^2 \leq C (\| \mathbf{g}_1 \|_\Omega^2 + \| g_2 \|_\Omega^2 + \| t g_3 \|_{H^{1/2}(\partial\Omega)}^2).$$

Additionally, we have

$$\begin{aligned}
\left\| \frac{1}{t} (\nabla \times \boldsymbol{\beta}) \right\|_\Omega &\leq \| t \Delta p \|_\Omega + \| g_2 \|_\Omega \\
&\leq C (\| \mathbf{g}_1 \|_\Omega + \| g_2 \|_\Omega + \| t g_3 \|_{H^{1/2}(\partial\Omega)})
\end{aligned}$$

which completes the proof. $\square$

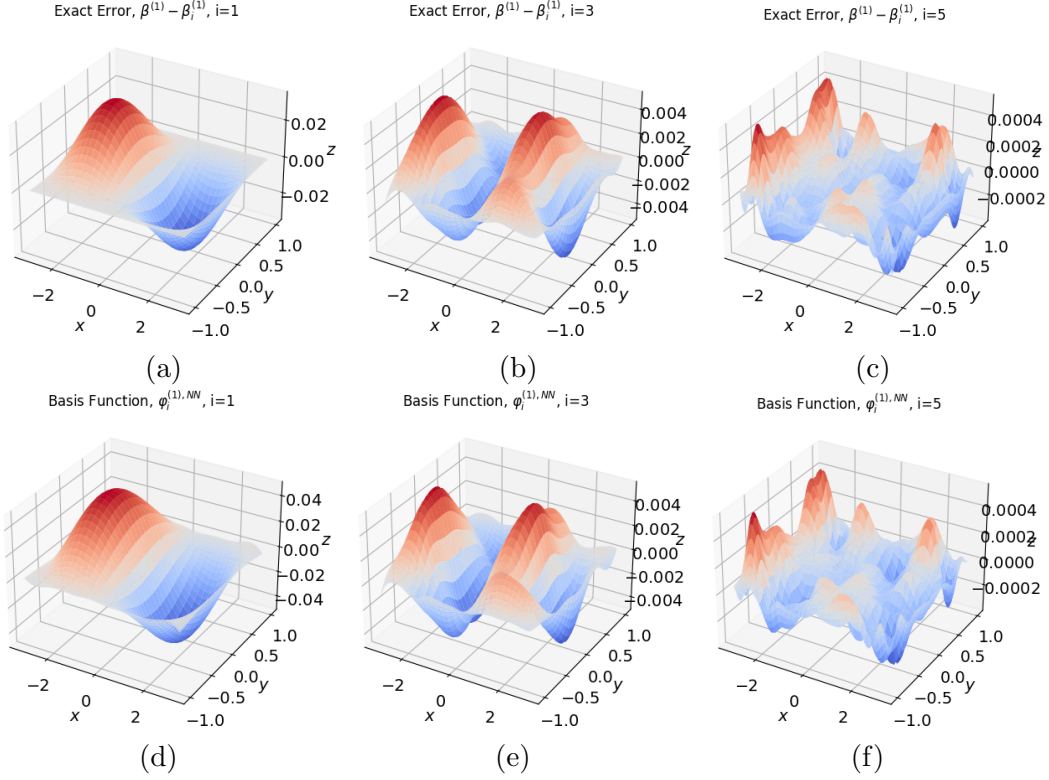


Figure 4.20: For $t = 10^{-2}$ in the problem with sinusoidal load with variational formulation on $\mathfrak{B}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: (a)-(c) True error $\beta^{(1)} - \beta_i^{(1)}$. (d)-(f) Basis function $\varphi_i^{(1), NN}$.

Approaches based on the H^{-1} norm of the residual of the second equation in (4.24) and the L^2 norm of the residual of the third equation have again been explored in [25, 30], but approaches based fully on the L^2 norm of the interior residuals, as described by $\mathfrak{B}_{BF}$, have not been used hitherto in practice due to the necessity of H^2 regularity. Nevertheless, Proposition 4.4.3 shows that the ratio M/α for the formulation based on $\mathfrak{B}_{BF}$ and $\mathfrak{L}_{BF}$ is independent of t and a quasi-optimal approximation should be expected.

We note that the term $t^2(\partial_n p, \partial_n q)_{H^{1/2}(\partial\Omega)}$ corresponds to the enforcement of the boundary condition $\partial_n p = 0$. The computation of $H^{1/2}$ inner products is not straightforward but could, in principle, be achieved using singular integrals [110]. Instead, for simplicity and to avoid unnecessary technical distractions, we elect to impose the boundary condition on p in a slightly weaker sense by considering the penalization $t^2(\partial_n p, \partial_n q)_{\partial\Omega}$. More specifically,

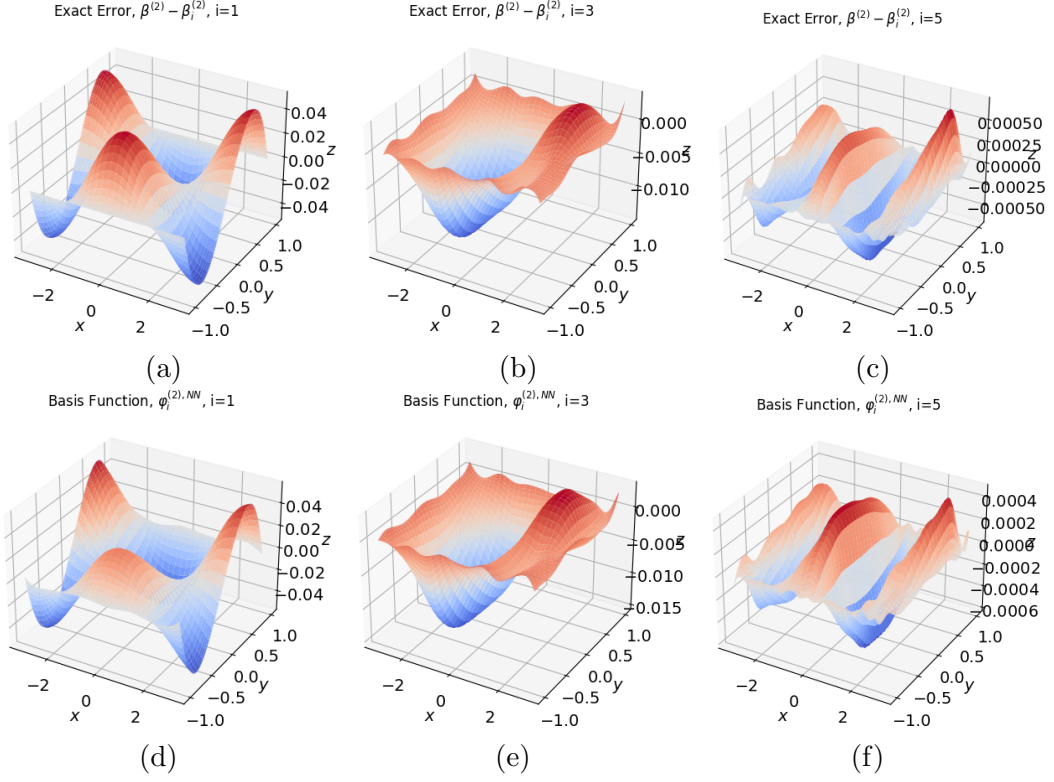


Figure 4.21: For $t = 10^{-2}$ in the problem with sinusoidal load with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: (a)-(c) True error $\beta^{(2)} - \beta_i^{(2)}$. (d)-(f) Basis function $\varphi_i^{(2),NN}$.

we consider the modified bilinear form

$$\begin{aligned} \tilde{\mathfrak{B}}_{BF}((\beta, p); (\varphi, q)) &:= (-\Delta\beta + \nabla^\perp p, -\Delta\varphi + \nabla^\perp q)_\Omega + t^{-2}(\nabla \times \beta + t^2 \Delta p, \nabla \times \varphi + t^2 \Delta q)_\Omega \\ &\quad + t^2(\partial_n p, \partial_n q)_{\partial\Omega}. \end{aligned}$$

See Chapter 6 for a discussion on proper enforcement of boundary penalizations.

In order to apply this formulation to the examples in 4.4.2.1 and 4.4.2.2, we must determine the appropriate boundary conditions on r and p . We impose $r = 0$ on $\partial\Omega$. Due to the periodic nature of β and ω along $x = -\pi$ and $x = \pi$, σ is also periodic along

$x = -\pi$ and $x = \pi$. Since we have

$$\begin{aligned} (\boldsymbol{\sigma} \cdot \mathbf{t})(x, y) &= (\nabla r \cdot \mathbf{t})(x, y) - (\nabla^\perp p \cdot \mathbf{t})(x, y) \\ &= -(\nabla^\perp p \cdot \mathbf{t})(x, y) = -(\nabla p \cdot \mathbf{n})(x, y) \quad \forall (x, y) \in \partial\Omega \end{aligned}$$

where $\mathbf{t}$ is the unit counterclockwise tangent vector, we must have $\nabla p \cdot \mathbf{n}|_{x=-\pi} + \nabla p \cdot \mathbf{n}|_{x=\pi} = 0$ on $\partial\Omega$.

4.4.3.1 Benchmark Problem with Constant Load

Figure 4.14 shows the loss function (estimated energy error with respect to $\mathfrak{B}_{BF}$) per Galerkin Neural Network iteration as well as the L^2 error of the rotation $\boldsymbol{\beta}$ and shear stress $\boldsymbol{\sigma}$ for $t = 1, 10^{-2}, 10^{-4}, 10^{-6}$ in the case when $n = 0$. We denote by $\boldsymbol{\varphi}_i^{NN}$ the basis functions for approximating $\boldsymbol{\beta}$ and by q_i^{NN} the basis functions for approximating p . The hyperparameters for this example and all remaining examples are chosen as described at the beginning of Section 4.4 with the exception that the activation function for each basis function for $\boldsymbol{\beta}$ and p is $\sigma_i(z) = \tanh((1 + 0.55i)z)$. Figure 4.15-4.18 show the true errors and basis functions for $i = 1, 3, 5$. We observe no issues with locking.

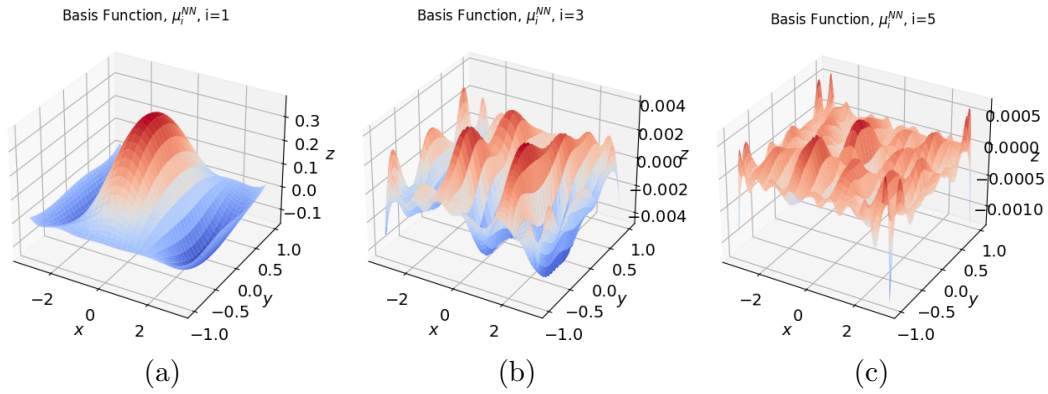


Figure 4.22: For $t = 10^{-2}$ in the problem with sinusoidal load with variational formulation on $\tilde{\mathfrak{B}}_{BF}, \tilde{\mathfrak{L}}_{BF}$: basis function μ_i^{NN} .

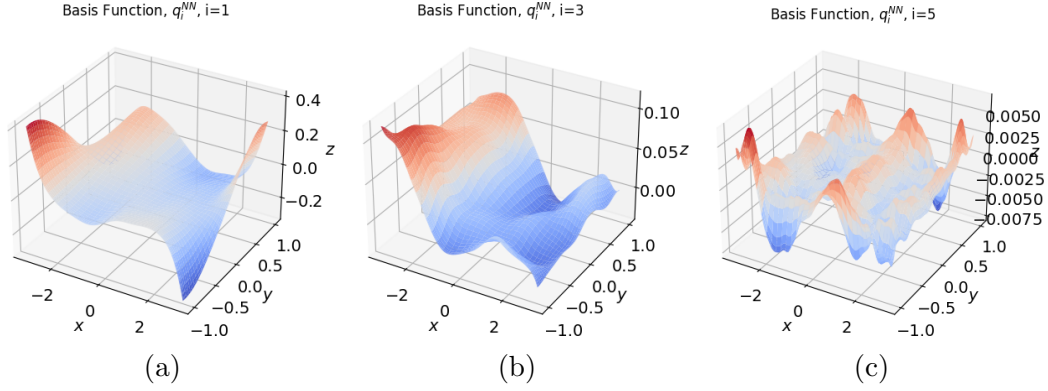


Figure 4.23: For $t = 10^{-2}$ in the problem with sinusoidal load with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: basis function q_i^{NN} .

4.4.3.2 Benchmark Problem with Sinusoidal Load and Boundary Layer

We next turn our attention to the model problem when $n = 1$. Figure 4.19 shows the loss function per Galerkin Neural Network iteration as well as the L^2 error of the rotation

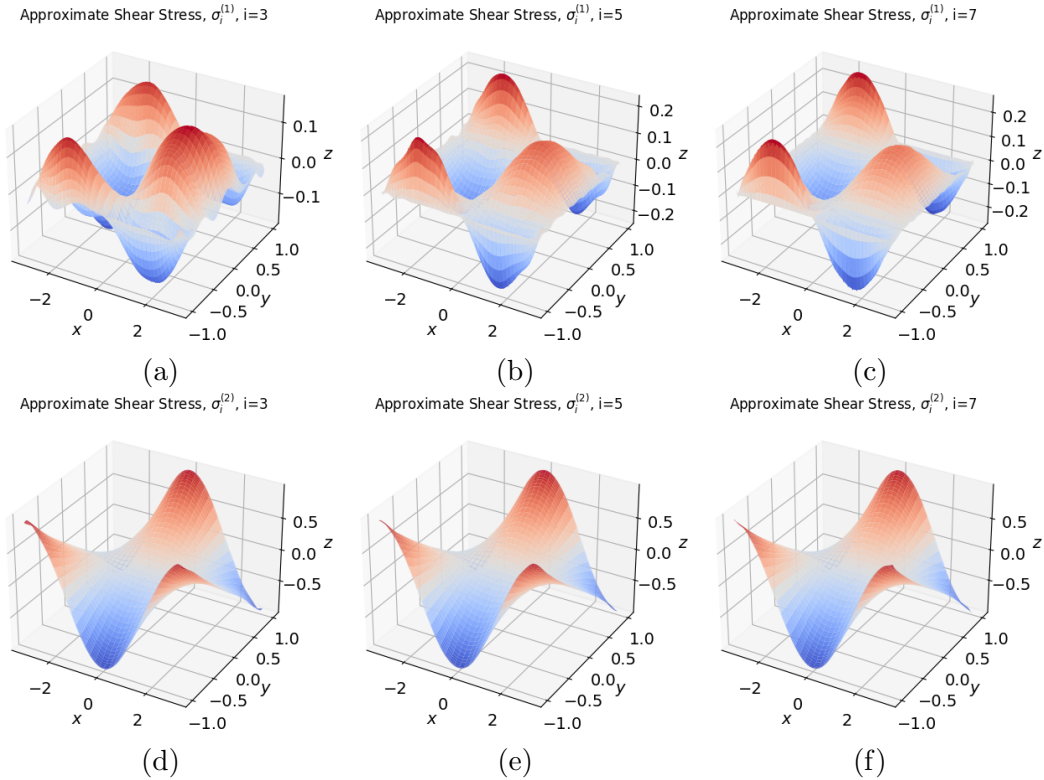


Figure 4.24: For $t = 10^{-2}$ in the problem with sinusoidal load with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: (a)-(c) Approximation $\sigma_i^{(1)} = \partial p_i / \partial y - \partial r_i / \partial x$. (d)-(f) Approximation $\sigma_i^{(2)} = -\partial p_i / \partial x - \partial r_i / \partial y$.

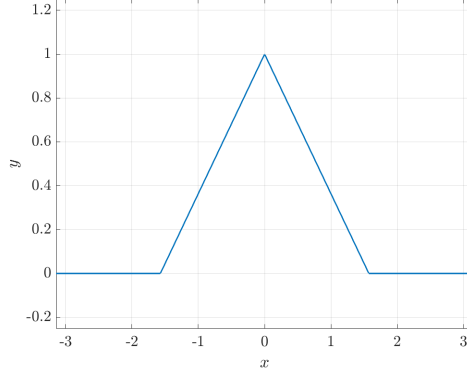


Figure 4.25: The triangular wave function g_T .

β and shear stress σ for $t = 1, 10^{-1}, 10^{-2}, 10^{-3}$ in the case when $n = 1$. Again, we observe no issues with locking. More importantly, we observe good resolution of the boundary layer as seen in Figure 4.24. Figures 4.20-4.23 show the true errors and basis functions for $i = 1, 3, 5$.

4.4.3.3 Triangular Wave Forcing Term

We next consider the case when the forcing term is the triangular wave shown in Figure 4.25. The domain and boundary conditions considered are the same as in 4.4.2.2. The triangular wave function has a Fourier series representation given by

$$g_T(x) = \frac{8}{\pi^2} \sum_{n=1}^{\infty} \frac{\sin^2(n\pi/4)}{n^2} \cos(nx),$$

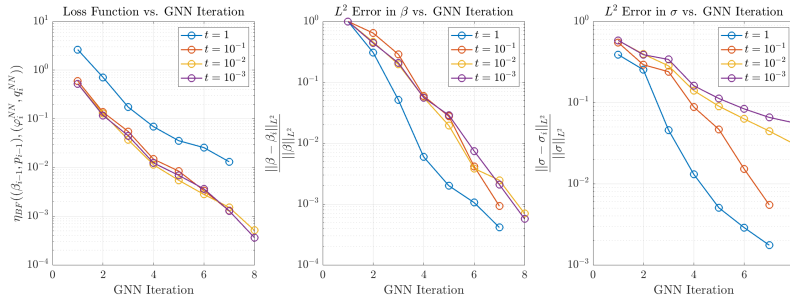


Figure 4.26: Energy error and L^2 error in β , ω , and σ for the problem with triangular wave load with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$.

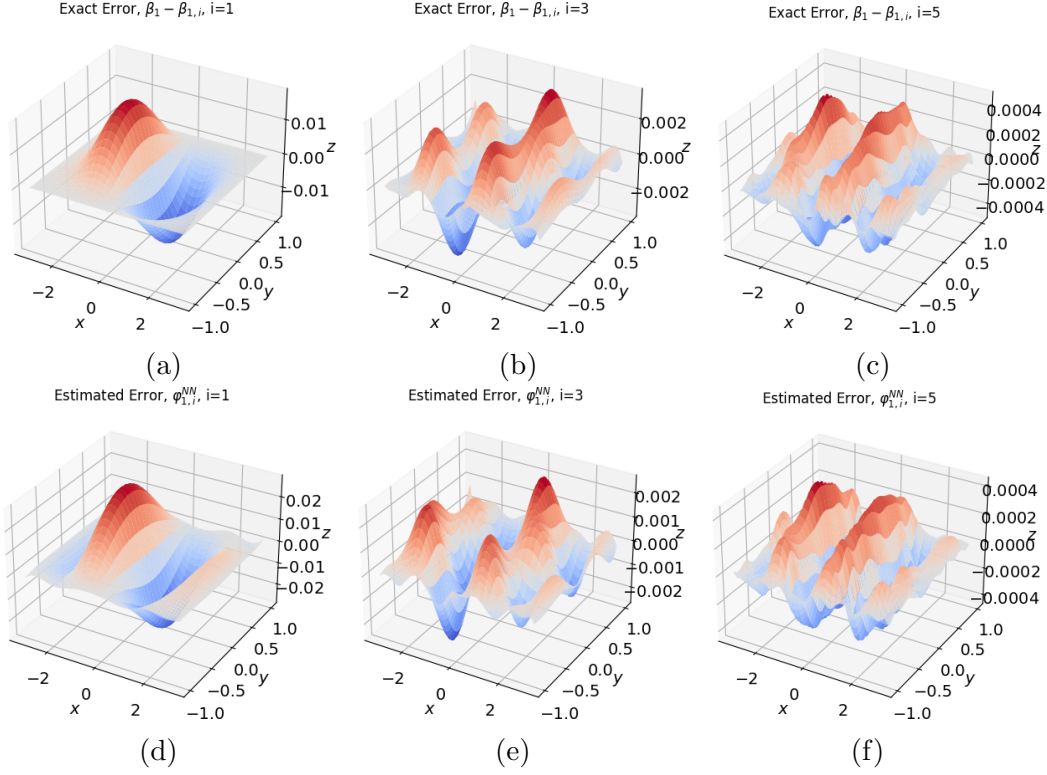


Figure 4.27: For $t = 10^{-6}$ in the triangular wave forcing term problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$ (a)-(c) True error $\beta^{(1)} - \beta_i^{(1)}$. (d)-(f) Basis function $\varphi_i^{(1),NN}$.

from which the exact solution may be obtained by superimposing the solutions corresponding to each n , as in Section 4.3.

Figure 4.26 shows the loss and L^2 error in β and σ while Figures 4.27-4.30 show the basis functions φ_i^{NN} , μ_i^{NN} , and q_i^{NN} as well as the approximations β_i and p_i . We again observe that the boundary layer is resolved correctly.

4.5 Conclusions

We have presented a neural network approach to approximating Reissner-Mindlin plates which is uniformly accurate in the plate thickness. The main contributions of this work are as follows. The neural network framework utilized is oblivious to the nature of the PDE and thus requires no structural modifications in order to be applied to the Reissner-Mindlin model other than a continuous, coercive, symmetric, positive-definite

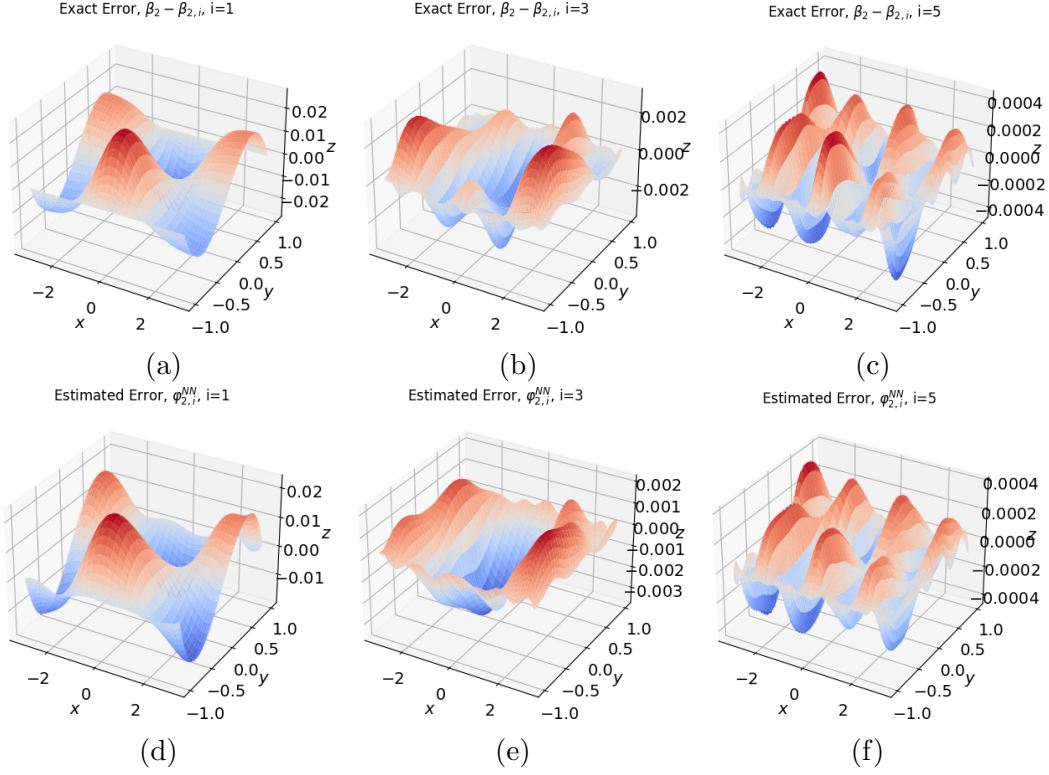


Figure 4.28: For $t = 10^{-6}$ in the triangular wave forcing term problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: (a)-(c) True error $\beta^{(2)} - \beta_i^{(2)}$. (d)-(f) Basis function $\varphi_i^{(2),NN}$.

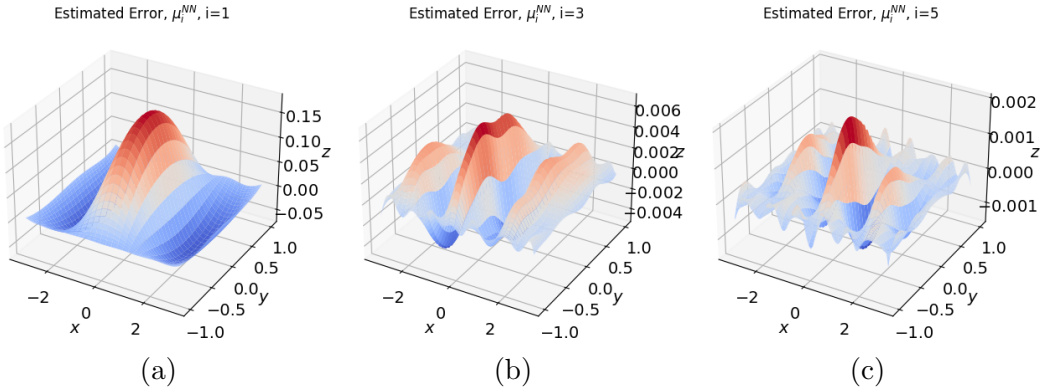


Figure 4.29: For $t = 10^{-2}$ in the triangular wave forcing term problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: basis function μ_i^{NN} .

bilinear operator. In presenting results for two new least squares variational formulations of the Reissner-Mindlin plate, we have demonstrated even for neural networks the importance of selecting a variational formulation which does not exhibit degenerate behavior as the plate thickness is reduced. Moreover, in comparison to traditional finite element methods, a reduction of the PDE to a first order system of least squares (FOSLS) is unnecessary

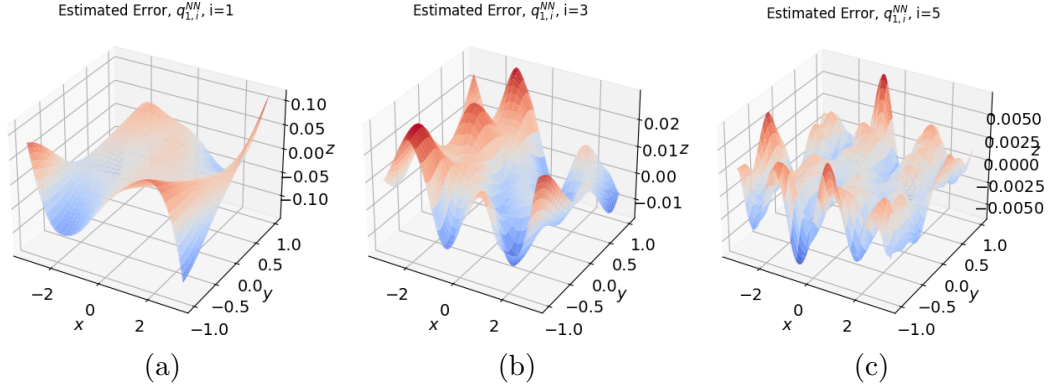


Figure 4.30: For $t = 10^{-2}$ in the triangular wave forcing term problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: basis function q_i^{NN} .

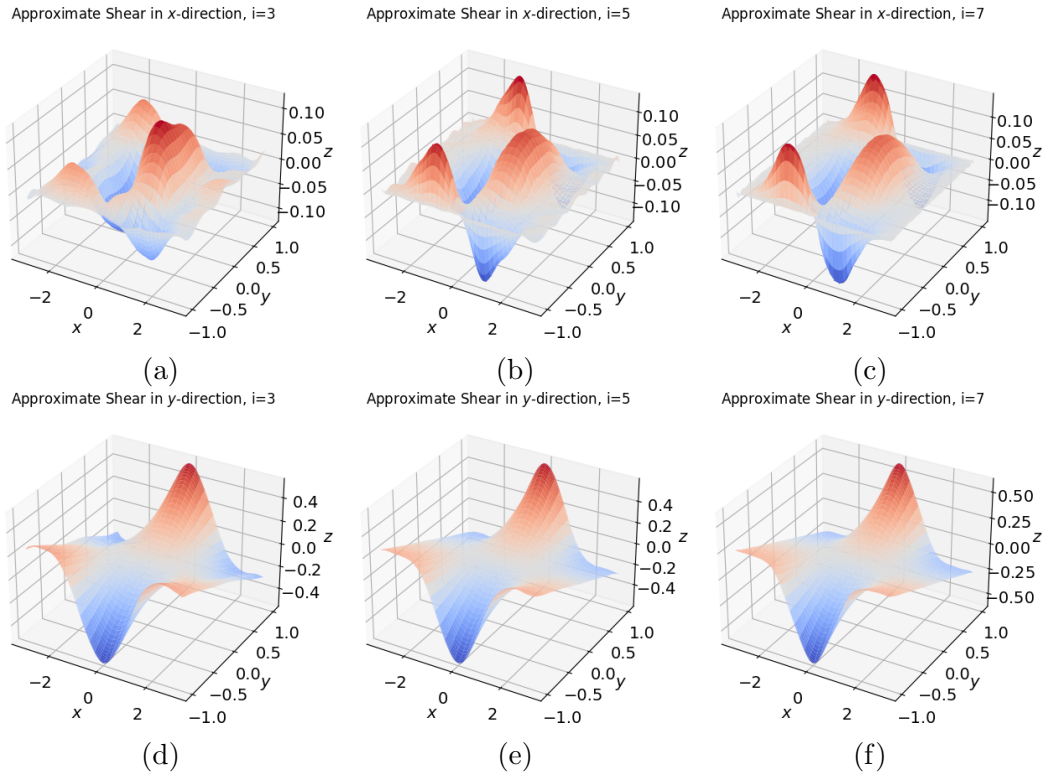


Figure 4.31: For $t = 10^{-2}$ in the triangular wave forcing term problem with variational formulation on $\tilde{\mathfrak{B}}_{BF}$, $\tilde{\mathfrak{L}}_{BF}$: (a)-(c) Approximation $\sigma_i^{(1)} = \partial p_i / \partial y - \partial r_i / \partial x$. (d)-(f) Approximation $\sigma_i^{(2)} = -\partial p_i / \partial x - \partial r_i / \partial y$.

with Galerkin neural networks as the activation functions are global without element continuity constraints and are only required to be elements of H^k , where k is the order of the PDE. Additionally, the accurate resolution boundary layer problems using finite element approaches typically requires the use of graded meshes around the boundary layer.

Our approach in comparison does not utilize any a priori knowledge of the location of the boundary layer. Finally, numerical results are provided for a complex benchmark problem which exhibits a boundary layer in the shear stress, and we also provide a framework for synthesizing a large class of test problems with analytic solutions which are crucial for evaluating performance of numerical methods – even beyond neural network approaches – applied to Reissner-Mindlin plates.

4.6 Acknowledgement

We thank the NSF (Grant No. 1644760) and the PhILMs Center (DE-SC0019453) for support.

4.7 Appendix

The coefficients $A_n(t)$, $B_n(t)$, $C_n(t)$, and $D_n(t)$ appearing in (4.13) are determined by the boundary conditions: $\beta_n = \mathbf{0}$ and $\omega_n = 0$ on Γ_D with periodic conditions on Γ_{per} . Namely, we require that $\beta_n(x, 1) = \mathbf{0}$ and $\omega_n(x, 1) = 0$ while the periodic boundary conditions on Γ_{per} are automatically satisfied thanks to the structure of β_n and ω_n . Additionally, satisfying the PDE $-\Delta\beta_n + t^{-2}(\beta_n - \nabla\omega_n) = \mathbf{0}$ gives rise to an additional constraint $\Upsilon_n''(y) - n^2\Upsilon_n(y, t) = \Psi_n(y, t) - 1$.

These requirements lead to a linear system of equations which determine the coefficients given by

$$\begin{cases} \lambda_n t \coth(\lambda_n) \cdot A_n(t) - n \cdot B_n(t) - n \cdot C_n(t) - n \cdot D_n(t) = -\frac{1}{n^3} \\ nt \cdot A_n(t) - (n \coth(n) + 1) \cdot B_n(t) - n \tanh(n) \cdot C_n(t) - n \tanh(n) \cdot D_n(t) = 0 \\ -B_n(t) - C_n(t) - (1 + t^2) \cdot D_n(t) = -\frac{t^2}{n^2} - \frac{1}{n^4} \\ 2n \cdot B_n(t) + \tanh(n) \cdot D_n(t) = 0 \end{cases} \quad (4.26)$$

The solution to (4.26) can be computed directly but has a rather complicated form. However, one can instead seek a series approximation valid for $t \ll 1$ of the form

$$\vec{\alpha}_n(t) = \vec{\gamma}_0 + \vec{\gamma}_1 t + \vec{\gamma}_2 \frac{t^2}{2} + \mathcal{O}(t^3),$$

where $\vec{\alpha}_n(t)$ is the vector of coefficients $\vec{\alpha}_n(t) = (A_n(t), B_n(t), C_n(t), D_n(t))^T$. Let $M_n(t)$ denote the coefficient matrix and $\vec{F}_n(t)$ the right-hand side of (4.26), respectively. Then expanding $M^{-1}(t)$ as a Taylor series about $t = 0$ yields the series expansion

$$\begin{aligned} \vec{\alpha}_n(t) = [M^{-1}(0) - M^{-1}(0)M'(0)M^{-1}(0)t + (-M^{-1}(0)M''(0)M^{-1}(0) + \\ 2M^{-1}(0)M'(0)M^{-1}(0)M'(0)M^{-1}(0))\frac{t^2}{2} + \mathcal{O}(t^3)]\vec{F}_n(t). \end{aligned}$$

The vectors $\vec{\gamma}_0$, $\vec{\gamma}_1$, and $\vec{\gamma}_2$ are thus given by

$$\begin{aligned}\vec{\gamma}_0 &= M^{-1}(0)\vec{F}_n(0) = \frac{\sinh(2n)}{n^4(2n + \sinh(2n))} \begin{bmatrix} 0 \\ -n \tanh(n) \\ n \coth(n) - 2n^2 + 1 \\ 2n^2 \end{bmatrix} \\ \vec{\gamma}_1 &= -M^{-1}(0)M'(0)M^{-1}(0)\vec{F}_n'(0) + M^{-1}(0)\vec{F}_n'(0) = \vec{0} \\ \vec{\gamma}_2 &= M^{-1}(0)\vec{F}_n''(0) - (M^{-1}(0)M''(0)M^{-1}(0) + 2M^{-1}(0)M'(0)M^{-1}(0)M'(0)M^{-1}(0))\vec{F}_n(0) \\ &= \begin{bmatrix} 2/n \\ 0 \\ 0 \\ 0 \end{bmatrix} + \frac{\sinh(2n)}{2n + \sinh(2n)} \begin{bmatrix} -4 \\ -2 \tanh(n)/n \\ 2(n \coth(n) - 2n^2 + 1)/n^2 \\ 4 \end{bmatrix} \\ &\quad + \left(\frac{\sinh(2n)}{2n + \sinh(2n)} \right)^2 \begin{bmatrix} 0 \\ 4 \tanh(n)/n \\ -4(n \coth(n) - 2n^2 + 1)/n^2 \\ -8 \end{bmatrix}.\end{aligned}$$

We note in particular that $A_n(t) = \mathcal{O}(t^2)$ and therefore, the factor $A_n(t)/t^2$ which appears in (4.16) remains bounded as $t \rightarrow 0$.

Moreover, Figure 4.32 shows plots of $A_n(t)$, $B_n(t)$, $C_n(t)$, and $D_n(t)$ versus t along with their approximations for $n = 1, 10, 20, 30$. It is clear that the series expansions are highly accurate for $t < 10^{-2}$. As n is increased, discrepancies between the coefficients and their series expansions appear to also increase for $t > 10^{-2}$. However, since the Fourier coefficients of β_n , ω_n , and σ_n generally decrease in magnitude as n increases, the net effect of these discrepancies is negligible. Indeed, Figure 4.33 shows the error in β , ω , and σ for the problem in Section 4.4.3.3 with $t = 10^{-2}$ when the exact coefficients are used compared to the series expansions. We thus use the simpler series expansions when

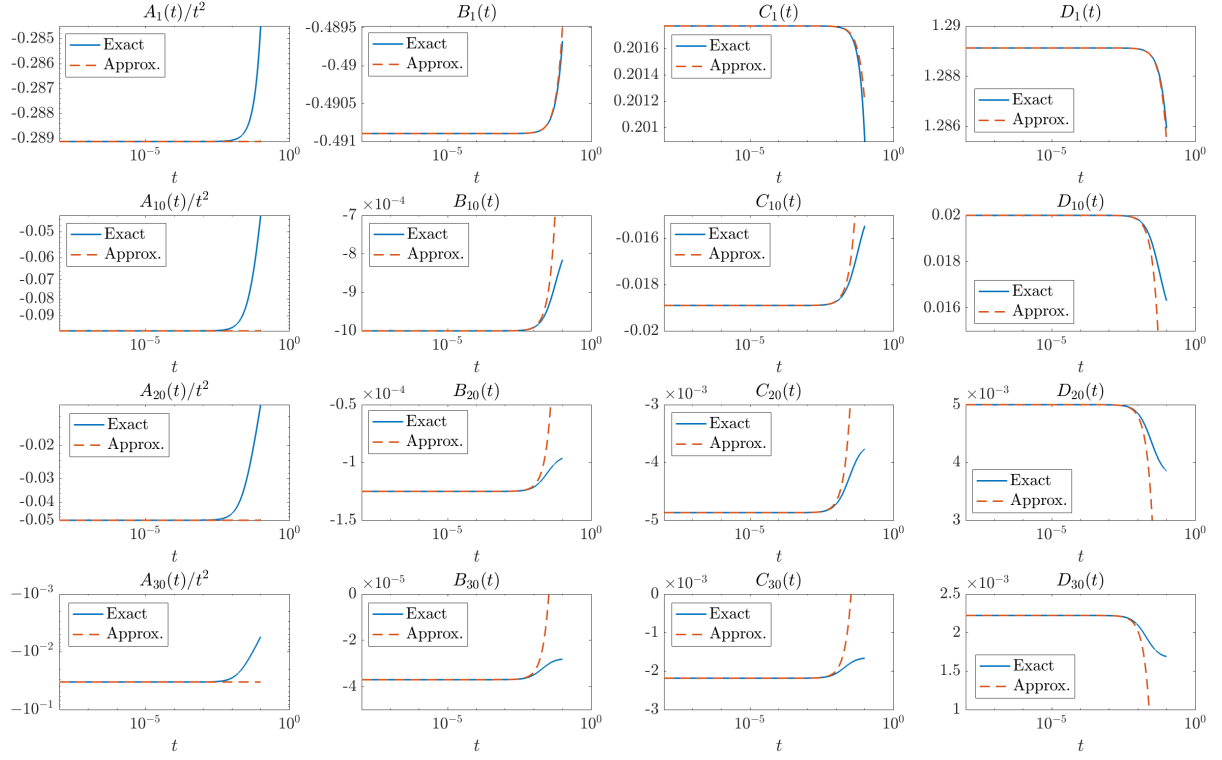


Figure 4.32: Coefficients $A_n(t)$, $B_n(t)$, $C_n(t)$, $D_n(t)$ and their series expansions for $n = 1, 10, 20, 30$.

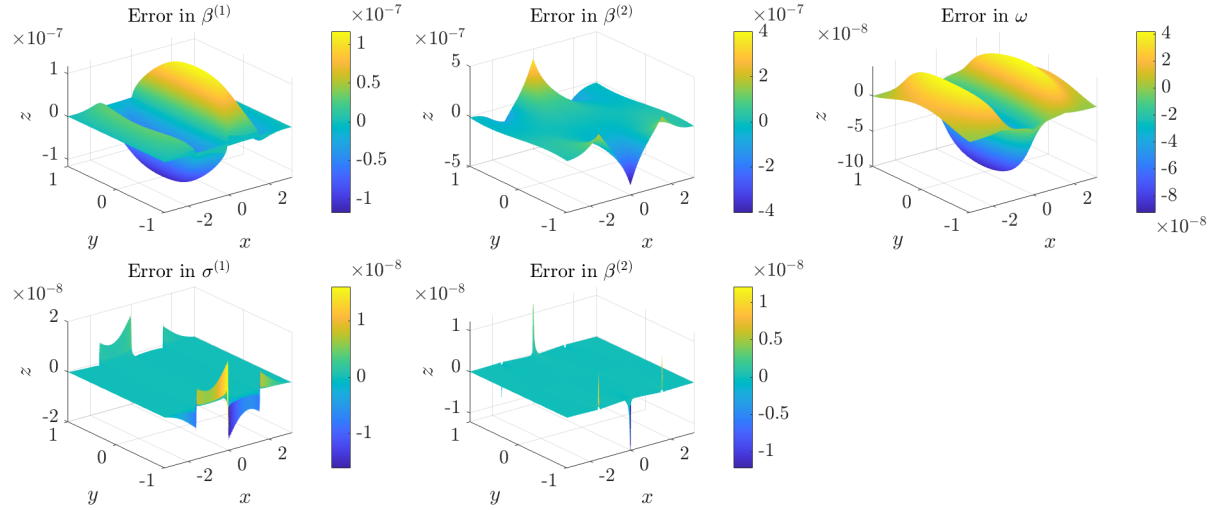


Figure 4.33: Errors in the true solution of the model problem in Section 4.4.3.3 with $t = 10^{-2}$ when using exact coefficients compared to their series expansions.

computing the exact solutions for all model problems.

Lastly, we note that the evaluation of quantities such as $\sinh(\lambda_n y)$ and $\cosh(\lambda_n)$ for small t in (4.13) requires the use of high-precision arithmetic libraries, such as [70] in

Python, in order to accurately evaluate the true solution.

CHAPTER 5

Application to Suspension Poiseuille Flows

The work in this chapter is part of a collaborative project in conjunction with my colleagues Amanda Howard and Panos Stinis (Pacific Northwest National Laboratory), Ravi Patel and Marta D’Elia (Sandia National Laboratory), and Martin Maxey (Brown University). This project seeks to apply machine learning techniques to learn stress closures in suspension flow models. The author’s contribution to this project was the analysis and application of Galerkin neural networks to learning particle stresses in these models and is summarized in this chapter. For full details of this interdisciplinary work, I refer the reader to [\[64\]](#).

5.1 Introduction

Suspension flows consist of suspended particles in a viscous fluid. Understanding the behavior of such particles is important in many applications, for instance red blood cells in blood, yeast in beer, and fiber suspensions in paper. We consider the simplest cases of spherical particles of uniform size – the so-called monodisperse suspension – and of

two distinct sizes – the so-called bidisperse suspension – in a shear flow. Early models of suspension flows include the phenomenological model of Leighton and Acrivos based on viscometric measurements of concentrated spherical particles in Newtonian fluids [78]; the diffusive flux model of Phillips, et. al. [104]; and the Suspension Balance Model (SBM) introduced by Nott and Brady [97] and refined in [92, 95] in which the normal stresses are assumed to be the principal drivers of particle migration.

We shall focus the majority of this chapter on the SBM and learning the stress closures required for the simplified normal stress setting of the model. Mathematically, the SBM is a first-order partial differential equation (PDE) which does not take into account individual particles but rather models the suspended particles as a continuous mass. In contrast to particle scale models proposed in [86, 132], this continuum scale model is not capable of reproducing all of the behavior observed in experiments. However, there is much interest among practitioners in analyzing the difference in particle migration between the SBM and particle scale models since the latter are often too computationally intractable when attempting to capture behavior such as segregation and migration in complex industrial flows. In contrast, the SBM is much less computationally intensive and thus more practical for many applications. Improvements to the SBM, especially in the case of bidisperse flows, are vital to ensuring that the use of these simpler models produces acceptable behavior.

Learning the stress closure in the SBM is critical to its accuracy and necessitates approximating the solution of the aforementioned PDE. In the monodisperse case, some previous work has been done to manually fit the stress closures in the SBM [95]. There are, to the best of our knowledge, very few works in the literature concerning the application of the SBM to bidisperse suspensions. Vollebregt, et. al. [124, 126] used an effective temperature route to find closures to the stress equations. Lyon and Leal made the assumption that the fluid and particles of smaller size behave as an effective suspending fluid for the larger particles [88], leading to poor agreement of their model

with experimental results.

In this work, we propose a simple coupled SBM for bidisperse suspension flows. To learn the stress closures for this model, there are a number of numerical techniques available from traditional methods such as finite elements [27], finite differences [82], and finite volumes [81] to more contemporary techniques such as those based on deep neural networks [42, 106, 134, 136]. We elect to approximate the PDE using neural networks for a number of reasons. First, even though the SBM is, from a mathematical perspective, a rather simple PDE, the data underlying the physical quantities of interest (volume fraction, shear stress, particle flux) are highly chaotic and contain multiscale features, thus making approximation of the SBM a challenging exercise even for traditional numerical methods, which would require very fine meshes/discretizations to adequately resolve; some recent results suggest that neural networks are capable of approximating such behavior without the use of (the analogue of) adaptive meshing or specially tailored activation functions [4, 85]. Second, neural networks are a relatively new technology in solving problems that have traditionally been the domain of numerical analysts and computational scientists. As such, many neural network approaches are demonstrated on straightforward model problems. It is both necessary and important to apply this technology to more realistic problems in order to identify key areas for improvement.

The key contributions of this work are the following:

- The presentation and analysis of a variational formulation for the SBM and its approximation using the Galerkin neural network framework. The learned stresses are compared with those produced by particle scale models and demonstrate that in the monodisperse case, the SBM stress closely matches the stress obtained from particle scale models.
- The proposal of a linearly coupled bidisperse SBM and a comparison of its performance with previously proposed uncoupled models.

The rest of this work is structured as follows. In Section 5.3, we present the mathematical formulation of the Suspension Balance Model for monodisperse suspensions and propose a simple coupled SBM for bidisperse suspensions. In Section 5.4, we describe the neural network framework used to approximate the SBM. Finally, we present numerical results for suspension flows in a channel in Section 5.5 and analyze the learned closures in relation to the results obtained from particle-scale models. Conclusions and a discussion of future work follow in Section 5.6.

5.2 The Force Coupling Method

In the subsequent sections, we will use data from particle-scale simulations as proxies for experimental data in order to validate the results obtained from the SBM. The particle-scale model we consider is the Force Coupling Method (FCM). We provide brief details here for context and refer the reader to [86, 132] for the complete description.

The equation of motions considered by the FCM is the incompressible Stokes flow

$$\begin{cases} \rho \frac{D\mathbf{u}}{Dt} - \eta_f \nabla^2 \mathbf{u} + \nabla p = \mathbf{f}(\mathbf{x}, t) \\ \nabla \cdot \mathbf{u} = 0, \end{cases} \quad (5.1)$$

where ρ is the fluid density, p is the pressure, and $\mathbf{f}$ is the contact force given by

$$f_i(\mathbf{x}, t) = \sum_{n=1}^{N_p} \left(F_i^n \Delta_M(\mathbf{x} - \mathbf{Y}^n) + G_{ij}^n \frac{\partial}{\partial x} \Delta_D(\mathbf{x} - \mathbf{Y}^n) \right),$$

with N_p being the number of particles in the suspension, F_i and G_{ij} being the force monopole and dipole moments, respectively, $\mathbf{Y}^n$ being the position of the n th particle, and Δ_M and Δ_D being the force envelopes described by Gaussian functions. For succinctness, we omit the definitions of Δ_M and Δ_D .

Once the fluid velocity $\mathbf{u}$ is known, we obtain the velocity of the n th particle at time t ,

denoted by $\mathbf{V}^n(t)$, by averaging over the region occupied by the particle:

$$\mathbf{V}^n(t) = \int_{\Omega_n} \mathbf{u}(\mathbf{x}, t) \Delta_M(\mathbf{x} - \mathbf{Y}^n) dV. \quad (5.2)$$

Similar post-processing techniques are used to obtain the particle stress and flux. We shall denote the stress and flux obtained by the FCM by σ_{22}^{FCM} and $j_{\perp}^{FCM}$, respectively.

5.3 The Suspension Balance Model

5.3.1 Monodisperse Suspensions

A detailed presentation of the SBM may be found in [95, 97]; a summary is presented here. We consider a monodisperse suspension of spherical particles of radius r in a fluid with viscosity η_f in a channel Ω . The volume fraction ϕ of particles at a given point y and time t evolves according to

$$\frac{\partial \phi(t, y)}{\partial t} + \mathbf{u} \cdot \nabla \phi = \nabla \cdot \mathbf{j}_{\perp}, \quad (5.3)$$

where $\mathbf{u}$ is the average suspension velocity and $\perp$ is the particle flux normal to the walls of the channel given by

$$\mathbf{j}_{\perp} = \frac{2r^2}{9\eta_f} f(\phi) \nabla \cdot \Sigma_p. \quad (5.4)$$

In (5.4), $f : \mathbb{R} \rightarrow \mathbb{R}$ is the hindrance function which measures how much other particles in the suspension impede movement at a particular point in the channel. In the monodisperse case, the hindrance function we use is given by Richardson and Zaki [108]:

$$f(\phi) = (1 - \phi)^{\alpha}. \quad (5.5)$$

The value of α is taken to be 5 as in [23] although we shall also present results for other

values of α . The flux, and therefore the particle migration, is driven by the particle stress Σ_p . Closures exist for the particle stress [95] expressed as a function of the volume fraction. However, in this work, we are interested in learning the particle stress in (5.4).

We shall assume for simplicity that the particle stress Σ_p can be approximated by the σ_{22} component of the stress tensor, leading to the PDE

$$\begin{cases} \frac{2r^2}{9\eta_f} f(\phi(y, \gamma)) \frac{\partial \sigma_{22}(y, \gamma)}{\partial y} = j_{\perp}(y, \gamma) & \text{in } (0, H_{\gamma}) \times \Omega \\ \sigma_{22}(y_0, \gamma) = g(\gamma) & \text{on } [0, H_{\gamma}] \times \{y_0\}. \end{cases} \quad (5.6)$$

Here, $\Omega = (\epsilon, H_y - \epsilon)$ is the one-dimensional spatial domain. We use the nondimensional strain unit $\gamma = t \cdot H_y / (2\mathbf{u})$ for time – with H_y the channel height, H_{γ} the maximum strain unit, and ϵ a small perturbation away from the channel walls. In experiments, monodisperse suspensions exhibit a layering of particles at the walls [54, 58, 116, 133]. This behavior is known to not be captured by the SBM and thus, the purpose of ϵ is to exclude the walls of the channel from the computational domain. The boundary is denoted by $\Gamma = \{y_0\} \times (0, H_{\gamma})$ while the function $g(\gamma)$ serves as a boundary condition.

In order for (5.6) to be well-defined, we provide the particle flux $j_{\perp}$ generated by the Force Coupling Method (FCM) [86, 132], which may be taken to be a proxy for experimental flux values. The FCM is a particle scale model in which the fluid is characterized by Stokes flow that takes into account particle interactions in a localized body force (see Section 5.2 for more details). We are particularly interested in comparing the continuum scale behavior of the SBM with the particle scale behavior of the FCM. The stress learned by solving (5.6) is then compared to the stress computed from the FCM.

5.3.2 Bidisperse Suspensions

We consider a bidisperse suspension with particle radii r_a and r_b with size ratio described by $\lambda = r_b/r_a$. We propose extending Morris and Boulay's suspension balance model for monodisperse suspensions to bidisperse suspensions using a simple linear coupling of the two phases. Namely, we consider the system of PDEs:

$$\begin{cases} \frac{2r_a^2}{9\eta_f} f_a(\phi_a, \phi_b) \left(\frac{\partial \sigma_{22,a}}{\partial y} + \varepsilon_{ab} \frac{\partial \sigma_{22,b}}{\partial y} \right) = j_{\perp,a} & \text{in } (0, H_\gamma) \times \Omega \\ \frac{2r_b^2}{9\eta_f} f_b(\phi_a, \phi_b) \left(\varepsilon_{ba} \frac{\partial \sigma_{22,a}}{\partial y} + \frac{\partial \sigma_{22,b}}{\partial y} \right) = j_{\perp,b} & \text{in } (0, H_\gamma) \times \Omega \\ \sigma_{22,a}(y_0, \gamma) = g_a(\gamma), \sigma_{22,b}(y_0, \gamma) = g_b(\gamma) & \text{on } [0, H_\gamma] \times \{y_0\}, \end{cases} \quad (5.7)$$

where ϕ_i , $\sigma_{22,i}$, and $j_{\perp,i}$ are the volume fractions, shear stresses, and fluxes of each phase for $i \in \{a, b\}$. We again assume that particle phase stresses can be approximated by $\sigma_{22,i}$. The coupling of the phases is controlled by the parameters $0 \leq \varepsilon_{ab}, \varepsilon_{ba} \leq 1$. When $\varepsilon_{ab} = \varepsilon_{ba} = 0$, (5.7) reduces to a decoupled system in which the flux of each particle phase is driven only by the respective particle stress of that phase.

We utilize two different hindrance functions f_i in the bidisperse case. The first, proposed by Shauly and colleagues [112], is given by

$$f_i(\phi_a, \phi_b) = \frac{1 - \phi_i}{\eta_s}, \quad (5.8)$$

where η_s is the shear viscosity given by

$$\eta_s(\phi) = \left(1 - \frac{\phi}{\phi_{\text{bi}}^c} \right)^{-2} \quad (5.9)$$

with critical volume fraction ϕ_{bi}^c given by

$$\phi_{\text{bi}}^c = \phi_{\text{mono}}^c \left(1 + c \left| \frac{r_b - r_a}{r_b + r_a} \right|^{3/2} \left(\frac{\phi_b}{\phi_b + \phi_a} \right)^{3/2} \left(\frac{\phi_a}{\phi_b + \phi_a} \right) \right). \quad (5.10)$$

The critical volume fraction in the monodisperse case is taken to be $\phi_{\text{mono}}^c = 0.59$ and the parameter $c \approx 1.5$. The second hindrance function we consider, proposed by Davis and Gecol [37], is given by the more complex relation

$$f_i(\phi_a, \phi_b) = (1 - \phi_a - \phi_b)^{-S_{ii}} \left(1 + \sum_{j \neq i} (S_{ij} - S_{ii}) \phi_j \right), \quad (5.11)$$

where S_{ij} are the sedimentation coefficients.

5.4 Galerkin Neural Network Formulation of the Problem

To approximate (5.6) and (5.7), we utilize Galerkin neural networks introduced in Chapter 3 [3] which approximate a given symmetric positive-definite variational formulation of the underlying PDE. In this section, we present the variational formulations utilized and demonstrate that they are well-posed.

5.4.1 The Monodisperse Case

The variational formulation associated with (5.6) is

$$\begin{aligned} \sigma_{22} \in L^2(0, H_\gamma; H^1(\Omega)) : & \quad (F(\phi) \frac{\partial \sigma_{22}}{\partial y}, F(\phi) \frac{\partial \tau}{\partial y})_{L^2(0, H_\gamma; L^2(\Omega))} + C(\sigma_{22}(y_0, \cdot), \tau(y_0, \cdot))_{L^2(0, H_\gamma)} \\ & = (j_\perp, F(\phi) \frac{\partial \tau}{\partial y})_{L^2(0, H_\gamma; L^2(\Omega))} + C(g, \tau(y_0, \cdot))_{L^2(0, H_\gamma)} \quad \forall \tau \in L^2(0, H_\gamma; H^1(\Omega)), \end{aligned} \quad (5.12)$$

where, for ease of notation, we set $F(\phi) := 2af(\phi)/(9\eta_f)$. The space $L^2(0, H_\gamma; H^1(\Omega))$ is the anisotropic Sobolev space [2] with norm defined by

$$\|u\|_{L^2(0, H_\gamma; H^1(\Omega_y))} := \left(\int_0^{H_\gamma} \|u(\gamma)\|_{H^1(\Omega)}^2 d\gamma \right)^{1/2},$$

where $u(\gamma) : (0, H_\gamma) \rightarrow H^1(\Omega)$ is the function $u(t) = u(\cdot, \gamma)$. We define the operators $a_{MD} : L^2(0, H_\gamma; H^1(\Omega)) \times L^2(0, H_\gamma; H^1(\Omega)) \rightarrow \mathbb{R}$ and $L_{MD} : L^2(0, H_\gamma; H^1(\Omega)) \rightarrow \mathbb{R}$ by

$$\begin{aligned} a_{MD}(\sigma, \tau) &:= (F(\phi) \frac{\partial \sigma}{\partial y}, F(\phi) \frac{\partial \tau}{\partial y})_{L^2(0, H_\gamma; L^2(\Omega))} + C(\sigma(y_0, \cdot), \tau(y_0, \cdot))_{L^2(0, H_\gamma)} \\ L_{MD}(\tau) &:= (j_\perp, F(\phi) \frac{\partial \tau}{\partial y})_{L^2(0, H_\gamma; L^2(\Omega))} + C(g, \tau(y_0, \cdot))_{L^2(0, H_\gamma)}. \end{aligned}$$

This formulation is well-posed as demonstrated in the following result.

Theorem 5.4.1. *There exist constants $C_1, C_2 > 0$ such that*

$$a_{MD}(\sigma, \tau) \leq C_1 \|\sigma\|_{L^2(0, H_\gamma; L^2(\Omega))} \|\tau\|_{L^2(0, H_\gamma; L^2(\Omega))}, \quad C_2 \|\tau\|_{L^2(0, H_\gamma; L^2(\Omega))}^2 \leq a_{MD}(\tau, \tau)$$

for all $\sigma, \tau \in L^2(0, H_\gamma; H^1(\Omega))$.

Proof. The first result is a direct consequence of the Cauchy-Schwarz inequality and a one-dimensional trace inequality:

$$\begin{aligned} a_{MD}(\sigma, \tau) &\leq \|F(\phi)\sigma\|_{L^2(0, H_\gamma; L^2(\Omega))} \|F(\phi)\tau\|_{L^2(0, H_\gamma; L^2(\Omega))} + C \|\sigma(y_0, \cdot)\|_{L^2(0, H_\gamma)} \|\tau(y_0, \cdot)\|_{L^2(0, H_\gamma)} \\ &\leq C_1 \|\sigma\|_{L^2(0, H_\gamma; L^2(\Omega))} \|\tau\|_{L^2(0, H_\gamma; L^2(\Omega))}. \end{aligned}$$

The constant C_1 is given by $C_1 = \max\{\|F(\phi)\|_{L^\infty(0, H_\gamma; L^\infty(\Omega))}^2, 2/|\Omega|\}$. The trace inequality $\|\tau(y_0, \cdot)\|_{L^2(0, H_\gamma)} \leq \tilde{C} \|\tau\|_{L^2(0, H_\gamma; L^2(\Omega))}$ can be easily shown from the relation

$$\tau(y_0, \gamma) = - \int_{y_0}^y \frac{\partial \tau(s, \gamma)}{\partial s} ds + \tau(y, \gamma). \quad (5.13)$$

Squaring both sides and integrating over y and γ , we obtain

$$\begin{aligned}\tau(y_0, \gamma)^2 &\leq 2 \int_{y_0}^y \left| \frac{\partial \tau(s, \gamma)}{\partial s} \right|^2 ds + 2\tau(y, \gamma)^2 \\ |\Omega| \cdot \|\tau(y_0, \cdot)\|_{L^2(0, H_\gamma)}^2 &\leq 2 \left\| \frac{\partial \tau}{\partial y} \right\|_{L^2(0, H_\gamma; L^2(y_0, y))}^2 + 2\|\tau\|_{L^2(0, H_\gamma; L^2(\Omega))}^2 \\ &\leq 2 \left\| \frac{\partial \tau}{\partial y} \right\|_{L^2(0, H_\gamma; L^2(\Omega))}^2 + 2\|\tau\|_{L^2(0, H_\gamma; L^2(\Omega))}^2,\end{aligned}$$

The second result is due to a partial Poincare inequality on $H^1(\Omega)$. Starting from (5.13) again, we obtain

$$\|\tau\|_{L^2(0, H_\gamma; L^2(\Omega))}^2 \leq 2 \left\| \frac{\partial \tau}{\partial y} \right\|_{L^2(0, H_\gamma; L^2(\Omega))}^2 + 2|\Omega| \cdot \|\tau(y_0, \cdot)\|_{L^2(0, H_\gamma)}^2, \quad (5.14)$$

which implies

$$\begin{aligned}\|\tau\|_{L^2(0, H_\gamma; H^1(\Omega))}^2 &\leq 2 \left\| \frac{\partial \tau}{\partial y} \right\|_{L^2(0, H_\gamma; L^2(\Omega))}^2 + 2|\Omega| \cdot \|\tau(y_0, \cdot)\|_{L^2(0, H_\gamma)}^2 \\ &\leq C_2 \left(\left\| F(\phi) \frac{\partial \tau}{\partial y} \right\|_{L^2(0, H_\gamma; L^2(\Omega))}^2 + \|\tau(y_0, \cdot)\|_{L^2(0, H_\gamma)}^2 \right),\end{aligned}$$

where $C_2 = 2 \max\{\|1/F(\phi)^2\|_{L^\infty(0, H_\gamma; L^\infty(\Omega))}, |\Omega|\}$. We note that $F(\phi) \in (0, 1)$ and is bounded away from 0 so that C does not blow up. $\square$

We represent the solution σ_{22} as a linear combination of hierarchical basis functions $\{\Sigma_i^{NN}\}_{i=1}^n$, each of which is the output of a shallow neural network:

$$\sigma_{22}(y, \gamma) \approx \sum_{i=1}^n c_i \Sigma_i^{NN}(y, \gamma). \quad (5.15)$$

The volume fraction ϕ is learned in a pretraining step based on data from the FCM.

Given $\phi^{FCM} \in L^2(0, H_\gamma; L^2(\Omega))$, the simple formulation for learning an approximation

$$\phi(y, \gamma) \approx \sum_{i=1}^n c_i \varphi_i^{NN}(y, \gamma) \quad (5.16)$$

is

$$\begin{aligned} \phi \in L^2(0, H_\gamma; L^2(\Omega)) : (\phi, \psi)_{L^2(0, H_\gamma; L^2(\Omega))} &= (\phi^{FCM}, \psi)_{L^2(0, H_\gamma; L^2(\Omega))} \\ \forall \psi \in L^2(0, H_\gamma; L^2(\Omega)). \end{aligned} \quad (5.17)$$

5.4.2 The Bidisperse Case

We present the analogous analysis to Section (5.4.1) here. Let $\mathcal{X} = \mathbf{L}^2(0, H_\gamma; \mathbf{H}^1(\Omega))$ endowed with the norm $\|\cdot\|_{\mathcal{X}}$ satisfying

$$\|(\tau_a, \tau_b)\|_{\mathcal{X}}^2 = \|\tau_a\|_{L^2(0, H_\gamma; H^1(\Omega))}^2 + \|\tau_b\|_{L^2(0, H_\gamma; H^1(\Omega))}^2.$$

The variational formulation associated with (5.7) is to seek $(\sigma_{22,a}, \sigma_{22,b}) \in \mathcal{X}$ such that

$$\begin{aligned} & (F_a(\phi_a, \phi_b) \left(\frac{\partial \sigma_{22,a}}{\partial a} + \varepsilon_{ab} \frac{\partial \sigma_{22,b}}{\partial y} \right), F_a(\phi_a, \phi_b) \left(\frac{\partial \tau_a}{\partial a} + \varepsilon_{ab} \frac{\partial \tau_b}{\partial y} \right))_{L^2(0, H_\gamma; L^2(\Omega))} \\ & + (F_b(\phi_a, \phi_b) \left(+\varepsilon_{ba} \frac{\partial \sigma_{22,a}}{\partial a} + \frac{\partial \sigma_{22,b}}{\partial y} \right), F_b(\phi_a, \phi_b) \left(\varepsilon_{ba} \frac{\partial \tau_a}{\partial a} + \frac{\partial \tau_b}{\partial y} \right))_{L^2(0, H_\gamma; L^2(\Omega))} \\ & + C(\sigma_{22,a}(y_0, \cdot), \tau_a(y_0, \cdot))_{L^2(0, H_\gamma)} + C(\sigma_{22,b}(y_0, \cdot), \tau_b(y_0, \cdot))_{L^2(0, H_\gamma)} = \\ & (j_{\perp,a}, F_a(\phi_a, \phi_b) \left(\frac{\partial \tau_a}{\partial a} + \varepsilon_{ab} \frac{\partial \tau_b}{\partial y} \right))_{L^2(0, H_\gamma; L^2(\Omega))} \\ & + (j_{\perp,b}, F_b(\phi_a, \phi_b) \left(\varepsilon_{ba} \frac{\partial \tau_b}{\partial a} + \frac{\partial \tau_b}{\partial y} \right))_{L^2(0, H_\gamma; L^2(\Omega))} \\ & + C(g_a, \tau_a(y_0, \cdot))_{L^2(0, H_\gamma)} + C(g_b, \tau_b(y_0, \cdot))_{L^2(0, H_\gamma)} \end{aligned} \quad (5.18)$$

for all $(\tau_a, \tau_b) \in \mathcal{X}$, where F_a and F_b are defined by

$$\begin{aligned} F_a(\phi_a, \phi_b) &= \frac{2r_a^2}{9\eta_f} f_a(\phi_a, \phi_b) \\ F_b(\phi_a, \phi_b) &= \frac{2r_b^2}{9\eta_f} f_b(\phi_a, \phi_b). \end{aligned}$$

For ease of notation, we define the operators $a_{BD} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ and $L_{BD} : \mathcal{X} \rightarrow \mathbb{R}$ by

$$\begin{aligned} a_{BD}(\boldsymbol{\sigma}, \boldsymbol{\tau}) &:= (F_a(\phi_a, \phi_b) \left(\frac{\partial \sigma_a}{\partial a} + \varepsilon_{ab} \frac{\partial \sigma_b}{\partial y} \right), F_a(\phi_a, \phi_b) \left(\frac{\partial \tau_a}{\partial a} + \varepsilon_{ab} \frac{\partial \tau_b}{\partial y} \right))_{L^2(0, H_\gamma; L^2(\Omega))} \\ &\quad + (F_b(\phi_a, \phi_b) \left(+\varepsilon_{ba} \frac{\partial \sigma_a}{\partial a} + \frac{\partial \sigma_b}{\partial y} \right), F_b(\phi_a, \phi_b) \left(\varepsilon_{ba} \frac{\partial \tau_a}{\partial a} + \frac{\partial \tau_b}{\partial y} \right))_{L^2(0, H_\gamma; L^2(\Omega))} \\ &\quad + C(\sigma_a(y_0, \cdot), \tau_a(y_0, \cdot))_{L^2(0, H_\gamma)} + C(\sigma_b(y_0, \cdot), \tau_b(y_0, \cdot))_{L^2(0, H_\gamma)} \\ L_{BD}(\boldsymbol{\tau}) &:= (j_{\perp, a}, F_a(\phi_a, \phi_b) \left(\frac{\partial \tau_a}{\partial a} + \varepsilon_{ab} \frac{\partial \tau_b}{\partial y} \right))_{L^2(0; H_\gamma; L^2(\Omega))} \\ &\quad + (j_{\perp, b}, F_b(\phi_a, \phi_b) \left(\varepsilon_{ba} \frac{\partial \tau_b}{\partial a} + \frac{\partial \tau_b}{\partial y} \right))_{L^2(0; H_\gamma; L^2(\Omega))}. \end{aligned}$$

The following result demonstrates the well-posedness of (5.18).

Theorem 5.4.2. *Suppose $0 < \varepsilon_{ab}, \varepsilon_{ba} < 1/2$. There exist constants $C_1, C_2 > 0$ such that*

$$a_{BD}(\boldsymbol{\sigma}, \boldsymbol{\tau}) \leq C_1 \|\boldsymbol{\sigma}\|_{\mathcal{X}} \|\boldsymbol{\tau}\|_{\mathcal{X}}, \quad C_2 \|\boldsymbol{\tau}\|_{\mathcal{X}}^2 \leq a_{BD}(\boldsymbol{\tau}, \boldsymbol{\tau})$$

for all $\boldsymbol{\sigma}, \boldsymbol{\tau} \in \mathcal{X}$.

Proof. The first inequality follows a similar strategy as in the monodisperse case. We apply the Cauchy-Schwarz inequality to all inner products followed by the trace inequality

to $\|\sigma_a(y_0, \cdot)\|_{L^2(0, H_\gamma)}$ and related terms, which yields

$$\begin{aligned}
a_{BD}(\boldsymbol{\sigma}, \boldsymbol{\tau}) &\leq 2\|F_a(\phi_a, \phi_b)\|_{L^\infty(0, H_\gamma; L^\infty(\Omega))}^2 \|(\sigma_a, \sigma_b)\|_{\mathcal{X}} \|(\tau_a, \tau_b)\|_{\mathcal{X}} \\
&\quad + 2\|F_b(\phi_a, \phi_b)\|_{L^\infty(0, H_\gamma; L^\infty(\Omega))}^2 \|(\sigma_a, \sigma_b)\|_{\mathcal{X}} \|(\tau_a, \tau_b)\|_{\mathcal{X}} \\
&\quad + \frac{2}{|\Omega|} \|\sigma_a\|_{L^2(0, H_\gamma; H^1(\Omega))} \|\tau_a\|_{L^2(0, H_\gamma; H^1(\Omega))} \\
&\quad + \frac{2}{|\Omega|} \|\sigma_b\|_{L^2(0, H_\gamma; H^1(\Omega))} \|\tau_b\|_{L^2(0, H_\gamma; H^1(\Omega))} \\
&\leq C_1 \|(\sigma_a, \sigma_b)\|_{\mathcal{X}} \|(\tau_a, \tau_b)\|_{\mathcal{X}},
\end{aligned}$$

where $C_1 = 2 \max\{\|F_a(\phi_a, \phi_b)\|_{L^\infty(0, H_\gamma; L^\infty(\Omega))}^2, \|F_b(\phi_a, \phi_b)\|_{L^\infty(0, H_\gamma; L^\infty(\Omega))}^2, 1/|\Omega|\}$.

The second inequality follows by applying (5.14) to τ_a and τ_b . We demonstrate the application to τ_a first:

$$\begin{aligned}
\|\tau_a\|_{L^2(0; H_\gamma; H^1(\Omega))}^2 &\leq 2 \left\| \frac{\partial \tau_a}{\partial y} \right\|_{L^2(0, H_\gamma; L^2(\Omega))}^2 + 2|\Omega| \cdot \|\tau_a(y_0, \cdot)\|_{L^2(0, H_\gamma)}^2 \\
&\leq 4 \left\| \frac{\partial \tau_a}{\partial y} + \varepsilon_{ab} \frac{\partial \tau_b}{\partial y} \right\|_{L^2(0, H_\gamma; L^2(\Omega))}^2 + 4\varepsilon_{ab}^2 \left\| \frac{\partial \tau_b}{\partial y} \right\|_{L^2(0, H_\gamma; L^2(\Omega))}^2 \\
&\quad + 2|\Omega| \cdot \|\tau_a(y_0, \cdot)\|_{L^2(0, H_\gamma)}^2 \\
&\leq C_{0,a} \left(\left\| F_a(\phi_a, \phi_b) \left(\frac{\partial \tau_a}{\partial y} + \varepsilon_{ab} \frac{\partial \tau_b}{\partial y} \right) \right\|_{L^2(0, H_\gamma; L^2(\Omega))}^2 + \|\tau_a(y_0, \cdot)\|_{L^2(0, H_\gamma)}^2 \right) \\
&\quad + 4\varepsilon_{ab}^2 \left\| \frac{\partial \tau_b}{\partial y} \right\|_{L^2(0, H_\gamma; L^2(\Omega))}^2,
\end{aligned}$$

where $C_{0,a} := \max\{4\|1/F_a(\phi_a, \phi_b)\|_{L^\infty(0, H_\gamma; L^\infty(\Omega))}^2, 2|\Omega|\}$. Similarly,

$$\begin{aligned}
\|\tau_b\|_{L^2(0; H_\gamma; H^1(\Omega))}^2 &\leq C_{0,b} \left(\left\| F_b(\phi_a, \phi_b) \left(\varepsilon_{ba} \frac{\partial \tau_a}{\partial y} + \frac{\partial \tau_b}{\partial y} \right) \right\|_{L^2(0, H_\gamma; L^2(\Omega))}^2 + \|\tau_b(y_0, \cdot)\|_{L^2(0, H_\gamma)}^2 \right) \\
&\quad + 4\varepsilon_{ba}^2 \left\| \frac{\partial \tau_a}{\partial y} \right\|_{L^2(0, H_\gamma; L^2(\Omega))}^2,
\end{aligned}$$

where $C_{0,b} := \max\{4\|1/F_b(\phi_a, \phi_b)\|_{L^\infty(0, H_\gamma; L^\infty(\Omega))}^2, 2|\Omega|\}$. Altogether, provided $0 < \varepsilon_{ab}, \varepsilon_{ba} <$

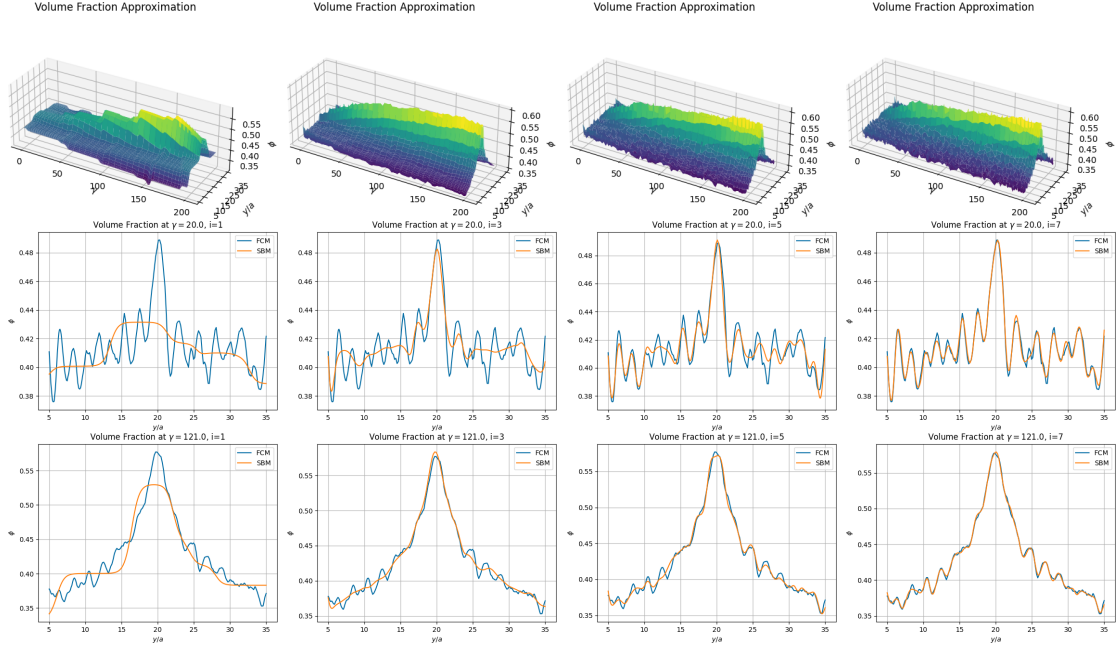


Figure 5.1: Top row: from left to right, the Galerkin Neural Network approximation to ϕ using 1, 3, 5, and 7 basis functions. Middle row: left to right, one-dimensional volume fraction profiles at $\gamma = 20$ using 1, 3, 5, and 7 basis functions. Bottom row: left to right, one-dimensional volume fraction profiles at $\gamma = 121$ using 1, 3, 5, and 7 basis functions.

1/2, we have

$$(1 - 4\varepsilon_{ba}^2) \|\tau_a\|_{L^2(0; H_\gamma; H^1(\Omega))}^2 + (1 - 4\varepsilon_{ab}^2) \|\tau_b\|_{L^2(0; H_\gamma; H^1(\Omega))}^2 \leq \max\{C_{0,a}, C_{0,b}\} a_{BD}(\boldsymbol{\tau}, \boldsymbol{\tau}).$$

Thus, the second claim holds with $C_2 := \min\{1 - 4\varepsilon_{ab}^2, 1 - 4\varepsilon_{ba}^2\} / \max\{C_{0,a}, C_{0,b}\}$. $\square$

5.5 Numerical Examples

5.5.1 Monodisperse Suspensions

We consider a channel of height $H_y = 40$ with truncation $\epsilon = 5$ over 200 strain units ($H_\gamma = 200$). The suspension flow consists of spherical particles of radius $r = 1$ and fluid viscosity $\eta_f = 1$. The initial volume fraction is $\phi = 0.40$ across the channel.

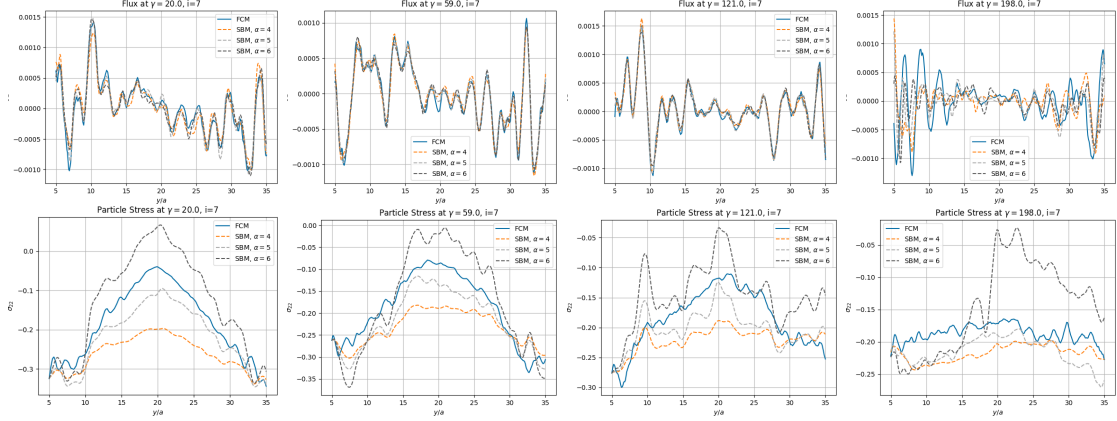


Figure 5.2: Top row: left to right, one-dimensional flux profiles at $\gamma = 20, 59, 121, 198$ using 7 basis functions. Bottom row: left to right, one-dimensional stress profiles at $\gamma = 20, 59, 121, 198$ using 7 basis functions.

5.5.1.1 Pretraining the Volume Fraction

The volume fraction is fitted with a neural network using seven Galerkin Neural Network basis functions. The i th basis function is the realization of a feedforward neural network consisting of two hidden layers and $20 \times 2^{i-1}$ neurons per layer and the training data consists of 128×256 Gauss-Legendre quadrature nodes [36] in the γy Cartesian plane.

The space-time surface plot of the learned volume fraction using 1, 3, 5, and 7 basis functions is shown in Figure 5.1 alongside the one-dimensional profiles at select times. We observe that after seven basis functions, the volume fraction is learned to within 0.5% relative error. Additionally, we observe the traditional hierarchical behavior of the basis functions – initially, the Galerkin Neural Network approximation is diffusive and captures only the large spike in the center of the channel; later, the basis functions begin to capture increasingly fine scale behavior.

5.5.1.2 Learning the Stress Closure

The particle stress σ_{22} is approximated using seven basis functions in the Galerkin Neural Network method with the same architecture as in Section 5.5.1.1. We display the one-dimensional flux and stress profiles using 7 basis functions for a range of γ and also

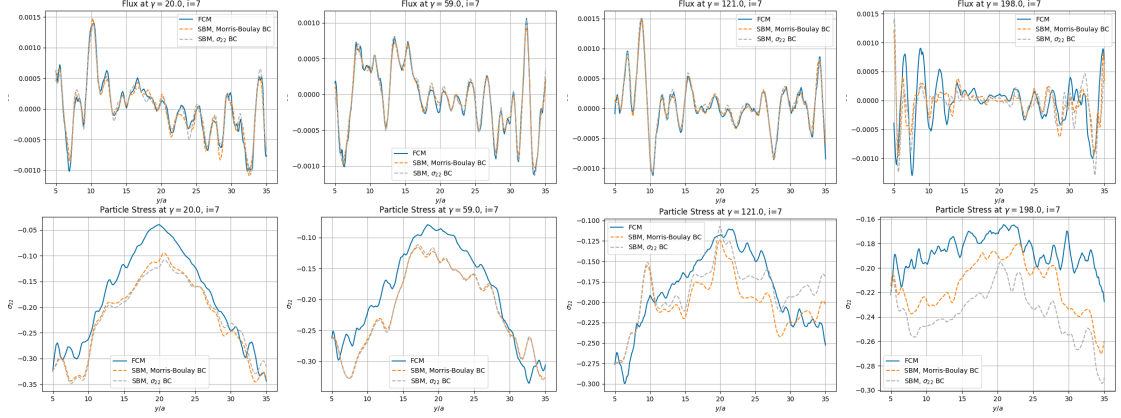


Figure 5.3: Top row: left to right, one-dimensional flux profiles at $\gamma = 20, 59, 121, 198$ using 7 basis functions. Bottom row: left to right, one-dimensional stress profiles at $\gamma = 20, 59, 121, 198$ using 7 basis functions.

consider $\alpha = 4, 5, 6$ in the Richardson-Zaki hindrance function 5.5.

We consider two choices of the boundary condition data. The first is to use the boundary values of the FCM stress σ_{22}^{FCM} . The second is to use the suspension pressure relation in [95]:

$$\sigma_{22}(y_0, \gamma) = \mu_n(\phi) \left(\frac{1 + \lambda_2 + \lambda_3}{3} \right) \dot{\gamma}_L, \quad \mu_n(\phi) = \eta_f K_n \left(\frac{\phi}{\phi_m} \right)^2 \left(1 - \frac{\phi}{\phi_m} \right)^{-2}. \quad (5.19)$$

Here, $K_n = 0.75$, $\phi_m = 0.585$, $\lambda_2 = 0.8$, and $\lambda_3 = 0.5$; $\dot{\gamma}_L$ is the shear rate

$$\dot{\gamma}_L = 2 \left(1 - \frac{y}{H_y} \right) \frac{u_c}{H_y} \quad (5.20)$$

We initially use the Morris and Boulay pressure relation (5.19) for the boundary condition data g . The learned stress in (5.6) is used to compute the particle flux and compared with the flux obtained from the FCM in Figure 5.2 (top row). We observe that for early and mid times in the suspension flow, the computed particle flux is within 1% relative error compared to the FCM flux. For late times ($\gamma = 198$, the discrepancies between the computed particle flux and FCM flux is larger. For all times, the value of α in the hindrance function does not appear to impact the computed SBM flux. However,

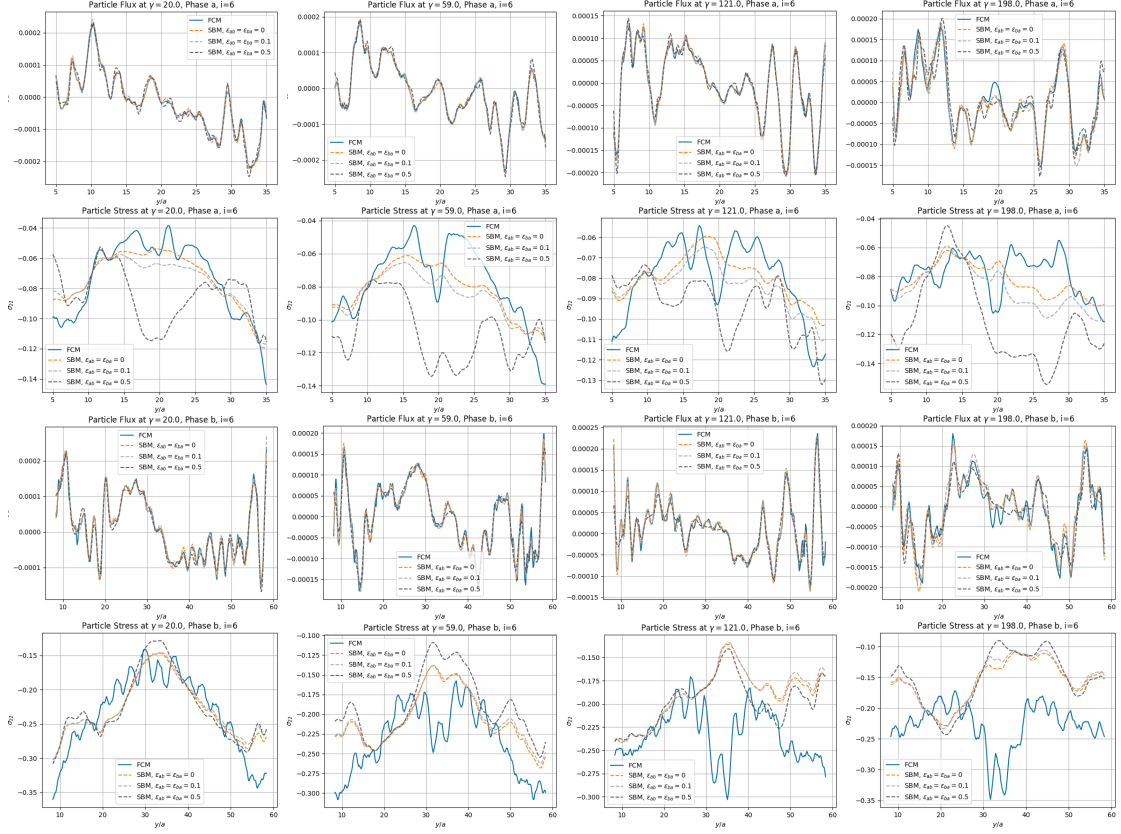


Figure 5.4: Top two rows: left to right, one-dimensional flux and stress profiles for phase a (large particles) at $\gamma = 20, 59, 121, 198$ using 6 basis functions. Bottom two rows: left to right, one-dimensional flux and stress profiles for phase b (small particles) at $\gamma = 20, 59, 121, 198$ using 6 basis functions. The Shauly hindrance function is used for both phases.

when comparing the learned stresses with the FCM stress, it is clear that $\alpha = 5$ produces the most accurate results in line with [23].

We also compare the results when using the Morris and Boulay boundary condition compared with the boundary condition from the FCM data σ_{22}^{FCM} . For early and mid times, the differences between the two boundary conditions are negligible. For late times ($\gamma = 198$), both boundary conditions do not produce accurate results, although the Morris and Boulay boundary condition produces a slightly more accurate result.

5.5.2 Bidisperse Suspensions

We again consider a channel of height $H_y = 40$ with truncation $\epsilon = 5$ over 200 strain units ($H_\gamma = 200$). The suspension flow consists of spherical particles of radius $r_a = 1$ and $r_b = 0.6$ and fluid viscosity $\eta_f = 1$. The initial volume fractions are $\phi_a = 0.10$ and $\phi_b = 0.30$ across the channel. The fluxes, stresses, and volume fractions in the bidisperse case are significantly noisier than in the monodisperse case. We average the FCM data over time in the following way: given data field $\chi_i(y) := \chi(\gamma_i, y)$, the averaged field $\bar{\chi}$ is defined over the stencil $\{i - k, \dots, i, \dots, i + k\}$ by

$$\bar{\chi}_i(y) := \frac{\sum_{n=-k}^k \chi_{i+n}(y) \omega_i(\gamma_{i+n})}{\sum_{n=-k}^k \omega_i(\gamma_{i+n})} \quad (5.21)$$

where ω_i is a weighting function such that $\omega_i(\gamma_i) = 1$ and $\omega_i(\gamma_{i-k}) = \omega(\gamma_{i+k}) = 0.1$ which is designed to give more weight to time data closer to $\chi_i(y)$. For all of the examples in this section, we take $k = 10$.

We shall first consider the case of a symmetric coupling ($\varepsilon_{ab} = \varepsilon_{ba} \in \{0.1, 0.5\}$) compared with the uncoupled case ($\varepsilon_{ab} = \varepsilon_{ba} = 0$). We apply the Galerkin Neural Network method with six basis functions for both $\sigma_{22,a}$ and $\sigma_{22,b}$. The i th basis function for each phase is the realization of a feedforward neural network consisting of two hidden layers and $20 \times 2^{i-1}$ neurons per layer and the training data consists of 128×256 Gauss-Legendre quadrature nodes. The one-dimensional flux and stress profiles for a variety of times are given in Figure 5.4 for the Shauly hindrance function. The analogous results for the Davis-Gecol hindrance function are given in Figure 5.5.

We observe that for phase a , the learned stresses generally match the FCM stress throughout the channel except at steady state, although for $\varepsilon_{ab} = \varepsilon_{ba} = 0.5$, the learned stresses are less accurate. It is interesting to note that in this case that the associated

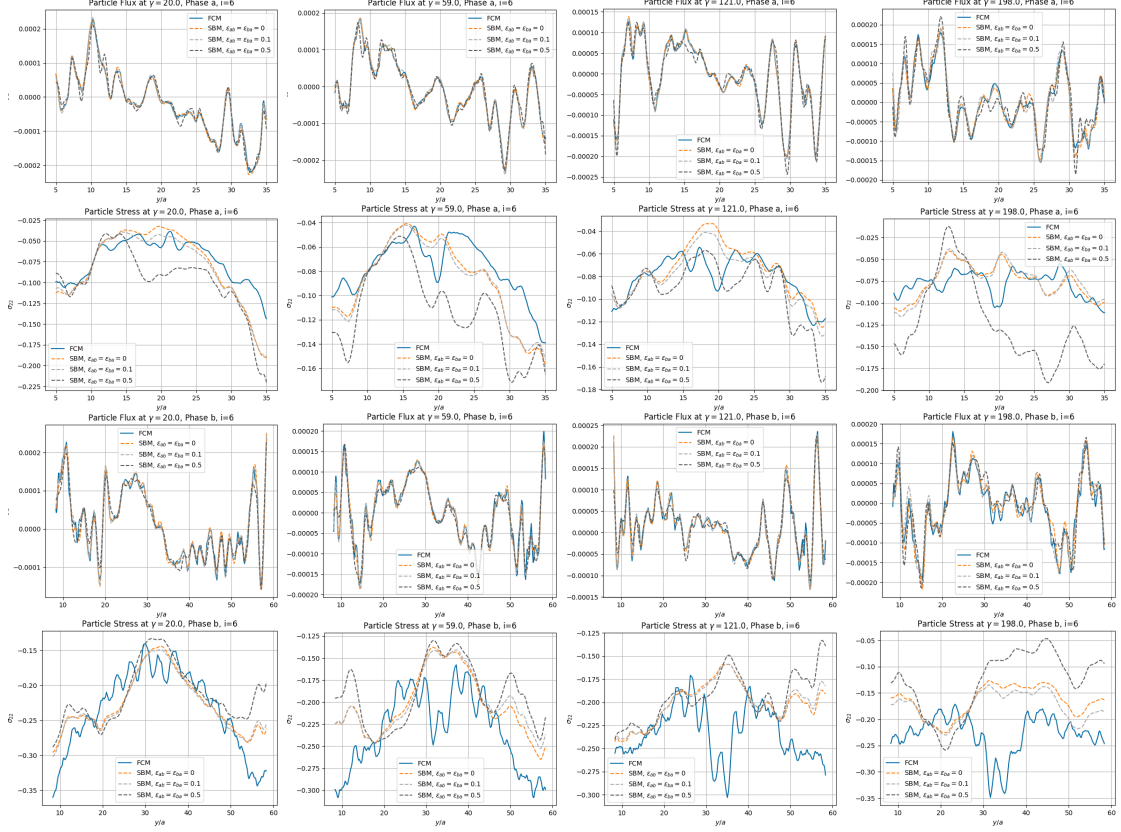


Figure 5.5: Top two rows: left to right, one-dimensional flux and stress profiles for phase a (large particles) at $\gamma = 20, 59, 121, 198$ using 6 basis functions. Bottom two rows: left to right, one-dimensional flux and stress profiles for phase b (small particles) at $\gamma = 20, 59, 121, 198$ using 6 basis functions. The Davis-Gecol hindrance function is used for both phases.

bilinear operator loses coercivity (Theorem 5.4.2). For phase b , the learned stress is more accurate away from the channel walls, although in both phases, the sudden drop in stress which is captured in the FCM is not properly captured in the SBM stresses. It appears that both the uncoupled bidisperse model and the weakly coupled model provide similar results. For stronger couplings, the learned stresses deteriorate.

5.6 Conclusions

We have presented a neural network method for learning stress closures in the Suspension Balance Model for both monodisperse and bidisperse suspensions. The monodisperse case utilizes the refined model presented by Morris and Boulay in [95] and demonstrates

that, aside from the particle layering near the channel walls, the SBM is capable of capturing the key features of the particle scale behavior in the FCM such as the larger particle stresses in the center of the channel. Is it interesting to note that Galerkin neural networks is capable of representing noisy data accurately, particularly the monodisperse volume fraction. In the bidisperse case, we have presented a linearly coupled model. Our results show that the weakly coupled case yields particle stresses which generally capture the parabolic nature of the FCM stresses but are not capable of capturing the sudden drop in the stress at the center of the channel. This suggests that improvements to the coupling in the bidisperse model are necessary, but nevertheless the presentation of the linearly coupled is an important advancement in improving continuum scale modeling of suspension flows.

An interesting continuation of this work would be to modify the coupling in the bidisperse SBM model using spatially dependent coupling parameters which take into account known migration and segregation phenomena throughout the channel. The proposed linear coupling in the present work is simple and does not properly capture all of the features of the FCM stress. Additionally, an inverse problem in which the coupling which satisfies the FCM stress data is learned would allow for a much more exhaustive exploration of the bidisperse model. Such more for general PDEs has been explored in [\[109\]](#).

5.7 Acknowledgments

The author would like to thank ORISE for their financial support in undertaking this work which was carried out as part of an NSF internship at Pacific Northwest National Laboratory (PNNL) as well as PNNL for the use of their high-performance computing resources. The author would also like to thank Amanda Howard for her invaluable mentorship and technical expertise as well as Panos Stinis for his mentorship and allowing me to attend his group meetings which exposed me to a wider range of work at PNNL.

CHAPTER 6

Extension to non-self-adjoint and indefinite problems

6.1 Introduction

Much progress has been made in the last decade developing numerical methods for approximating partial differential equations (PDEs) using neural networks. These methods are diverse and span a wide range of applications including approximating forward problems [42, 77, 106, 113, 136] and inverse problems [100, 106], learning solution operators [83, 87] of PDEs, and even complementing existing numerical methods, e.g., by learning finite difference stencils [17] or guiding adaptive mesh refinement [129].

The authors’s past work (see Chapter 3) on Galerkin neural networks considers an approach inspired by Galerkin methods for approximating variational forms of PDEs. An adaptive subspace is constructed using shallow neural networks to learn basis functions based on Riesz representations of a sequence of weak residuals [3]. In this past work, which focused on the restricted setting of self-adjoint positive-definite problems, rigorous theory was established demonstrating convergence of the method and a number of attractive properties such as the loss function naturally inducing an a posteriori estimator for the

energy error. Applications to a singularly-perturbed problem with boundary layers was presented in [4].

In this chapter, we describe an extension of Galerkin neural networks to non-self-adjoint and/or indefinite problems of the general form

$$\begin{cases} \mathcal{L}[u] = f & \text{in } \Omega \\ \mathcal{B}[u] = g & \text{on } \partial\Omega, \end{cases} \quad (6.1)$$

where $\mathcal{L}$ and $\mathcal{B}$ are linear differential operators, based on least squares variational principles. The most common approach in the literature for approximating (6.1) in a least squares sense is to minimize the loss function based on the residual

$$u \mapsto \|f - \mathcal{L}[u]\|_{L^2(\Omega)}^2 + \|g - \mathcal{B}[u]\|_{L^2(\partial\Omega)}^2, \quad (6.2)$$

or variations thereof replacing the L^2 norms with discrete ℓ^2 norms or a mean-squared error. And yet, there are several concerns with this approach that we aim to reveal and address in this work.

1. When the minimization of (6.2) is formulated as a variational problem, the norms must be chosen to ensure that the underlying bilinear operator is both continuous and coercive with respect to the solution space. Most frequently, this is not achieved when the residuals are taken in the L^2 norm, thus resulting in a formulation which is not well-posed.
2. In many applications, such as computational fluid dynamics and solid mechanics, the solution of (6.1) contains singular features even when the data is smooth. Traditional numerical methods such as finite element methods typically capture this behavior through the use of techniques such as adaptive mesh refinement and hp -refinement. One might expect neural networks to offer an interesting alternative to these problems

due to their nonlinear approximation properties. Indeed, one can view the realization of ReLU deep neural networks (DNNs) as equivalent to continuous piecewise linear finite element methods whose underlying mesh is allowed to change as the nonlinear parameters of the DNN are learned [59]. However, when utilizing least squares type formulations such as (6.2) for problems with low-regularity solutions, caution must be exercised when $\mathcal{L}[u] \notin L^2(\Omega)$ and/or $\mathcal{B}[u] \notin L^2(\partial\Omega)$.

Each of these issues underscores the need to choose an appropriate formulation of the PDE; one cannot simply apply neural networks indiscriminately and expect to obtain satisfactory results for arbitrary problems.

Moreover, DNNs attempt to learn solutions to complex physical phenomena starting from a random initial state. In particular, the weights of the network are usually initialized according to a probability distribution in order to prevent activations from saturating [52]. However, some of the most challenging problems have well-known solution structures even if an analytic solution is unknown. These structures are exploited in a number of numerical methods which supplement existing approaches with so-called knowledge-based functions. Examples include extended finite element methods [121] and generalized finite element methods [120]. We note also the recent work in [14] which utilizes specific asymptotic structures in the neural network architecture in order to be more effectively learn boundary layers in relatively straightforward problems. We shall demonstrate that Galerkin neural networks are capable of effectively incorporating such knowledge-based functions to approximate singular features in the solution.

The main contributions of the present work are the following:

- The presentation and analysis of a new least squares variational formulation which may be used easily in conjunction with Galerkin neural networks to approximate PDEs of the general form (6.1). We discuss appropriate function spaces for problems with singular data, namely weighted Sobolev spaces.

- A procedure for incorporating knowledge-based functions into the Galerkin neural network framework. We present two approaches: (i) knowledge-based subspaces which may be used to augment the basis functions learn by Galerkin neural networks, and (ii) a simple modified neural network architecture which allows for supplemental activation functions based on known solution structures. We demonstrate how to choose these knowledge-based functions for several different problems.

Numerical results are presented for several challenging benchmark problems using Galerkin neural networks in order to demonstrate the efficacy of the proposed method.

The rest of this chapter is structured as follows. In Section 6.2.1, we summarize the basic Galerkin neural network procedure and the relevant underlying theory to the work in the chapter. In Section 6.2.2, we present and analyze a least squares variational formulation for a very general class of PDE followed by a description of knowledge-based functions and subspaces and their incorporation into the Galerkin neural network procedure in 6.2.4. We describe in Section 6.3 how the preceding theory can be applied to several interesting applications, including Stokes flow in polygonal domains. Conclusions follow in Section 6.4.

6.2 Galerkin neural networks

6.2.1 The basic framework

For completeness, we summarize the basic Galerkin neural network framework here before describing the proposed modifications. We consider the prototypical variational problem

$$u \in X : a(u, v) = L(v) \quad \forall v \in X, \quad (6.3)$$

where a is symmetric positive-definite, continuous, and coercive with respect to $(X, \|\cdot\|)$.

$\|_X$), and L is continuous. Given an initial approximation $u_0 \in X$ to (6.3), the goal of the Galerkin neural network is to construct a finite-dimensional subspace $S_j := \text{span}\{u_0, \varphi_1^{NN}, \dots, \varphi_j^{NN}\}$ such that the basis function φ_i^{NN} satisfies

$$\varphi_i^{NN} = \arg \max_{v \in V_n^\sigma \cap B} \langle r(u_{i-1}), v \rangle, \quad (6.4)$$

where $r(u_{i-1})$ is the weak residual of (3.4) with respect to u_{i-1} whose duality pairing on X is given by $\langle r(u_{i-1}), v \rangle = L(v) - a(u_{i-1}, v)$, B is the closed unit ball in X , V_n^σ is the set

$$V_n^\sigma := \left\{ v : v(x) = \sum_{i=1}^n c_i \sigma(x \cdot W_i + b_i), W \in \mathbb{R}^{d \times n}, b \in \mathbb{R}^{1 \times n}, c \in \mathbb{R}^{n \times 1} \right\}. \quad (6.5)$$

The standard Galerkin neural network approach is to generate an approximation u_i by solving the finite-dimensional variational problem

$$u_i \in S_i : a(u, v) = L(v) \quad \forall v \in S_i. \quad (6.6)$$

One can show that the basis function φ_i^{NN} is in fact an approximation to the Riesz representation $\varphi_i := (u - u_{i-1}) / |||u - u_{i-1}|||$ of the weak residual $r(u_{i-1})$, and thus the subspace S_j may be viewed as augmenting the initial approximation u_0 with a sequence of increasingly finer-scale error correction terms. The approximation u_i is exponentially convergent in the number of basis functions generated, and one can also show that the objective function (i.e., weak residual) is an a posteriori estimator of the energy error $|||u - u_{i-1}|||$. These properties are summarized in the following theorem.

Theorem 6.2.1. [3] *Given $u_0 \in (X, ||| \cdot |||)$, for each $\varepsilon_i \in (0, 1)$ there exists $n_i \in \mathbb{N}$ and $\varphi_i^{NN} \in V_{n_i}^\sigma$ as defined by (6.4) such that*

$$|||\varphi_i - \varphi_i^{NN}||| \leq \frac{2\varepsilon_i}{1 - \varepsilon_i}. \quad (6.7)$$

Moreover, the approximation u_i given in (4.4) satisfies the estimate

$$|||u - u_i||| \leq |||u - u_0||| \cdot \prod_{j=1}^i \min \left\{ 1, \frac{2\varepsilon_j}{1 - \varepsilon_j} \right\} \quad (6.8)$$

and, if $\varepsilon_i < 1/3$, then each φ_i^{NN} satisfies

$$\frac{1 - \varepsilon_i}{1 + \varepsilon_i} \langle r(u_{i-1}), \varphi_i^{NN} \rangle \leq |||u - u_{i-1}||| \leq \frac{1 - \varepsilon_i}{1 - 3\varepsilon_i} \langle r(u_{i-1}), \varphi_i^{NN} \rangle. \quad (6.9)$$

Using (6.9), one can generate an approximation u_i such that $|||u - u_{i-1}||| \lesssim \text{tol}$ for a given tolerance tol by checking whether $\langle r(u_{i-1}), \varphi_i^{NN} \rangle < \text{tol}$ after φ_i^{NN} is generated. In the affirmative case, we assume that the energy error is within the desired tolerance and terminate the subspace generation procedure. Otherwise, we generate another basis function until the desired tolerance is reached.

The training procedure for learning φ_i^{NN} consists of a standard gradient-based step to update the hidden parameters W and b in conjunction with the least squares solution of the linear system

$$\begin{cases} Ac &= F \\ A_{jk} &= a(\sigma(x \cdot W_j + b_j), \sigma(x \cdot W_k + b_k)) \\ F_j &= L(\sigma(x \cdot W_j + b_j)) - a(u_{i-1}, \sigma(x \cdot W_k + b_k)) \end{cases} \quad (6.10)$$

in order to update the activation coefficients c . The linear system in (6.10) corresponds to the orthogonal projection of the error $u - u_{i-1}$ onto the subspace $\Phi := \text{span}\{\sigma(x \cdot W_j + b_j)\}_{j=1}^n$. That is, given a parameter initialization $(W^{(0)}, b^{(0)}, c^{(0)})$, we update the

parameters by the rules

$$W^{(m+1)} = W^{(m)} + \nabla_{W^{(m)}} \left[\frac{\langle r(u_{i-1}), v \rangle}{|||v|||} \right] \quad (6.11)$$

$$b^{(m+1)} = b^{(m)} + \nabla_{b^{(m)}} \left[\frac{\langle r(u_{i-1}), v \rangle}{|||v|||} \right] \quad (6.12)$$

$$c^{(m+1)} = \arg \min_{c \in \mathbb{R}^n} \|A^{(m)}c - F^{(m)}\|_{\ell^2}. \quad (6.13)$$

6.2.2 Extension to non-self adjoint positive-definite problems

We require that the operator a is symmetric positive-definite in order for 6.2.1 to hold. In the case of non-symmetric positive-definite problems, we may extend Galerkin neural networks by utilizing least squares variational formulations. Let us consider the generic boundary value problem

$$\begin{cases} \mathcal{L}[\mathbf{u}] = \mathbf{f} & \text{in } \Omega \\ \mathcal{B}[\mathbf{u}] = \mathbf{g} & \text{on } \partial\Omega, \end{cases} \quad (6.14)$$

where $\mathcal{L} : \mathcal{X} \rightarrow \mathcal{V}$ and $\mathcal{B} : \mathcal{X} \rightarrow \mathcal{W}$ are linear differential operators, $\mathbf{f} \in \mathcal{V}$, and $\mathbf{g} \in \mathcal{W}$. We assume that $\mathcal{V}, \mathcal{W}$, and $\mathcal{X}$ are Hilbert spaces. We further assume the existence of a unique solution $\mathbf{u} \in \mathcal{X}$ which depends continuously on the data $\mathbf{f}$ and $\mathbf{g}$:

$$\|\mathbf{u}\|_{\mathcal{X}} \leq C(\|\mathbf{f}\|_{\mathcal{V}} + \|\mathbf{g}\|_{\mathcal{W}}). \quad (6.15)$$

The least squares formulation associated with (6.14) which we shall consider is:

$$\begin{aligned} \mathbf{u} \in \mathcal{X} : a_{LS}(\mathbf{u}, \mathbf{v}) &:= (\mathcal{L}[\mathbf{u}], \mathcal{L}[\mathbf{v}])_{\mathcal{V}} + \delta(\mathcal{B}[\mathbf{u}], \mathcal{B}[\mathbf{v}])_{\mathcal{W}} = \\ &(\mathbf{f}, \mathcal{L}[\mathbf{v}])_{\mathcal{V}} + \delta(\mathbf{g}, \mathcal{B}[\mathbf{v}])_{\mathcal{W}} =: L_{LS}(\mathbf{v}) \quad \forall \mathbf{v} \in \mathcal{X} \end{aligned} \quad (6.16)$$

with $\delta > 0$. The operator $a_{LS} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is symmetric positive-definite regardless of the structure of $\mathcal{L}$. Least squares variational formulations have seen use in conjunction with Galerkin finite element methods [21], especially for mixed problems where they do not require the underlying approximation spaces to satisfy an inf-sup condition. Typically, one converts (6.14) to a first-order system so that $\mathcal{X} = \mathbf{H}^1(\Omega)^d$ and H^1 -conforming elements may be used. In contrast, Galerkin neural networks can directly utilize least squares formulation on higher-order spaces due to the global regularity of the activation function σ ; one need only choose $\sigma \in \mathcal{X}$.

In the literature, one often sees ad hoc choices of $\mathcal{V}$ and $\mathcal{W}$. For instance, a very common approach for training DNNs to learn the solution of the Poisson equation with essential boundary conditions (i.e. $\mathcal{L} = -\Delta$ and $\mathcal{B} = \text{Id}$) is to simply minimize the L^2 norms of the interior and boundary residuals which corresponds to $\mathcal{V} = L^2(\Omega)$ and $\mathcal{W} = L^2(\partial\Omega)$. However, in order for the energy norm associated with a_{LS} to be norm equivalent to $\|\cdot\|_{\mathcal{X}}$, we emphasize that the spaces $\mathcal{V}$ and $\mathcal{W}$ must be chosen according to (6.15).

Theorem 6.2.2. *Let $\mathbf{f} \in \mathcal{V}$, $\mathbf{g} \in \mathcal{W}$ and suppose that there exist constants $C_1, C_2 > 0$ such that*

$$1. \quad \|\mathcal{L}[\mathbf{v}]\|_{\mathcal{V}} \leq C_1 \|\mathbf{v}\|_{\mathcal{X}}, \text{ and}$$

$$2. \quad \|\mathcal{B}[\mathbf{v}]\|_{\mathcal{W}} \leq C_2 \|\mathbf{v}\|_{\mathcal{X}}$$

for all $\mathbf{v} \in \mathcal{X}$. Suppose further that there exists a unique solution $\mathbf{u} \in \mathcal{X}$ of (6.14) and a constant $C_3 > 0$ such that

$$\|\mathbf{u}\|_{\mathbf{X}} \leq C_3(\|\mathbf{f}\|_{\mathcal{V}} + \|\mathbf{g}\|_{\mathcal{W}}). \tag{6.17}$$

Then there exist constants $\mathcal{C}_1, \mathcal{C}_2 > 0$ such that

$$a_{LS}(\mathbf{u}, \mathbf{v}) \leq \mathcal{C}_1 \|\mathbf{u}\|_{\mathcal{X}} \|\mathbf{v}\|_{\mathcal{X}}, \quad \mathcal{C}_2 \|\mathbf{v}\|_{\mathcal{X}}^2 \leq a_{LS}(\mathbf{v}, \mathbf{v})$$

for all $\mathbf{u}, \mathbf{v} \in \mathcal{X}$.

Proof. By the Cauchy-Schwarz inequality and continuity of $\mathcal{L}$ and $\mathcal{B}$, we have

$$\begin{aligned} a_{LS}(\mathbf{u}, \mathbf{v}) &\leq \|\mathcal{L}[\mathbf{u}]\|_{\mathcal{V}} \|\mathcal{L}[\mathbf{v}]\|_{\mathcal{V}} + \|\mathcal{B}[\mathbf{u}]\|_{\mathcal{W}} \|\mathcal{B}[\mathbf{v}]\|_{\mathcal{W}} \\ &\leq C \|\mathbf{u}\|_{\mathcal{X}} \|\mathbf{v}\|_{\mathcal{X}}. \end{aligned}$$

To prove the second inequality, we define for any $\mathbf{v} \in \mathcal{X}$ the following: $\mathbf{f}_0(x) := \mathcal{L}[\mathbf{v}](x)$ for $x \in \Omega$, $\mathbf{g}_0(x) := \mathcal{B}[\mathbf{v}](x)$ for $x \in \partial\Omega$. Then $\mathbf{v}$ is the solution of the BVP

$$\begin{cases} \mathcal{L}[\mathbf{v}] = \mathbf{f}_0 & \text{in } \Omega \\ \mathcal{B}[\mathbf{v}] = \mathbf{g}_0 & \text{on } \partial\Omega \end{cases}$$

satisfies the a priori estimate (6.17). Hence,

$$\|\mathbf{v}\|_{\mathcal{X}}^2 \leq C(\|\mathbf{f}_0\|_{\mathcal{V}} + \|\mathbf{g}_0\|_{\mathcal{W}}) \leq C a_{LS}(\mathbf{v}, \mathbf{v}) \quad \forall v \in \mathcal{X}.$$

□

We conclude the preceding discussion with a simple example that demonstrates the necessity of choosing $\mathcal{V}$ and $\mathcal{W}$ so that the a_{LS} -energy norm is norm equivalent to $\|\cdot\|_{\mathcal{X}}$.

Example 6.2.3. Consider the sequence of harmonic functions $u_n(r, \theta) = r^n \sin(n\theta)$ in the unit circle. A straightforward computation reveals that

$$\|u_n\|_{H^2(\Omega)} \propto n^{3/2}, \quad \|u_n\|_{H^s(\partial\Omega)} \propto n^s,$$

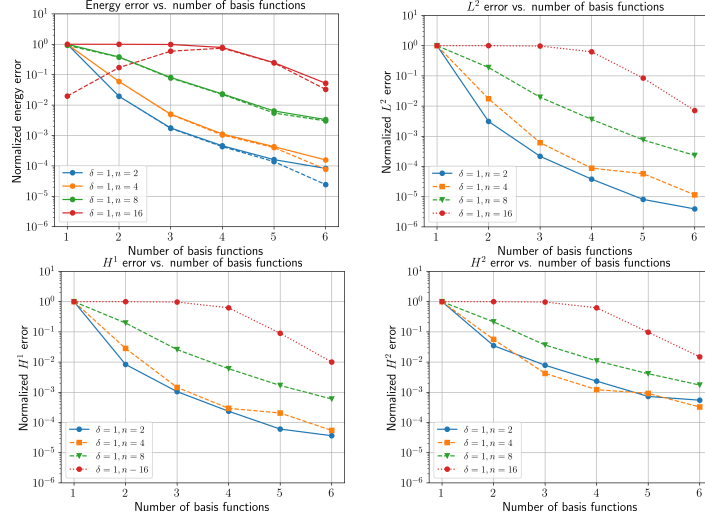


Figure 6.1: Errors with respect to the number of basis functions for $\delta = 1$, $\mathcal{W} = L^2(\partial\Omega)$, and the sequence of problems $n = 2, 4, 8, 16$.

and thus we have the ratio $\delta \|u_n\|_{H^s(\partial\Omega)} / \|u_n\|_{H^2(\Omega)} \propto \delta n^{s-3/2}$. The case $\mathcal{W} = L^2(\partial\Omega)$ corresponds to $s = 0$, and $n^{-3/2} \rightarrow 0$ as $n \rightarrow \infty$ which implies that the constant $\mathcal{C}_2$ in 6.2.2 becomes vanishingly small. Indeed, the estimate (6.15) for the Poisson equation reads [74]

$$\|u\|_{H^2(\Omega)} \leq C(\|f\|_{L^2(\Omega)} + \|g\|_{H^{3/2}(\partial\Omega)}),$$

which suggests that the $H^2(\Omega)$ formulation of the Poisson equation based on (6.16) requires the choice $\mathcal{W} = H^{3/2}(\partial\Omega)$.

For this simple example in which the true solution only contains a single frequency n , we fix $\mathcal{X} = H^2(\Omega)$ and $\mathcal{V} = L^2(\Omega)$ and we may approximately enforce $\mathcal{W} = H^s(\partial\Omega)$ by choosing $\mathcal{W} = L^2(\partial\Omega)$ and $\delta = n^\alpha$ with $\alpha = 2s$ since $n^{2s} \|u_n\|_{L^2(\partial\Omega)}^2 = \|u_n\|_{H^{3/2}(\partial\Omega)}^2$.

We first fix $\delta = n^\alpha$, $\alpha = 0$, which corresponds to the case of $\mathcal{W} = L^2(\partial\Omega)$ penalization of the boundary conditions, and consider the sequence of PDEs corresponding to $n = 2, 4, 8, 16$ in order to demonstrate the gradual loss of coercivity for this problem. Figure 6.1 shows the error in various metrics for the Galerkin neural network method with formulation (6.16). For $n = 2, 4$, the L^2 boundary penalization is sufficient since $\|u_n\|_{L^2(\partial\Omega)}, \|u_n\|_{H^{3/2}(\partial\Omega)} = \mathcal{O}(1)$. However, for larger n we begin to observe reduced coercivity and the rate of

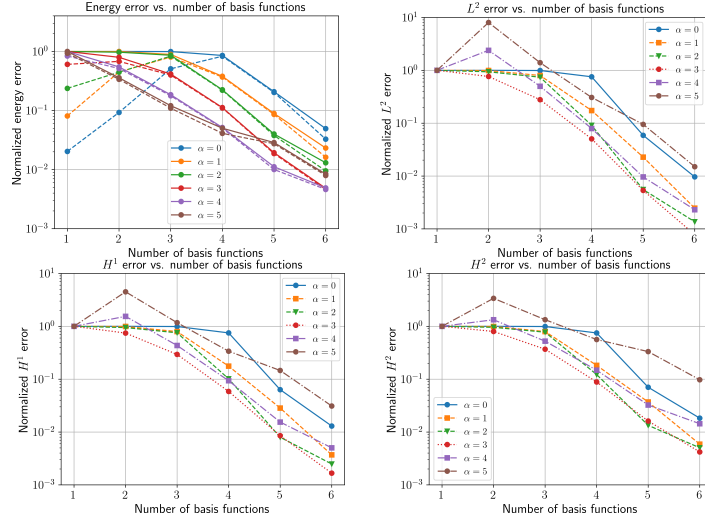


Figure 6.2: Errors with respect to the number of basis functions for several choices of penalty parameter $\delta = n^\alpha$.

convergence of the errors in all metrics is decreased as expected.

Next, we fix $n = 16$ and consider the choice $\delta = n^\alpha$ for $\alpha = 0, 1, 2, 3, 4, 5$ in order to demonstrate the effect of boundary penalization in different norms. Figure 6.2 shows the analogous results for this set of experiments. We observe that the error in the energy norm $||| \cdot ||| := a_{LS}(\cdot, \cdot)^{1/2}$ decreases monotonically regardless of the choice of α due to Theorem 6.2.1. For $\alpha = 0, 1$ ($L^2(\partial\Omega)$ and approximate $H^{1/2}(\partial\Omega)$ boundary penalization, respectively), the errors in every metric are slow to converge. It is in this regime that the loss of coercivity is strongest. Rates improve for $\alpha = 2$ (approximate $H^1(\partial\Omega)$ boundary penalization) and are optimal for $\alpha = 3$ (approximate $H^{3/2}(\partial\Omega)$ boundary penalization). On the other hand, the overpenalization corresponding to $\alpha = 4, 5$ (approximate $H^2(\partial\Omega)$ and $H^{5/2}(\partial\Omega)$ boundary penalization, respectively) results in a loss of convergence and in particular, the clear loss of norm equivalence between $||| \cdot |||$ and $|| \cdot ||_{H^k(\Omega)}$, $k \in \mathbb{N}$.

Remark 6.2.1. The computation of necessary fractional order boundary norms is possible using techniques such as those in [110]. However, generally speaking their computation is impractical and we instead utilize the highest possible integer Sobolev norm whose order is less than the order of the desired Sobolev norm. For instance, the $H^1(\partial\Omega)$ penalization in 6.2 still performs quite well when compared to the optimal $H^{3/2}(\partial\Omega)$ penalization.

6.2.4 Knowledge-based subspaces

Many of the problems we shall consider contain well-known solution structures. Suppose such a structure is known and is given by $\Psi(x; \lambda)$, parameterized by λ , which we would like to incorporate into the finite-dimensional space S_i . For instance, for the Poisson equation, the solution near corners of a polygonal domain is characterized by $\Psi(r, \theta; \lambda) \propto r^\lambda \sin(\lambda\theta)$. Such knowledge-based functions may be easily incorporated into the Galerkin neural network framework in two different ways.

1. The first approach is to make the initial guess $u_0 = \Psi(\cdot; \lambda)$. If there are multiple functions to be incorporated, say $\{\Psi_i(\cdot; \lambda)\}_{i=1}^m$, we form the subspace $S_0 := \text{span}\{\Psi_i(\cdot; \lambda)\}_{i=1}^m$ and obtain u_0 by solving

$$u_0 \in S_0 : a(u_0, v) = L(v) \quad \forall v \in S_0. \quad (6.18)$$

2. The second approach is to incorporate $\Psi_i(\cdot; \lambda)$ into the neural network architecture. Defining

$$\tilde{V}_m^\Psi := \{v : v(x) = \sum_{i=1}^m c_{\lambda,i} \Psi(x; \lambda_i), c_\lambda \in \mathbb{R}^m\}, \quad (6.19)$$

we replace (6.4) with

$$\varphi_i^{NN} = \arg \max_{v \in V_n^\sigma \oplus V_{n_\lambda}^\Psi \cap B} \langle r(u_{i-1}), v \rangle \quad (6.20)$$

and learn the parameters λ during the training step:

$$\lambda^{(m+1)} = \lambda^{(m)} + \nabla_{\lambda^{(m)}} \left[\frac{\langle r(u_{i-1}), v(\cdot; \lambda^{(m)}) \rangle}{|||v(\cdot; \lambda^{(m)})|||} \right]. \quad (6.21)$$

The parameters c and c_λ are updated simultaneously by solving the following least

squares system

$$[c^{(m+1)}; c_\lambda^{(m+1)}] = \arg \min_{c \in \mathbb{R}^n, c_\lambda \in \mathbb{R}^{n_\lambda}} \|A^{(m)}[c; c_\lambda] - F^{(m)}\|_{\ell^2} \quad (6.22)$$

for an appropriate matrix $A^{(m)}$ and vector $F^{(m)}$. Full details of their construction may be found in [Appendix 6.B](#).

The first approach may be used when the structure of $\Psi(x; \lambda)$ and its parameter dependence is known completely. The second approach is useful when the exact structure of $\Psi(x; \lambda)$ is not known either due to unknown parameters λ or c_λ . We shall discuss in [Section 6.3](#) the choice of Ψ and also demonstrate that Galerkin neural networks is capable of learning the correct parameter λ when treated as a trainable parameter.

6.3 Applications

We consider several indefinite and non-self adjoint boundary value problems and present well-posed least squares variational formulations for each of them. Of particular interest to us are problems whose solutions contain singular features, for instance due to low-regularity data or nonconvex domain corners.

As a motivating example, consider the incompressible stationary Stokes flow in velocity-pressure form given by the boundary value problem

$$\begin{cases} -\Delta \mathbf{u} + \nabla p = \mathbf{f} & \text{in } \Omega \\ \operatorname{div} \mathbf{u} = g & \text{in } \Omega \\ \mathbf{u} = \mathbf{u}_D & \text{on } \partial\Omega, \end{cases} \quad (6.23)$$

where $\Omega \subset \mathbb{R}^2$ is a bounded and connected polygonal domain, $\mathbf{u}$ is the fluid velocity, and p the fluid pressure. The most natural variational formulation associated with [\(6.23\)](#) is to

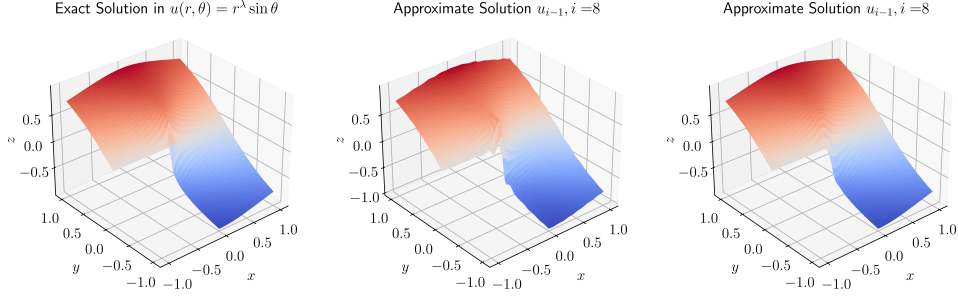


Figure 6.3: The true solution $u(r, \theta) = r^\lambda \sin \theta$ (left) and approximate solutions obtain using Galerkin neural networks and (6.27) with $\beta = 0$ (middle) and $\beta = 1$ (right).

seek $(\mathbf{u}, p) \in \mathbf{H}_0^1(\Omega) \times (L^2(\Omega)/\mathbb{R})$ such that

$$\begin{cases} (\nabla \mathbf{u}, \nabla \mathbf{v})_\Omega - (p, \operatorname{div} \mathbf{v})_\Omega &= (\mathbf{f}, \mathbf{v})_\Omega + (\mathbf{u}_D, \mathbf{v})_{\partial\Omega} \\ (\operatorname{div} \mathbf{u}, q)_\Omega &= (g, q)_\Omega \end{cases} \quad (6.24)$$

for all $(\mathbf{v}, q) \in \mathbf{H}_0^1(\Omega) \times (L^2(\Omega)/\mathbb{R})$. Formulation (6.24) is not SPD and thus we must use (6.16) in conjunction with Galerkin neural networks. However, if Ω contains nonconvex corners, then $\mathbf{u} \notin \mathbf{H}^2(\Omega)$ and $p \notin H^1(\Omega)$, and the spaces $\mathcal{X}$, $\mathcal{V}$, and $\mathcal{W}$ must be chosen carefully to account for the reduced regularity.

6.3.1 Poisson equation

We demonstrate the construction of a well-posed weighted least squares variational formulation for the Poisson equation:

$$\begin{cases} -\Delta u = f & \text{in } \Omega \\ u = g & \text{on } \partial\Omega. \end{cases} \quad (6.25)$$

We consider the case where Ω is a polygon with vertices $\{x^{(i)}\}_{i=1}^M$.

6.3.1.1 Regularity and variational formulation

The following a priori estimate is a result of Kondrat'ev [74].

Theorem 6.3.1. [74] Suppose $f \in L^2_\beta(\Omega)$ and $g \in H^{3/2}_\beta(\partial\Omega)$. Then (6.25) admits a unique solution $u \in H^2_\beta(\Omega)$ and

$$\|u\|_{H^2_\beta(\Omega)} \leq C(\|f\|_{L^2_\beta(\Omega)} + \|g\|_{H^{3/2}_\beta(\partial\Omega)}),$$

for some $C > 0$ independent of u , f , and g , where

$$\|u\|_{H^{3/2}_\beta(\partial\Omega)} := \inf_{U|_{\partial\Omega}=u} \|U\|_{H^2_\beta(\Omega)}.$$

The spaces $H^s_\beta(\Omega)$ are weighted Sobolev spaces defined by

$$H^s_\beta(\Omega) := \left\{ u : \prod_{i=1}^M r_i^{\beta-s+|\alpha|} D^\alpha u \in L^2(\Omega), 0 \leq |\alpha| \leq s \right\}, \quad (6.26)$$

where r_i is the local radial coordinate centered at vertex $x^{(i)}$. In what follows, we will commonly use $(\cdot, \cdot)_{\beta, \Omega}$ to denote the $L^2_\beta(\Omega)$ inner product. We propose the weighted least

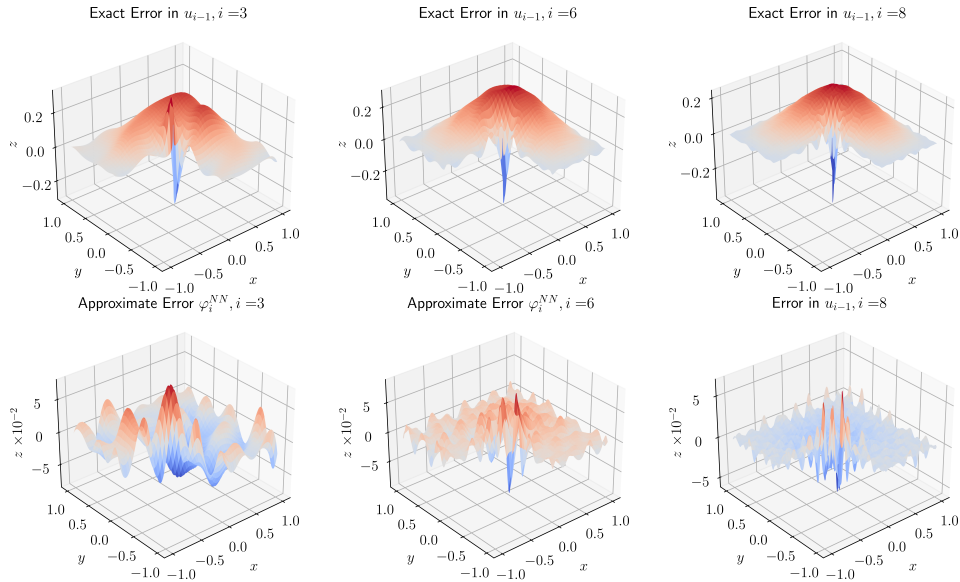


Figure 6.4: Example 6.3.2: for $\beta = 0$, the errors $u - u_{i-1}$ and learned basis functions φ_i^{NN} .

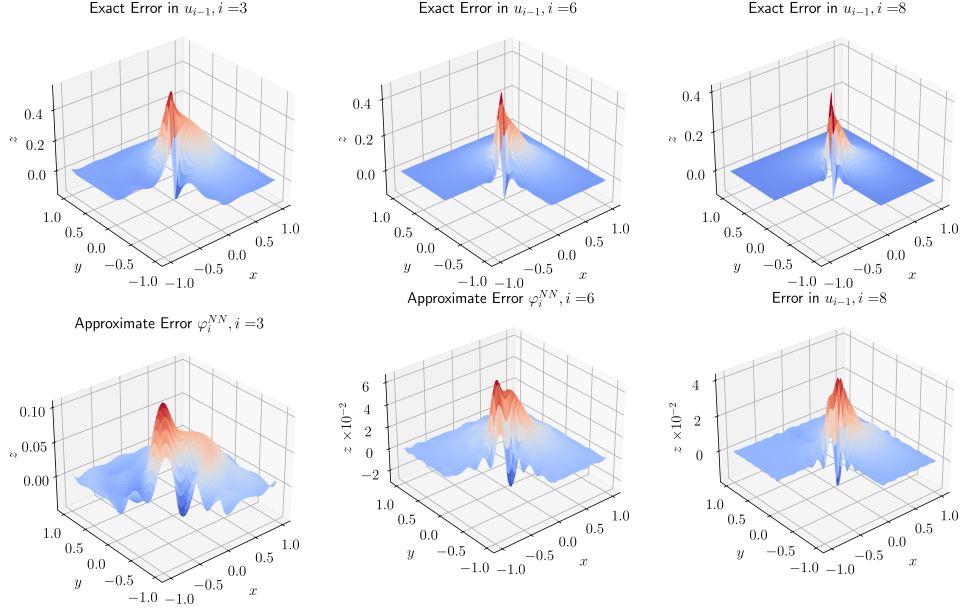


Figure 6.5: Example 6.3.2: for $\beta = 1$, the errors $u - u_{i-1}$ and learned basis functions φ_i^{NN} .

squares variational formulation given by

$$u \in H_{\beta}^2(\Omega) : (\Delta u, \Delta v)_{\beta, \Omega} + (u, v)_{H_{\beta}^{3/2}(\partial\Omega)} = -(f, \Delta v)_{\beta, \Omega} + (g, v)_{H_{\beta}^{3/2}(\partial\Omega)} \quad \forall v \in H_{\beta}^2(\Omega). \quad (6.27)$$

Example 6.3.2. We consider the case of the true solution $u(r, \theta) = r^{\lambda} \sin \theta$ for $\lambda = 1/4$ in the domain $\Omega = (-1, 1)^2 \setminus (-1, 0)^2$. In this case, $f \notin L^2(\Omega)$ but $f \in L_{\beta}^2(\Omega)$ for $\beta > 1 - \lambda$. We demonstrate that if $\beta = 0$ in (6.27), then the approximation obtained is incorrect and does not converge to the true solution. Figure 6.3 shows the approximation using 8 basis functions learned with networks of width $n_i = 20 \cdot 2^{i-1}$ and activation function $\sigma_i(t) = \tanh((1 + i/4)t)$.

We observe spurious oscillations of the approximation near $x = 0$ when $\beta = 0$ which are largely removed when $\beta = 1$. Figures 6.4-6.5 show the true errors $u - u_{i-1}$ and basis functions φ_i^{NN} for $\beta = 0$ and $\beta = 1$, respectively. We observe that in the $\beta = 0$ case, the learned basis functions contain many high frequency modes not present in the corresponding error functions. In the $\beta = 1$ case, the basis functions take on large values

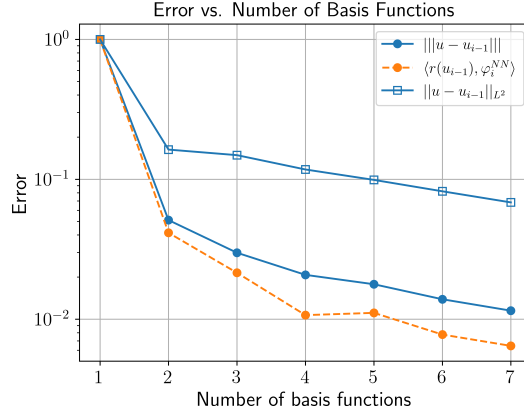


Figure 6.6: Example 6.3.2: for $\beta = 1$, the errors in the L^2 and energy norms after each basis function is added to the learned subspace.

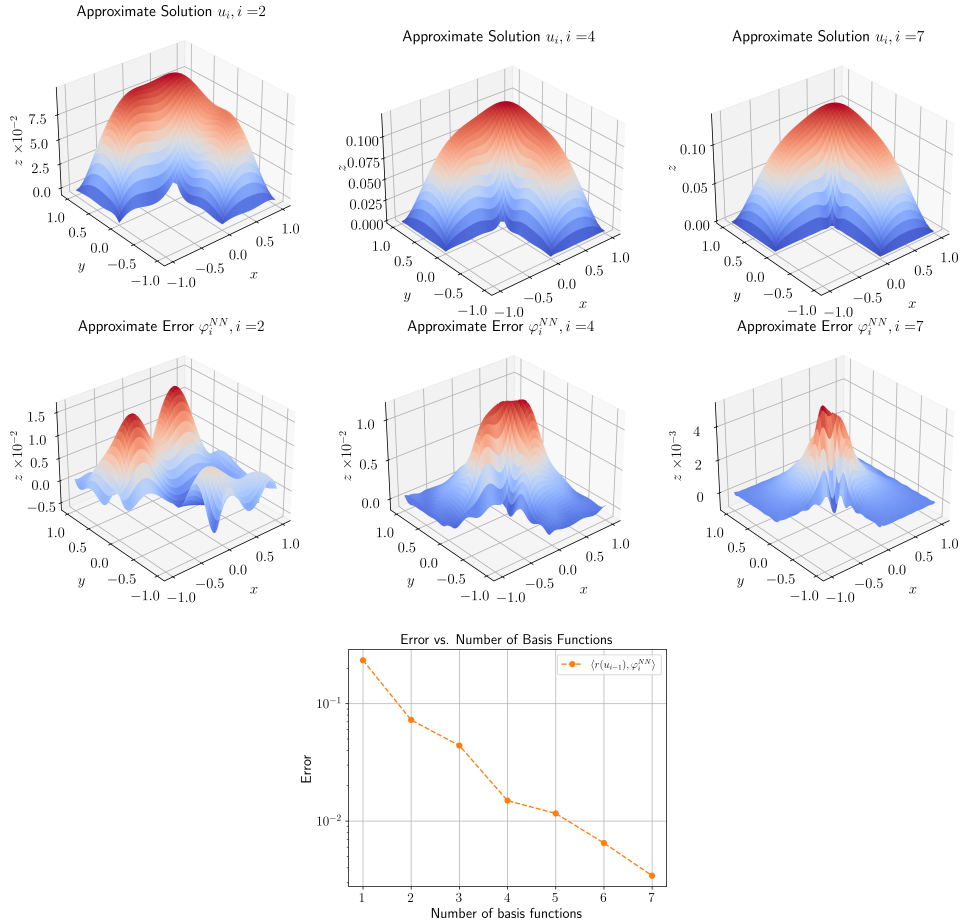


Figure 6.7: Example 6.3.3: for $\beta = 4/3$, the approximate solutions u_i and learned basis functions φ_i^{NN} .

near $x = 0$ and correctly identify the presence of error the approximation of the singularity, but convergence is still slow (see the error the energy and L^2 norms in Figure 6.6).

6.3.2.1 Knowledge-based functions

Suppose Ω contains a single corner of interest $x^{(0)}$ with interior angle $\alpha^{(0)}$. For instance, the L -shaped domain $(-1, 1)^2 \setminus (-1, 0)^2$ contains a nonconvex corner with angle $\alpha^{(0)} = 3\pi/2$ at the origin. Solutions of (6.25) in such a domain have the well-known structure

$$u(x, y) = \sum_{\lambda_j} c_{\lambda_j} r^{\lambda_j} \sin(\lambda_j \theta) + u^*(x, y), \quad (6.28)$$

where $u^* \in H^2(\Omega)$ and λ_n are the eigenvalues of the operator pencil corresponding to the Laplacian operator [76]. The eigenvalues are given by

$$\lambda_j = \frac{j\pi}{\alpha^{(0)}}, \quad j = 1, 2, 3, \dots$$

Example 6.3.3. *We consider the problem with homogeneous Dirichlet boundary condition in the domain $\Omega = (-1, 1)^2 \setminus (-1, 0)^2$ and data $f = 1$. This problem was already considered in Chapter 3 for which an the usual H^1 variational formulation of the Poisson equation was utilized. Nevertheless, this problem demonstrates many of the explemary singular features we wish to approximate and serves as an excellent example for how to utilize knowledge-based functions in the Galerkin-neural network framework.*

While there is no closed form solution to this problem, we provide the value of the a posteriori error indicator $\langle r(u_{i-1}), \varphi_i^{NN} \rangle$ for each basis function learned as well as the basis functions φ_i^{NN} themselves, which, according to Theorem 6.2.1, provide a function representation of the error $u - u_{i-1}$. Figure 6.7 shows the convergence of the error indicator as well as the first several basis functions using a standard feedforward neural network for learning each basis function with widths $n_i = 20 \cdot 2^{i-1}$ and activation function $\sigma_i(t) = \tanh((1 + i/4)t)$.

We observe that while the error indicator suggests that the approximation is converging to the true solution, the rate of convergence is quite slow and the basis functions φ_i^{NN}

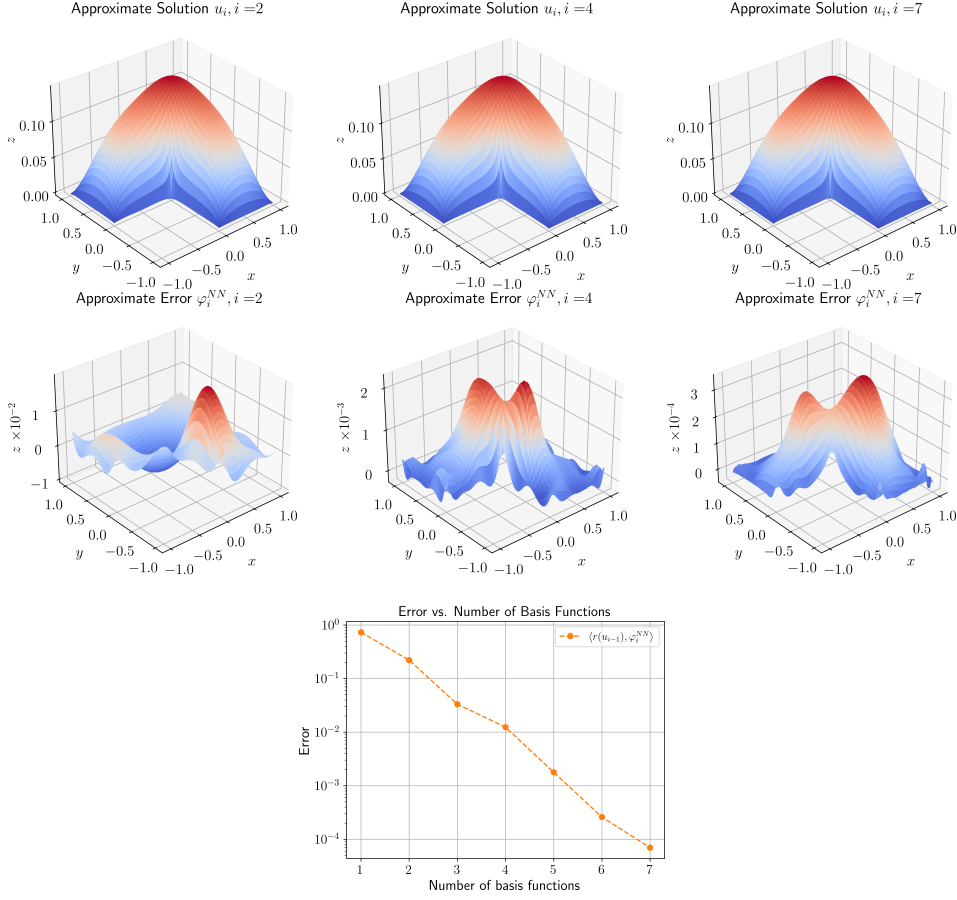


Figure 6.8: Example 6.3.3: for $\beta = 4/3$, the approximate solutions u_i and learned basis functions φ_i^{NN} .

show that the magnitude of the pointwise error is approximately $\mathcal{O}(10^{-2})$ in much of the domain while large errors are also accrued in the vicinity of the singularity.

Next, we repeat the simulation, this time with an augmented neural network architecture. Namely, we compute φ_i^{NN} according to (6.20) with $\Psi(r, \theta; \lambda) = r^\lambda \sin(\lambda\theta)$. Since the eigenvalues are known, we simply take $\lambda_j = 2j/3$. However, we shall demonstrate shortly the effect of allowing λ to be a trainable parameter that has been initialized away from the exact eigenvalues. Figure 6.8 shows the analogous results to Figure 6.7. We observe significantly faster convergence and the approximate pointwise errors decrease steadily from $\mathcal{O}(10^{-2})$ to $\mathcal{O}(10^{-5})$. Additionally, we note that the singularity is clearly captured correctly by the knowledge-based functions.

Finally, we consider the effect of allowing the eigenvalues λ_j to be treated as trainable

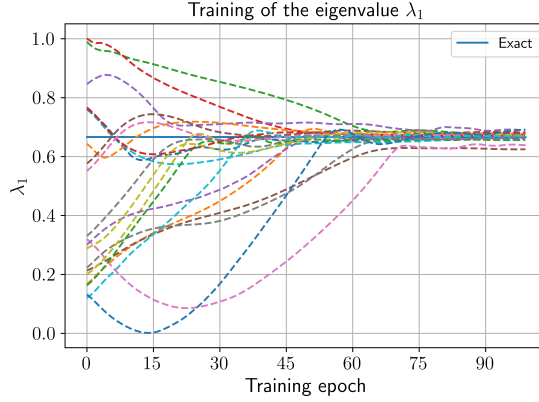


Figure 6.9: Example 6.3.3: training of λ_1 and over several runs with initialization $\lambda \sim \mathcal{U}(0, 1)$.

parameters. The robustness of this training becomes important for problems where the exact eigenvalues of the operator pencil are unknown. We supplement the feedforward neural network with only a single knowledge-based function corresponding to λ_1 which is initialized according to $\mathcal{U}(0, 1)$. Figure 6.9 shows how λ_1 changes value during the backpropagation step of training over 20 independent runs. We observe that in each case, λ_1 moves toward the true value of $2/3$. The remaining eigenvalues correspond to much smoother behavior which may be learned by the smooth part of the neural network (i.e. $\sigma(x \cdot W + b)$). In contrast, the more singular behavior associated λ_1 is the dominant behavior near $x = 0$ and is much more difficult to represent with functions of the form $\sigma(x \cdot W + b)$.

6.3.4 Stokes flow

The next application we consider is the incompressible steady Stokes flow (6.23).

6.3.4.1 Regularity and variational formulation

Schwab and Guo proved the following result in [57] for a polygonal domain Ω .

Theorem 6.3.2. [57] Suppose $\mathbf{f} \in \mathbf{L}_\beta^2(\Omega)$, $g \in H_\beta^1(\Omega)$, $\mathbf{u}_D \in \mathbf{H}_\beta^{3/2}(\partial\Omega)$, and $\int_\Omega g = 0$. Then (6.23) admits a unique solution $(\mathbf{u}, p) \in \mathbf{H}_\beta^2(\Omega) \times H_\beta^1(\Omega)$ and

$$\|\mathbf{u}\|_{\mathbf{H}_\beta^2(\Omega)} + \|p\|_{H_\beta^1(\Omega)} \leq C(\|\mathbf{f}\|_{\mathbf{L}_\beta^2(\Omega)} + \|g\|_{H_\beta^1(\Omega)} + \|\mathbf{u}_D\|_{\mathbf{H}_\beta^{3/2}(\partial\Omega)}) \quad (6.29)$$

for some $C > 0$ independent of $\mathbf{u}$, p , $\mathbf{f}$, g , and $\mathbf{u}_D$.

Most notable is that (6.3.2) suggests that the residual of the incompressibility equation in (6.23) should actually be taken in the H_β^1 norm, which leads to the continuous and coercive formulation: seek $(\mathbf{u}, p) \in \mathbf{H}_\beta^2(\Omega) \times H_\beta^1(\Omega)$ such that

$$\begin{aligned} (-\Delta \mathbf{u} + \nabla p, -\Delta v + \nabla q)_{\beta, \Omega} + (\operatorname{div} \mathbf{u}, \operatorname{div} \mathbf{v})_{H_\beta^1(\Omega)} + (\mathbf{u}, \mathbf{v})_{H_\beta^{3/2}(\partial\Omega)} \\ = (\mathbf{f}, -\Delta \mathbf{v} + \nabla q)_{\beta, \Omega} + (g, \operatorname{div} \mathbf{v})_{H_\beta^1(\Omega)} + (\mathbf{u}_D, \mathbf{v})_{\mathbf{H}_\beta^{3/2}(\partial\Omega)} \end{aligned}$$

for all $(\mathbf{v}, q) \in \mathbf{H}_\beta^2(\Omega) \times H_\beta^1(\Omega)$. For ease of notation, we set $X := \mathbf{H}_\beta^2(\Omega) \times H_\beta^1(\Omega)$ endowed with the norm $\|(\mathbf{v}, q)\|_X := (\|\mathbf{v}\|_{H_\beta^2(\Omega)}^2 + \|q\|_{H_\beta^1(\Omega)}^2)^{1/2}$ and define the bilinear operator $B : X \times X \rightarrow \mathbb{R}$ by

$$B((\mathbf{u}, p), (\mathbf{v}, q)) := (-\Delta \mathbf{u} + \nabla p, -\Delta v + \nabla q)_{\beta, \Omega} + (\operatorname{div} \mathbf{u}, \operatorname{div} \mathbf{v})_{H_\beta^1(\Omega)} + (\mathbf{u}, \mathbf{v})_{H_\beta^{3/2}(\partial\Omega)}. \quad (6.30)$$

6.3.4.2 Knowledge-based functions

Many features of interest in the Stokes flow develop in the corners of Ω , including singularities at reentrant corners and eddies in convex corners that are driven by the far-field flow. In these cases, the solution structure is well-known. In particular, the

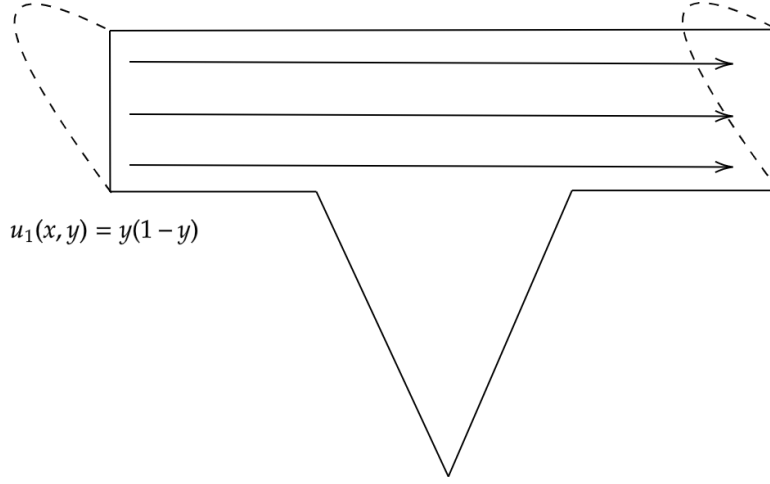


Figure 6.10: Channel with triangular cavity for Example 6.3.5.

streamfunction ψ of the flow takes the form

$$\psi(x, y) = \sum_{j=1}^M \chi(r^{(j)}) \sum_{\lambda_n^{(j)}} c_{\lambda_n^{(j)}} r^{\lambda_n^{(j)}} \Phi_{\lambda_n^{(j)}}(\theta^{(j)}) + \psi^*(x, y), \quad (6.31)$$

where $(r^{(j)}, \theta^{(j)})$ are the local polar coordinates centered at $x^{(j)}$, $\{\lambda_n^{(j)}\}$ are the eigenvalues associated with the operator pencil of the biharmonic operator at vertex $x^{(j)}$ and $\Phi_{\lambda_n^{(j)}}$ are the corresponding generalized eigenfunctions, $\chi_j \in C^s(\bar{\Omega})$ is a cutoff function centered at $x^{(j)}$, and $\psi^* \in H^4(\Omega)$.

For $\lambda \neq 0, 1, 2$, the eigenvalues of the biharmonic operator pencil at $x^{(j)}$ are given by [20]

$$\sin^2((\lambda - 1)\alpha_j) - (\lambda - 1)^2 \sin^2(\alpha_j) = 0, \quad (6.32)$$

where α_j is the interior angle of $x^{(j)}$. The eigenfunctions [20] are given by

$$\Phi_\lambda(\theta) = A \sin(\lambda\theta) + B \cos(\lambda\theta) + C \sin((\lambda - 2)\theta) + D \cos((\lambda - 2)\theta), \quad (6.33)$$

where the constants A, B, C, D are chosen so that Φ satisfies the boundary conditions.

In the case when $\alpha_j \lesssim 146^\circ$, solutions of (6.32) are complex and we replace $r^{\lambda_n} \Phi_{\lambda_n}(\theta)$ by $\Re[r^{\lambda_n} \Phi_{\lambda_n}(\theta)]$ in (6.31). Interestingly, complex eigenvalues are associated with infinite sequences of eddies in the flow due to disturbances in the far field. This behavior is described mathematically by, for instance

$$\Re(r^{a+ib} \cos((a+ib)\theta)) = r^a \cos(b \log r) \cos(a\theta) \cosh(b\theta).$$

One can show, as in [94], that each successive eddy decays exponentially in magnitude. Thus, Stokes flows in polygonal domains must be resolved to a very high degree of accuracy in order to adequately capture these eddies.

Example 6.3.5. *Let Ω be the channel with triangular cavity as shown in Figure 4.2.*

The flow takes a parabolic profile prescribed at the inflow and outflow. At the reentrant corners, we expect a singularity to manifest in both the velocity and pressure. At the bottom corner, we expect infinitely cascading eddies driven by the channel flow. As described in Section 6.3.4.2, we shall supplement the standard feedforward network with $\Psi(r, \theta; \lambda) = \chi(r) \Re[r^\lambda \Phi_\lambda(\theta)]$ in the bottom corner and $\Psi(r, \theta; \lambda) = \chi(r) r^\lambda \Phi_\lambda(\theta)$ at the reentrant corners. Precise details may be found in Appendix 6.A.

We first demonstrate the importance of choosing the Sobolev weight parameter β appropriately. In contrast with Example 6.3.2, this example contains smooth data f and g and continuous data $\mathbf{u}_D$. Figure 6.11 shows the approximated velocities and pressures for $\beta = 0$. The basis functions are learned using a standard feedforward network consisting of a single hidden layer each. We shall use $\varphi_{1,i}^{NN}$, $\varphi_{2,i}^{NN}$, and q_i^{NN} to denote the i th basis functions for the velocities and pressure, respectively. The resulting velocity and pressure fields in Figure 6.11 show that even after learning six basis functions for each field, the flow is nonphysical; we expect a near constant velocity u_1 along the x -direction in the channel. Figure 6.11 also shows the analogous results for $\beta = 5/3$. Despite not adequately capturing the singularities and the expected sequence of eddies, the approximate solutions

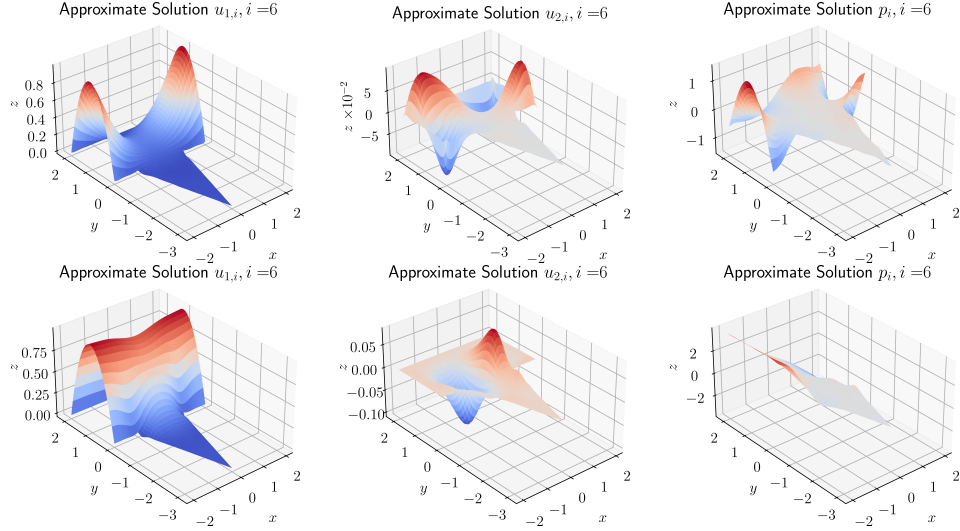


Figure 6.11: Example 6.3.5: velocity and pressure with $\beta = 0$ (top) and $\beta = 5/3$ (bottom) in (6.30).

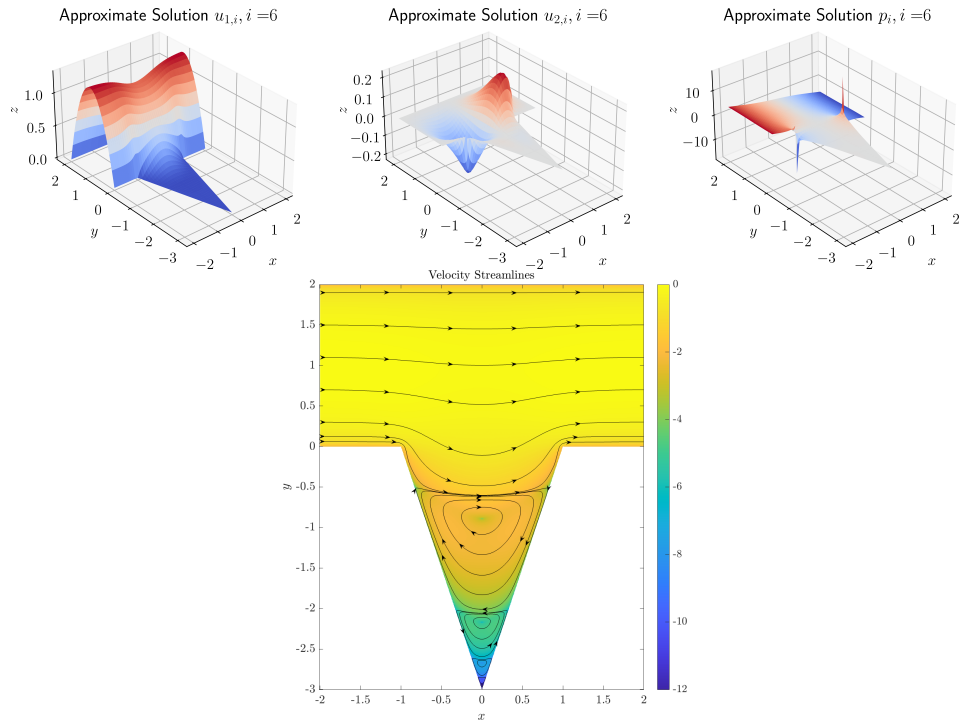


Figure 6.12: Example 6.3.5: velocity and pressure with $\beta = 5/3$ in (6.30) and basis functions learned from enriched neural network with knowledge-based functions.

do not exhibit spurious oscillations and the behavior in the channel is as expected.

We fix $\beta = 5/3$ next and approximate the solution using six basis functions learned by Galerkin neural networks. We remarkably observe that with the supplemental activation

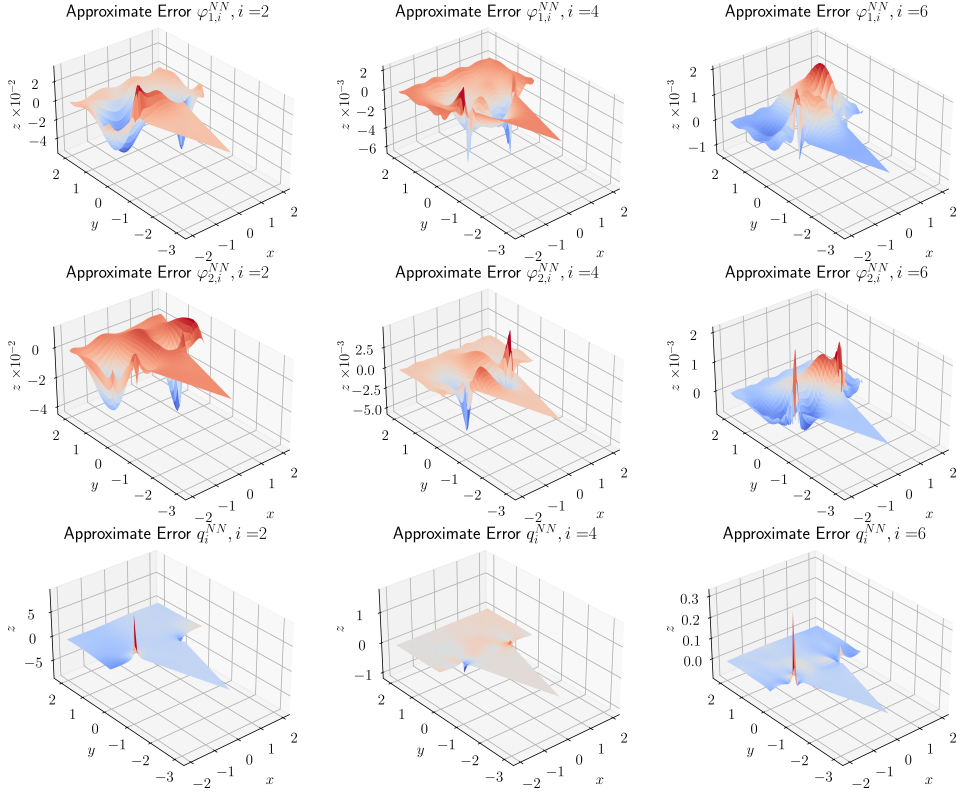


Figure 6.13: Example 6.3.5: velocity and pressure basis functions with $\beta = 5/3$ in (6.30) and basis functions learned from enriched neural network with knowledge-based functions.

functions, i.e. $\varphi_{1,i}^{NN}, \varphi_{2,i}^{NN}, q_i^{NN} \in V_n^\sigma \oplus V_{n\lambda}^\Psi$, both the singular features at the reentrant corners and the eddies in the cavity are captured accurately. In fact, as Figure 6.12 shows, we are able to capture infinitely many eddies. The basis functions used to approximate the velocities and pressures are shown in Figure 6.13. An interesting observation is the knowledge-based functions are required at each Galerkin iteration – for instance, at $i = 1$, the pressure approximation contains only one “side” of the singularities at the reentrant corners. The basis function q_2^{NN} corrects this by learning the missing “side.”

Finally, when β is too large, the solution space contains functions which may have worse singular behavior than the true solution due to the strong damping effect of r^β on the singularities. In particular, the bilinear operator B may admit as arguments functions with far stronger singularities than those of the true solution. We show the results for $\beta = 4$ in Figure 6.14.

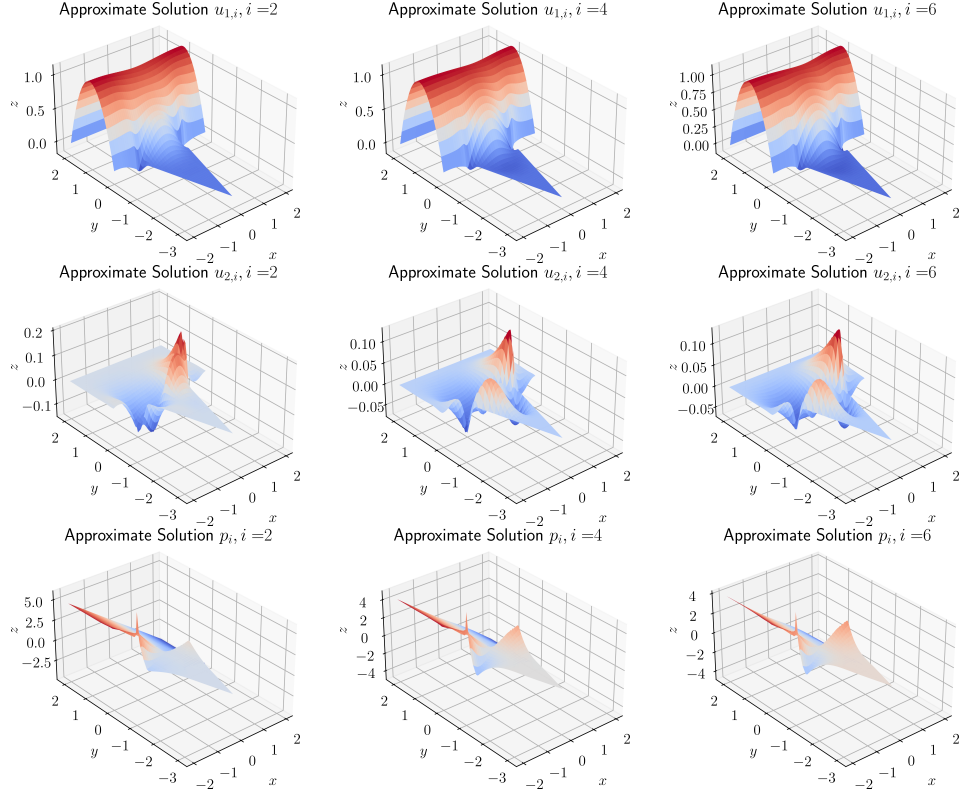


Figure 6.14: Example 6.3.5: velocity and pressure basis functions with $\beta = 4$ in (6.30) and basis functions learned from enriched neural network with knowledge-based functions.

6.4 Conclusions

We have presented a least squares variational formulation for a general class of boundary value problems which is compatible with approximation Galerkin neural network. This formulation is straightforward to implement and does not require a recasting of the PDE as a first-order system as is often necessary or desired in finite element contexts. We have demonstrated the importance of the selection of underlying norms in the proposed formulation, in particular that often the popular choice of $L^2(\Omega)$ and $L^2(\partial\Omega)$ seen in many neural network methods leads to formulations that are not well-posed and induced energy norms which are not equivalent to desired target norms.

We have also demonstrated how to appropriately develop least squares formulations with low-regularity data and/or solutions through the choice of weighted Sobolev spaces as the underlying function space of the variational formulation. In conjunction with weighted

least squares formulations, we have shown how Galerkin neural networks can easily incorporate so-called knowledge-based functions into the neural network representation space. Such functions are vital to the accurate approximation of singular solutions and we demonstrate that Galerkin neural networks are even capable of learning the eigenvalues and eigenfunctions of the underlying operator pencils that form the singular solutions for such problems.

6.5 Acknowledgments

J.D. thanks the National Science Foundation Graduate Research Fellowship for its financial support under Grant No. 1644760.

Appendix

6.A Knowledge-based functions for Stokes flow

Given a vertex $x^{(i)}$ of Ω with interior angle $\alpha^{(i)}$, the solution of the Stokes flow in the vicinity of $x^{(i)}$ may be written into terms of the eigenfunctions of the operator pencil of the Stokes operator [76]. It is simpler to first consider the streamfunction ψ , from which the velocity may be recovered with the relation $\mathbf{u} = \nabla^\perp \psi$. The streamfunction takes the form

$$\psi(r, \theta) = \sum_{\lambda_n} c_{\lambda_n} r^{\lambda_n} \Psi_{\lambda_n}(\theta),$$

where for $\lambda_n \neq 0$, Ψ_{λ_n} is given by

$$\Psi_{\lambda_n}(\theta) := A \cos(\lambda_n \theta) + B \sin(\lambda_n \theta) + C \cos((\lambda - 2)\theta) + D \sin((\lambda - 2)\theta).$$

The constants A, B, C, D are determined by the boundary conditions. The eigenfunctions for the pressure are given in [76]. Altogether, we have

$$\begin{aligned}
\begin{bmatrix} u_1(r, \theta) \\ u_2(r, \theta) \\ p(r, \theta) \end{bmatrix} &= A \sum_{\lambda_n} \begin{bmatrix} r^{\lambda_n-1} (\cos(\lambda_n \theta) \sin \theta - \sin(\lambda_n \theta) \cos \theta) \\ r^{\lambda_n-1} (\cos(\lambda_n \theta) \cos \theta + \sin(\lambda_n \theta) \sin \theta) \\ r^{\lambda_n-2} \cos(\lambda_n \theta) \end{bmatrix} \\
&+ B \sum_{\lambda_n} \begin{bmatrix} r^{\lambda_n-1} (\sin(\lambda_n \theta) \sin \theta + \cos(\lambda_n \theta) \cos \theta) \\ r^{\lambda_n-1} (\sin(\lambda_n \theta) \cos \theta - \cos(\lambda_n \theta) \sin \theta) \\ r^{\lambda_n-2} \sin(\lambda_n \theta) \end{bmatrix} \\
&+ C \sum_{\lambda_n} \begin{bmatrix} r^{\lambda_n-1} (\lambda_n \cos((\lambda_n - 2)\theta) \sin \theta - (\lambda_n - 2) \sin(\lambda_n \theta) \cos \theta) \\ r^{\lambda_n-1} (\lambda_n \cos(\lambda_n - 2)\theta) \cos \theta + (\lambda_n - 2) \sin(\lambda_n \theta) \sin \theta \\ r^{\lambda_n-2} \cos((\lambda_n - 2)\theta) \end{bmatrix} \\
&+ D \sum_{\lambda_n} \begin{bmatrix} r^{\lambda_n-1} (\lambda_n \sin((\lambda_n - 2)\theta) \sin \theta + (\lambda_n - 2) \cos(\lambda_n \theta) \cos \theta) \\ r^{\lambda_n-1} (\lambda_n \sin(\lambda_n - 2)\theta) \cos \theta - (\lambda_n - 2) \cos(\lambda_n \theta) \sin \theta \\ r^{\lambda_n-2} \sin((\lambda_n - 2)\theta) \end{bmatrix}
\end{aligned}$$

We allow the Galerkin neural network training procedure determine A, B, C, D during the the least square step (6.22).

In the special case of complex eigenvalues $\lambda_n = \xi_n + i\zeta_n$, we must take $\Re[r^{\lambda_n} \Psi(\theta)]$:

$$\begin{aligned}
\begin{bmatrix} u_1(r, \theta) \\ u_2(r, \theta) \end{bmatrix} &= A \sum_{\lambda_n} \begin{bmatrix} \frac{\partial}{\partial r} \Re[r^{\lambda_n}] \Re[\Psi_{\lambda_n}(\theta)] \sin \theta + \Re[r^{\lambda_n}] \frac{\partial}{\partial \theta} \Re[\Psi_{\lambda_n}(\theta)] \frac{\cos \theta}{r} \\ -\frac{\partial}{\partial r} \Re[r^{\lambda_n}] \Re[\Psi_{\lambda_n}(\theta)] \cos \theta + \Re[r^{\lambda_n}] \frac{\partial}{\partial \theta} \Re[\Psi_{\lambda_n}(\theta)] \frac{\sin \theta}{r} \end{bmatrix} \\
&+ A \sum_{\lambda_n} \begin{bmatrix} -\frac{\partial}{\partial r} \Im[r^{\lambda_n}] \Im[\Psi_{\lambda_n}(\theta)] \sin \theta - \Im[r^{\lambda_n}] \frac{\partial}{\partial \theta} \Im[\Psi_{\lambda_n}(\theta)] \frac{\cos \theta}{r} \\ \frac{\partial}{\partial r} \Im[r^{\lambda_n}] \Im[\Psi_{\lambda_n}(\theta)] \cos \theta - \Im[r^{\lambda_n}] \frac{\partial}{\partial \theta} \Im[\Psi_{\lambda_n}(\theta)] \frac{\sin \theta}{r} \end{bmatrix} \\
&+ B \sum_{\lambda_n} \begin{bmatrix} \frac{\partial}{\partial r} \Re[r^{\lambda_n}] \Im[\Psi_{\lambda_n}(\theta)] \sin \theta + \Re[r^{\lambda_n}] \frac{\partial}{\partial \theta} \Im[\Psi_{\lambda_n}(\theta)] \frac{\cos \theta}{r} \\ -\frac{\partial}{\partial r} \Re[r^{\lambda_n}] \Im[\Psi_{\lambda_n}(\theta)] \cos \theta + \Re[r^{\lambda_n}] \frac{\partial}{\partial \theta} \Im[\Psi_{\lambda_n}(\theta)] \frac{\sin \theta}{r} \end{bmatrix} \\
&+ B \sum_{\lambda_n} \begin{bmatrix} \frac{\partial}{\partial r} \Im[r^{\lambda_n}] \Re[\Psi_{\lambda_n}(\theta)] \sin \theta + \Im[r^{\lambda_n}] \frac{\partial}{\partial \theta} \Re[\Psi_{\lambda_n}(\theta)] \frac{\cos \theta}{r} \\ -\frac{\partial}{\partial r} \Im[r^{\lambda_n}] \Re[\Psi_{\lambda_n}(\theta)] \cos \theta - \Im[r^{\lambda_n}] \frac{\partial}{\partial \theta} \Re[\Psi_{\lambda_n}(\theta)] \frac{\sin \theta}{r} \end{bmatrix}, \quad (6.34)
\end{aligned}$$

where

$$\begin{aligned}
\Re[r^{\lambda_n}] &= r^{\xi_n-1} \cos(\log(r)\zeta_n), \quad \Im[r^{\lambda_n}] = r^{\xi_n-1} \sin(\log(r)\zeta_n) \\
\Re[\Psi_{\lambda_n}(\theta)] &= \cos(\xi_n \alpha^{(i)}) \cos((\xi_n - 2)\theta) \cosh(\zeta_n \alpha^{(i)}) \cosh(\zeta_n \theta) \\
&\quad + \sin(\xi_n \alpha^{(i)}) \sin((\xi_n - 2)\theta) \sinh(\zeta_n \alpha^{(i)}) \sinh(\zeta_n \theta) \\
&\quad - \cos((\xi_n - 2)\alpha^{(i)}) \cos((\xi_n - 2)\theta) \cosh(\zeta_n \alpha^{(i)}) \cosh(\zeta_n \theta) \\
&\quad + \sin((\xi_n - 2)\alpha^{(i)}) \sin(\xi_n \theta) \sinh(\zeta_n \alpha^{(i)}) \sinh(\zeta_n \theta) \\
\Im[\Psi_{\lambda_n}(\theta)] &= -\cos(\xi_n \alpha^{(i)}) \sin((\xi_n - 2)\theta) \cosh(\zeta_n \alpha^{(i)}) \sinh(\zeta_n \theta) \\
&\quad - \sin(\xi_n \alpha^{(i)}) \cos((\xi_n - 2)\theta) \sinh(\zeta_n \alpha^{(i)}) \cosh(\zeta_n \theta) \\
&\quad + \cos((\xi_n - 2)\alpha^{(i)}) \sin(\xi_n \theta) \cosh(\zeta_n \alpha^{(i)}) \sinh(\zeta_n \theta) \\
&\quad + \sin((\xi_n - 2)\alpha^{(i)}) \cos(\xi_n \theta) \sinh(\zeta_n \alpha^{(i)}) \cosh(\zeta_n \theta).
\end{aligned}$$

We note that (6.34) has been calculated to exactly satisfy a homogeneous Dirichlet boundary condition along $\theta = \pm \alpha^{(i)}$. The transformation $\theta \mapsto \theta - \phi$, $\phi \in \mathbb{R}$ should be taken if necessary depending on specific domain geometry. Additionally, the constants A and B here are solely for determining magnitude of the velocity and are learned/updated during the least squares step (6.22).

Finally, the pressure in the complex eigenvalue scenario are given by

$$\begin{aligned}
p(r, \theta) = & Ar^{\xi_n-2} \left(\cos(\xi_n \alpha^{(i)}) \cosh(\zeta_n \alpha^{(i)}) (\cos((\xi_n - 2)\theta) \sinh(\zeta_n \theta) \cdot \right. \\
& (-(\xi_n - 1) \sin(\log(r) \zeta_n) - \zeta_n \cos(\log(r) \zeta_n)) - \sin((\xi_n - 2)\theta) \cosh(\zeta_n \theta) \cdot \\
& (\zeta_n \sin(\log(r) \zeta_n) - (\xi_n - 1) \cos(\log(r) \zeta_n))) \\
& + \sin(\xi_n \alpha^{(i)}) \sinh(\zeta_n \alpha^{(i)}) (\cos((\xi_n - 2)\theta) \sinh(\zeta_n \theta) \cdot \\
& ((\xi_n - 1) \cos(\log(r) \zeta_n) - \zeta_n \sin(\log(r) \zeta_n)) + \sin((\xi_n - 2)\theta) \cosh(\zeta_n \theta) \cdot \\
& \left. ((\xi_n - 1) \sin(\log(r) \zeta_n) + \zeta_n \cos(\log(r) \zeta_n))) \right) \tag{6.35} \\
& + Br^{\xi_n-2} \left(\sin(\xi_n \alpha^{(i)}) \sinh(\zeta_n \alpha^{(i)}) (\cos((\xi_n - 2)\theta) \sinh(\zeta_n \theta) \cdot \right. \\
& ((\xi_n - 1) \sin(\log(r) \zeta_n) + \zeta_n \cos(\log(r) \zeta_n)) + \sin((\xi_n - 2)\theta) \cosh(\zeta_n \theta) \cdot \\
& (\zeta_n \sin(\log(r) \zeta_n) - (\xi_n - 1) \cos(\log(r) \zeta_n))) \\
& + \cos(\xi_n \alpha^{(i)}) \cosh(\zeta_n \alpha^{(i)}) (\cos((\xi_n - 2)\theta) \sinh(\zeta_n \theta) \cdot \\
& ((\xi_n - 1) \cos(\log(r) \zeta_n) - \zeta_n \sin(\log(r) \zeta_n)) + \sin((\xi_n - 2)\theta) \cosh(\zeta_n \theta) \cdot \\
& \left. ((\xi_n - 1) \sin(\log(r) \zeta_n) + \zeta_n \cos(\log(r) \zeta_n))) \right).
\end{aligned}$$

The author notes that it may be entirely possible to simplify (6.35) into its component parts such as $\cos((\xi_n - 2)\theta) \sinh(\zeta_n \theta) \sin(\log(r) \zeta_n)$ and $\cos((\xi_n - 2)\theta) \sinh(\zeta_n \theta) \cos(\log(r) \zeta_n)$, etc. and allow their coefficients to be linear parameters learned in the least squares step (6.22), however the authors have not tested this.

6.B Training procedure for Galerkin neural networks with knowledge-based functions

When knowledge-based functions are incorporated into the feedforward neural network architecture as in (6.20), the least squares solve for c and c_λ involves a block matrix

system, i.e. (6.22). This system takes the form

$$A^{(m)} = \begin{bmatrix} A_\sigma & A_{\sigma\Psi} \\ A_{\sigma\Psi}^T & A_\Psi \end{bmatrix}, \quad F^{(m)} = \begin{bmatrix} F_\sigma \\ F_\Psi \end{bmatrix},$$

with

$$A_{\sigma,jk} = a(\sigma(x \cdot W_j^{(m)} + b_j^{(m)}), \sigma(x \cdot W_k^{(m)} + b_k^{(m)})), \quad 1 \leq j, k \leq n$$

$$A_{\sigma,jk} = a(\sigma(x \cdot W_j^{(m)}), \Psi(x; \lambda_k^{(m)})), \quad 1 \leq j \leq n, 1 \leq k \leq n_\lambda$$

$$A_{\Psi,jk} = a(\Psi(x; \lambda_j), \Psi(x; \lambda_k)), \quad 1 \leq j, k \leq n_\lambda.$$

and

$$F_{\sigma,j} = L(\sigma(x \cdot W_j^{(m)} + b_j^{(m)})) - a(u_{i-1}, \sigma(x \cdot W_j^{(m)} + b_j^{(m)})), \quad 1 \leq j \leq n$$

$$F_{\Psi,j} = L(\Psi(x; \lambda_j)) - a(u_{i-1}, \Psi(x; \lambda_j)), \quad 1 \leq j \leq n_\lambda.$$

CHAPTER 7

Conclusions

This chapter concludes the presentation, discussion, and analysis of Galerkin neural networks. We summarize for the reader the major contributions and innovations of the method as well as discuss future directions of research.

7.1 Summary of Contributions

The major contribution of this work is the development of a novel technique for approximating PDEs that leverages the advantages of traditional Galerkin methods, including their theoretical guarantees of best-in-class approximation error, and the nonlinear approximation power of neural networks. We have established rigorous mathematical theory that guides the application of the method. This is in contrast to many works in the literature which both do not have any or only very sparse underlying theory and analysis to explain and guide application, and often observe stagnation of the loss/error in training. Moreover, in practice, we observe no issues in error stagnation and are able to obtain highly accurate approximations to challenging problems such as Reissner-Mindlin plate deformation, Moffatt eddies in Stokes flow, and even problems with noisy data in which we demonstrate the ability to accurately capture high-frequency noisy behavior.

7.2 Future Directions

7.2.1 Nonlinear Problems

In this thesis, we have covered linear elliptic, parabolic, and hyperbolic problems including time-dependent problems. The most important generalization of the current work is to nonlinear PDEs such as the Navier-Stokes equations. The main limitation of Galerkin neural networks to nonlinear problems is the linear least squares training step in (6.11); the update of the nonlinear parameters does not require any inherent linearity in the differential operators or variational formulation.

There are several approaches one can take to deal with the nonlinear system in (6.11). The first is to use a fixed-point method such as a Picard iteration or a Gauss-Newton method [128]. Aside from the computational cost incurred from applying Newton’s method at every training epoch, another concern is the choice of initial guess for Newton’s method given the complicated parameter landscapes of DNNs. Some approaches to the initial guess are suggested by approaches in the literature for traditional numerical methods. For instance, in [61], the authors use an approximate solution of the Stokes equation as the initial guess of a Newton iteration for the Navier-Stokes flow using a pseudo-spectral method.

7.2.2 Dense Least Squares

The main computational cost of Galerkin neural networks is the solution of the linear least squares system (6.11). Unlike in traditional Galerkin methods which usually employ bases with compact support, the activation functions in a DNN are usually globally supported, thus resulting in a dense linear system. The size of this linear system is $n \times n$, where n is the width of the last hidden layer in the network. We have thus far only utilized direct methods ($\mathcal{O}(n^3)$ computational complexity) for solving this system, which, for moderately large n , becomes cost prohibitive.

The work in this thesis has mainly been focused on the theoretical development of Galerkin neural networks and thus there may be several straightforward avenues for improving computational efficiency that have yet to be explored. There are several divergent approaches to consider for this problem. The first is that a proper investigation of the least squares step in (6.11) may be carried out to ascertain whether any accuracy of the approximation is lost by performing the least squares update after every k gradient descent steps; the present approach performs the least squares update after every gradient descent step. Another consideration is to utilize preconditioners. In [96], the authors demonstrate that a simple Cholesky preconditioner applied to dense normal matrices for the conjugate gradient normal residual (CGNR), which has traditionally been viewed as a method for sparse matrices only, yields excellent results even for moderately ill-conditioned systems.

7.2.3 A Posteriori Error Estimation

Finally, the Galerkin neural network framework naturally yields an a posteriori error estimator for the energy of a given initial approximation u_0 . While the work in this thesis has utilized this estimator to guide the generation of an adaptive subspace based on neural networks, there are several interesting applications of the estimator involving traditional scientific computing methods.

One such application is to use the Galerkin neural network a posteriori error estimator to guide procedures such as adaptive mesh refinement by taking u_0 to be, for example, a coarse finite element approximation. In Galerkin neural networks, one typically needs to approximate the Riesz representation $u - u_0$ to a suitable degree of accuracy in order to obtain algebraic convergence results (see Theorem 6.2.1). For adaptive mesh refinement, often a crude estimate of the error is adequate. For instance, in Example 6.3.2, the basis functions φ_i^{NN} generally indicate large errors in the vicinity of the singularity, but the curvature and width of the error spike are not learned correctly, leading to a slow

convergence rate for Galerkin neural networks. This basis function φ_i^{NN} would likely be sufficient for guiding mesh refinement, however, and is inexpensive to compute if only the lowest frequency basis functions are utilized.

A major hurdle to address in this potential work is that, as has been emphasized many times throughout this thesis, Galerkin neural networks may take advantage of variational formulations posed on high-order Sobolev spaces (in fact, they often *require* such formulations when the underlying PDE is not self-adjoint and positive-definite) that would often be impractical for finite element methods which would otherwise need to resort to C^1 (or even higher regularity) elements. Given a finite element approximation u_0 , when computing the weak residual $\langle r(u_0), v \rangle$ one requires that $u_0 \in X$ which will typically not be true for higher order variational formulations. Nevertheless, a good starting point for validation of this approach is to first consider simple problems, such as the Poisson equation, for which low-order variational formulations may be utilized freely by Galerkin neural networks.

CHAPTER 8

Appendices

In this section, we present miscellaneous results regarding the architecture of the networks used with the Galerkin neural network framework. We also present a comparison with other neural network approaches in the literature.

Appendix

8.A Network Architecture

8.A.1 Effect of network depth

As stated in Chapters 3-6, we utilize feedforward networks consisting of only one hidden layer each to approximate the basis functions φ_i^{NN} . The form of these basis functions is thus

$$\varphi_i^{NN}(x; \theta) = \sum_{i=1}^n c_i \sigma(x \cdot W_i + b_i).$$

As in 2, deeper networks can be expressed as

$$u_{NN}(x; \theta) = T_\ell \circ (\sigma \circ T_{\ell-1} \cdots \circ \sigma \circ T_i \circ \cdots \circ \sigma \circ T_1)(x), \quad (8.1)$$

with $T_i(x) = x \cdot W^{(i)} + b^{(i)}$, $i = 1, \dots, \ell - 1$ and $T_\ell(x) = x \cdot c$. For deeper networks, the ResNet [60] modification to (8.1) is given by

$$u_{NN}(x; \theta) = T_\ell \circ ((Id + \sigma \circ T_{\ell-1}) \circ \cdots \circ (Id + \sigma \circ T_i) \circ \cdots \circ (\sigma \circ T_1))(x). \quad (8.2)$$

The addition of the identity operator to each of the layers beyond the first hidden layer is referred to as a “shortcut connection” whose purpose is to ensure that when adding

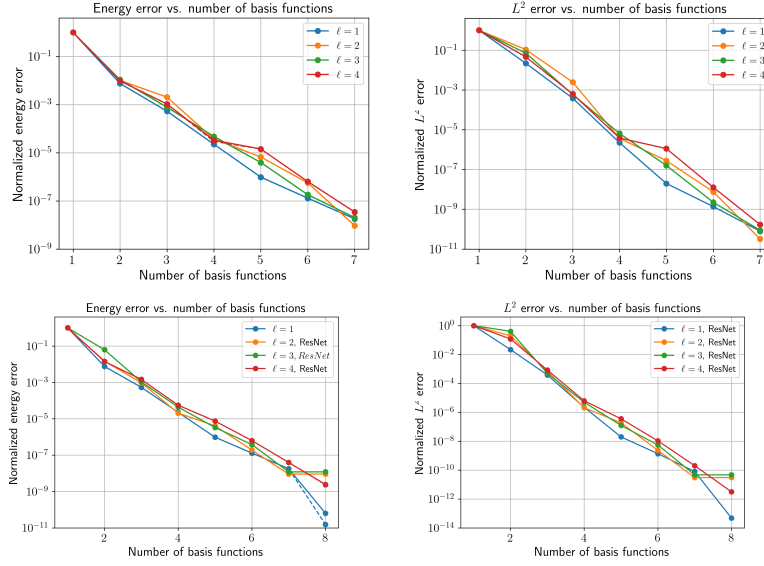


Figure 8.A.1: Convergence of the error for plain feedforward networks (top) and ResNets (bottom) when networks of depth $\ell = 1, 2, 3, 4$ are used.

depth to the network, the deeper network's training error is no worse than that of the shallower network.

In the author's experience, increasing the network depth does not, in most cases, yield any increase in accuracy of the Galerkin neural network approximation.

Example 8.A.2. *We consider the simple example*

$$\begin{cases} -u''(x) = f(x) & \text{in } (0, 1) \\ u(0) - \varepsilon u'(0) = u(1) + \varepsilon u'(1) = 0 \end{cases}$$

with true solution $u(x) = \sin(2\pi x) + 2\pi\varepsilon/(2\varepsilon + 1)(1 - 2x)$ and approximate the solution of the variational problem: find $u \in H^1(0, 1)$ such that

$$a(u, v) = (u', v')_{\Omega} + \frac{1}{\varepsilon}[u(0)v(0) + u(1)v(1)] = (f, v)_{\Omega} =: L(v) \quad \forall v \in H^1(0, 1).$$

We apply Galerkin neural networks with the number of hidden layers $\ell = 1, 2, 3, 4$. The width of each layer for the i th basis function φ_i^{NN} is given by $n_i = 5 \cdot 2^{i-1}$ when $\ell = 1$ and

$n_i = 5 \cdot 1.8^{i-1}$ when $\ell > 1$. The activation function for all layers in the i th basis function is $\sigma_i(t) = \tanh((1 + 1.8^{i-1})t)$. For $\ell > 1$, we consider both standard fully-connected feedforward networks as well as ResNets.

Figure 8.A.1 shows the error in the energy and L^2 norms after each basis function is added to the subspace. For this example, a single hidden layer is sufficient for accurately approximating the Riesz representations that comprise the basis functions. We note that for deeper networks, both of the plain and ResNet variety, there does not appear to be appreciable loss of accuracy when compared to the single layer case. However, the deeper networks require significantly smaller learning rates and are slower to train. The lack of benefit of using deeper networks has been the author's experience with most other examples presented in this thesis, including examples with boundary layers and singularities. Thus, it is usually not recommended to use deep networks in the Galerkin neural network framework. However, we present a single example to the contrary next.

Example 8.A.3. This second example is one for which feedforward networks consisting of two hidden layers yield better approximation results when compared to shallower or deeper networks. This example is a function fitting (L^2 projection in the variational context) problem given data $f \in L^2(\Omega)$: find $u \in L^2(\Omega)$ such that

$$(u, v)_\Omega = (f, v)_\Omega \quad \forall v \in L^2(\Omega).$$

The data f is the volume fraction ϕ in the monodisperse suspension flow of Chapter 5. Of particular note is that this data is highly noisy and oscillatory and, when viewed as a pure function approximation problem in which we attempt to approximate the noisy function without any pre-smoothing of the data, is remarkably able to be well-approximated by Galerkin neural networks.

Figure 8.A.3 shows the approximation to f using seven Galerkin neural network basis functions whose network depth is $\ell = 1, 2, 3$. While the case when $\ell = 1$ yields an

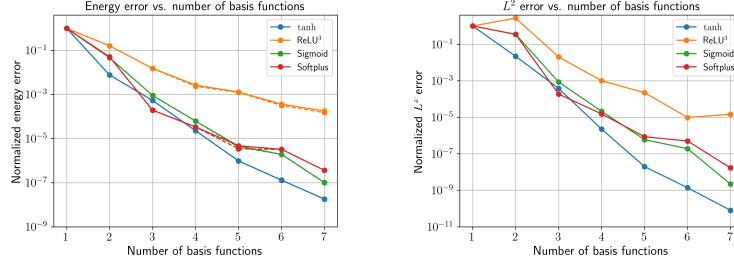


Figure 8.A.2: Convergence of the error for different activation functions.

approximation which is able to capture most of the features in the target functions, it fails to capture all of the modes present in the data. There is a noticeable improvement when $\ell = 2$ which is also observed for $\ell = 3$. However, $\ell = 3$ does not appear to confer any additional improvement in accuracy.

8.A.4 Effect of Activation Function

We next consider the effect of the choice of activation function. For this section, we shall fix the depth of the networks for each basis function φ_i^{NN} at $\ell = 1$ and consider the following choices for σ :

Sigmoid :	$\frac{1}{1 + \exp(-t)}$
Hyperbolic tangent :	$\tanh(t)$
ReLU ^k :	$\max\{0, t^k\}$
Softplus :	$\log(1 + \exp t).$

Figure 8.A.2 shows the analogous results to Figure 8.A.1 for Example 8.A.2. In general, we elect to use the hyperbolic tangent activation as it provides the most consistent results across several examples. Every PDE that has been presented in this thesis has been approximated using tanh networks. Both sigmoid and softplus perform reasonably well while ReLU performs relatively poorly.

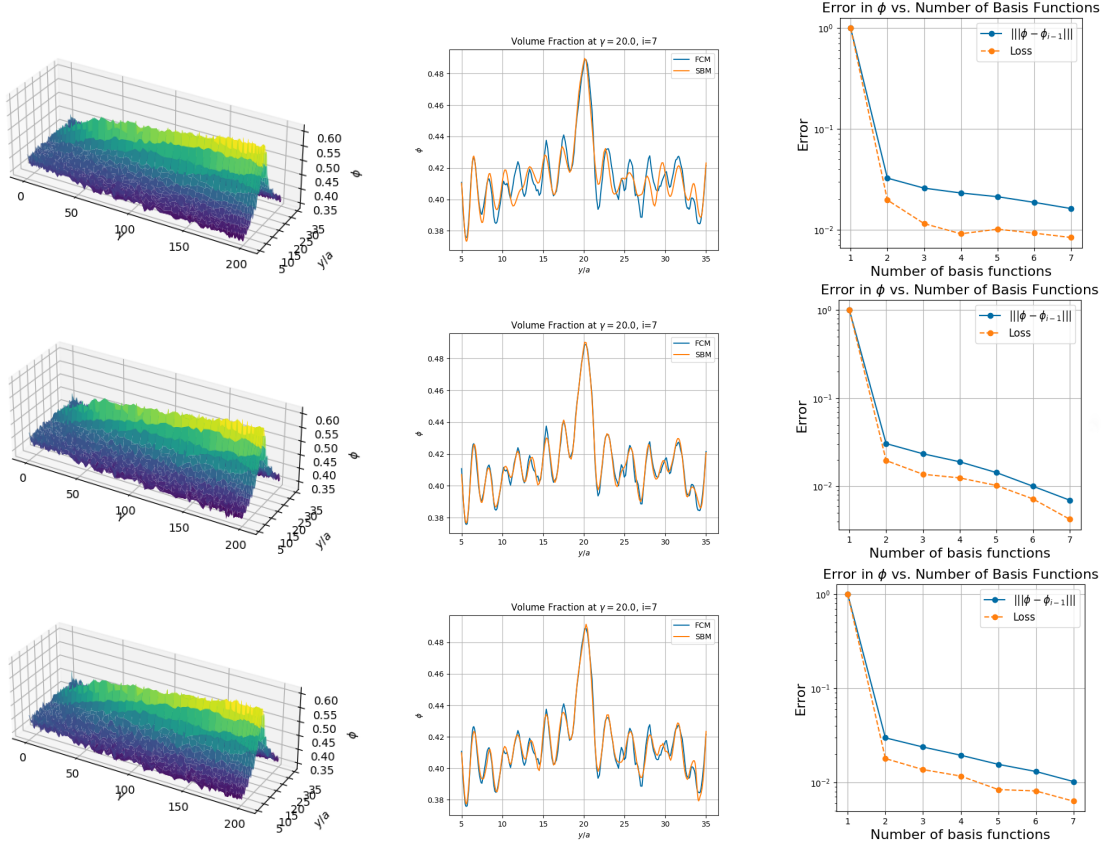


Figure 8.A.3: Approximation of the target function f for Example 8.A.3 using one (top), two (middle), and three (bottom) hidden layers in a ResNet architecture.

It is interesting to note that softplus is the convolution of a mollifier with the ReLU function. A clear issue with ReLU for higher order variational formulations is that one must instead use ReLU^k in order to avoid issues with derivatives that are not in L^2 . The second weak derivative of ReLU is not a function but rather the dirac delta distribution, which would pose issues for the variational formulations presented in this thesis as the associated bilinear operators assume at least L^2 or L^2_β regularity. The fact that the softplus activation appears to resolve much of the accuracy issues with ReLU (even when ReLU^k is used with $k = 3$) suggests that ReLU may not be suitable for problems involving high-order derivatives due to its lack of smoothness.

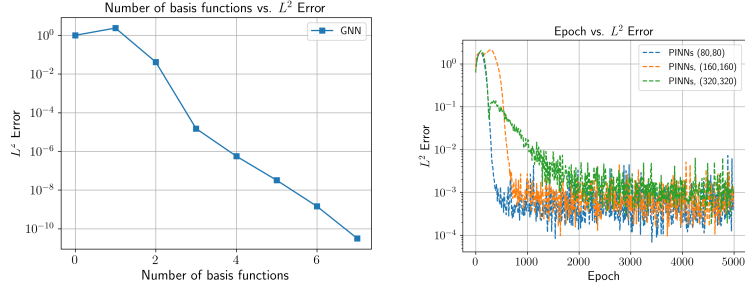


Figure 8.B.1: L^2 error in Galerkin neural networks (left) and PINNs (right).

8.B Comparison with other neural network approaches

We next compare the results of Galerkin neural networks with those of physics-informed neural networks (PINNs), which is, at the time of writing, one of, if not the most prominent works in the literature concerning neural network approximation of PDEs.

We shall first consider the BVP

$$\begin{cases} \mathcal{L}[u] = f & \text{in } \Omega \\ \mathcal{B}[u] = g & \text{on } \partial\Omega \end{cases}$$

and an associated weak formulation with a symmetric, positive-definite bilinear operator a .

$$u \in X : a(u, v) = L(v) \quad \forall v \in X.$$

The loss function for both PINNs and Galerkin neural networks is provided below for completeness:

$$\begin{aligned} \text{Galerkin neural networks :} & \quad \arg \max_v \frac{\langle r(u_{i-1}), v \rangle}{a(v, v)^{1/2}} \\ \text{PINNs :} & \quad \arg \min_v ||\mathcal{L}[v] - f||_{\Omega}^2 + ||\mathcal{B}[v] - g||_{\partial\Omega}^2 \end{aligned}$$

Example 8.B.1. Take the Poisson equation, $\mathcal{L} = -d^2/dx^2$, with homogeneous Dirichlet

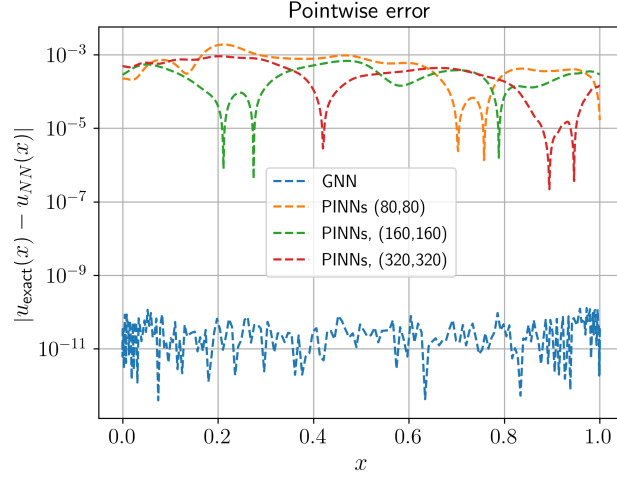


Figure 8.B.2: Pointwise error in Galerkin neural networks and PINNs.

boundary condition and solution $u(x) = \sin(2\pi x) + \sin(4\pi x) + \sin(6\pi x)$. For Galerkin neural networks, we utilize networks consisting of one hidden layer and $n_i = 5 \cdot 2^{i-1}$ neurons each. We generate eight basis functions, the last of which consists of 640 total trainable parameters. For PINNs, we use a ResNet with two hidden layers of constant widths $n = 80, 160$, and 320 .

While it is difficult to directly compare PINNs and Galerkin neural networks, we show in Figure 8.B.1 the normalized L^2 error after each Galerkin neural network basis function is added to the subspace as well as the normalized L^2 error at each training epoch for PINNs. Figure 8.B.2 shows the absolute pointwise errors of the approximations. We observe that PINNs is able to reduce the L^2 error to around $\mathcal{O}(10^{-4})$. Increasing the width of the layers does not improve this accuracy. In comparison, after learning four Galerkin neural network basis functions consisting of 5, 10, 20, and 40 neurons, respectively, the approximation error is better than of the PINNs method and approaches machine $\mathcal{O}(10^{-11})$ after the last basis function (640 neurons/trainable parameters). When considering total number of trainable parameters, the (80,80), (160,160), and (320,320) networks for PINNs contain 6640, 26080, and 103360 parameters, respectively.

Example 8.B.2. We will next consider the function approximation problem from Example 8.A.3. For PINNs, this corresponds to $\mathcal{L} = Id$ and no boundary terms in the loss function.

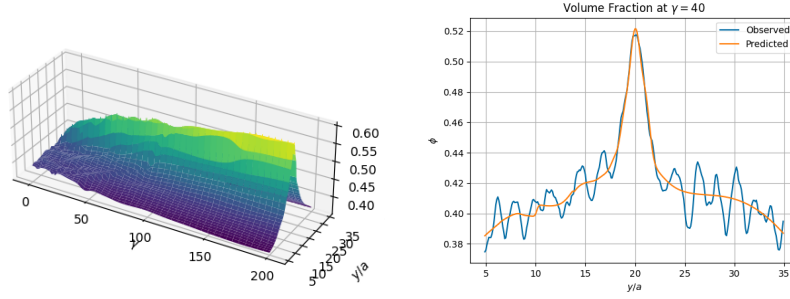


Figure 8.B.3: Pointwise error in Galerkin neural networks and PINNs.

We use ResNets with two hidden layers and $n_i = 20 \cdot 2^{i-1}$ neurons per layer for Galerkin neural networks and three hidden layers with 150 neurons per layer for PINNs. Figure 8.B.3 shows both the approximate function and a slice along $\gamma = 40$ with PINNs.

We observe that PINNs is not able to capture the multiscale behavior of the target function, instead only capturing the lowest frequency along the centerline of the channel. There are proposed modifications and improvements to PINNs such as those in [127] and [67] that perhaps more accurately approximate this target function using the PINNs methodology. However, we note that Galerkin neural networks is able to capture all of the relevant features without any special techniques that address multiscale behavior.

References

- [1] M. ABADI, A. AGARWAL, P. BARHAM, E. BREVDO, Z. CHEN, C. CITRO, G. S. CORRADO, A. DAVIS, J. DEAN, M. DEVIN, S. GHEMAWAT, I. GOODFELLOW, A. HARP, G. IRVING, M. ISARD, Y. JIA, R. JOZEFOWICZ, L. KAISER, M. KUDLUR, J. LEVENBERG, D. MANÉ, R. MONGA, S. MOORE, D. MURRAY, C. OLAH, M. SCHUSTER, J. SHLENS, B. STEINER, I. SUTSKEVER, K. TALWAR, P. TUCKER, V. VANHOUCKE, V. VASUDEVAN, F. VIÉGAS, O. VINYALS, P. WARDEN, M. WATTENBERG, M. WICKE, Y. YU, AND X. ZHENG, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. Software available from tensorflow.org.
- [2] R. A. ADAMS AND J. J. FOURNIER, *Sobolev spaces*, Elsevier, 2003.
- [3] M. AINSWORTH AND J. DONG, *Galerkin neural networks: A framework for approximating variational equations with error control*, SIAM Journal on Scientific Computing, 43 (2021), pp. A2474–A2501.
- [4] —, *Galerkin neural network approximation of singularly-perturbed elliptic systems*, Computer Methods in Applied Mechanics and Engineering, 402 (2022), p. 115169.
- [5] M. AINSWORTH AND J. T. ODEN, *A posteriori error estimation in finite element analysis*, Computer methods in applied mechanics and engineering, 142 (1997), pp. 1–88.
- [6] M. AINSWORTH AND K. PINCHEDEZ, *The hp-mitc finite element method for the reissner–mindlin plate problem*, Journal of computational and applied mathematics, 148 (2002), pp. 429–462.
- [7] M. AINSWORTH AND Y. SHIN, *Plateau phenomenon in gradient descent training of relu networks: Explanation, quantification, and avoidance*, SIAM Journal on Scientific Computing, 43 (2021), pp. A3438–A3468.

- [8] —, *Active neuron least squares: A training method for multivariate rectified neural networks*, SIAM Journal on Scientific Computing, 44 (2022), pp. A2253–A2275.
- [9] S.-I. AMARI, *Backpropagation and stochastic gradient descent method*, Neurocomputing, 5 (1993), pp. 185–196.
- [10] D. N. ARNOLD, *Discretization by finite elements of a model parameter dependent problem*, Numerische Mathematik, 37 (1981), pp. 405–421.
- [11] D. N. ARNOLD AND R. S. FALK, *Edge effects in the reissner-mindlin plate theory*, in Presented at the winter annual meeting of the American Society of Mechanical Engineers, vol. 1, 1989, p. I2.
- [12] D. N. ARNOLD AND R. S. FALK, *A uniformly accurate finite element method for the reissner–mindlin plate*, SIAM Journal on Numerical Analysis, 26 (1989), pp. 1276–1290.
- [13] —, *The boundary layer for the reissner–mindlin plate model*, SIAM Journal on Mathematical Analysis, 21 (1990), pp. 281–312.
- [14] A. ARZANI, K. W. CASSEL, AND R. M. D’SOUZA, *Theory-guided physics-informed neural networks for boundary layer problems with singular perturbation*, Journal of Computational Physics, 473 (2023), p. 111768.
- [15] I. BABUŠKA AND M. SURI, *Locking effects in the finite element approximation of elasticity problems*, Numerische Mathematik, 62 (1992), pp. 439–463.
- [16] D. BALDUZZI, M. FREAN, L. LEARY, J. LEWIS, K. W.-D. MA, AND B. MCWILLIAMS, *The shattered gradients problem: If resnets are the answer, then what is the question?*, in International Conference on Machine Learning, PMLR, 2017, pp. 342–350.
- [17] Y. BAR-SINAI, S. HOYER, J. HICKEY, AND M. P. BRENNER, *Learning data-*

- driven discretizations for partial differential equations*, Proceedings of the National Academy of Sciences, 116 (2019), pp. 15344–15349.
- [18] J. BERG AND K. NYSTRÖM, *A unified deep artificial neural network approach to partial differential equations in complex geometries*, Neurocomputing, 317 (2018), pp. 28–41.
- [19] M. J. BERGER AND J. OLIGER, *Adaptive mesh refinement for hyperbolic partial differential equations*, Journal of computational Physics, 53 (1984), pp. 484–512.
- [20] H. BLUM, R. RANNACHER, AND R. LEIS, *On the boundary value problem of the biharmonic operator on domains with angular corners*, Mathematical Methods in the Applied Sciences, 2 (1980), pp. 556–581.
- [21] P. B. BOCHEV AND M. D. GUNZBURGER, *Least-squares finite element methods*, vol. 166, Springer Science & Business Media, 2009.
- [22] J. BOHN AND M. FEISCHL, *Recurrent neural networks as optimal mesh refinement strategies*, Computers & Mathematics with Applications, 97 (2021), pp. 61–76.
- [23] F. BOYER, O. POULIQUEN, AND É. GUZZELLI, *Dense suspensions in rotating-rod flows: normal stresses and particle migration*, Journal of Fluid Mechanics, 686 (2011), pp. 5–25.
- [24] 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.
- [25] J. BRAMBLE AND T. SUN, *A negative-norm least squares method for reissner-mindlin plates*, Mathematics of computation, 67 (1998), pp. 901–916.
- [26] —, *A negative-norm least squares method for reissner-mindlin plates*, Mathematics of computation, 67 (1998), pp. 901–916.

- [27] S. C. BRENNER, L. R. SCOTT, AND L. R. SCOTT, *The mathematical theory of finite element methods*, vol. 3, Springer, 2008.
- [28] F. BREZZI, K.-J. BATHE, AND M. FORTIN, *Mixed-interpolated elements for reissner–mindlin plates*, International Journal for Numerical Methods in Engineering, 28 (1989), pp. 1787–1801.
- [29] F. BREZZI AND M. FORTIN, *Numerical approximation of mindlin-reissner plates*, Mathematics of computation, 47 (1986), pp. 151–158.
- [30] Z. CAI, *Least squares for the perturbed stokes equations and the reissner–mindlin plate*, SIAM Journal on Numerical Analysis, 38 (2000), pp. 1561–1581.
- [31] Z. CAI, X. YE, AND H. ZHANG, *Least-squares finite element approximations for the reissner-mindlin plate*, Numerical linear algebra with applications, 6 (1999), pp. 479–496.
- [32] D. CHENAIS AND J.-C. PAUMIER, *On the locking phenomenon for a class of elliptic problems*, Numerische Mathematik, 67 (1994), pp. 427–440.
- [33] G. CYBENKO, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems, 2 (1989), pp. 303–314.
- [34] E. C. CYR, M. A. GULIAN, R. G. PATEL, M. PEREGO, AND N. A. TRASK, *Robust training and initialization of deep neural networks: An adaptive basis viewpoint*, arXiv preprint arXiv:1912.04862, (2019).
- [35] —, *Robust training and initialization of deep neural networks: An adaptive basis viewpoint*, in Mathematical and Scientific Machine Learning, PMLR, 2020, pp. 512–536.
- [36] P. J. DAVIS AND P. RABINOWITZ, *Methods of numerical integration*, Courier Corporation, 2007.

- [37] R. H. DAVIS AND H. GECOL, *Hindered settling function with no empirical parameters for polydisperse suspensions*, AIChE journal, 40 (1994), pp. 570–575.
- [38] T. DE RYCK, S. LANTHALER, AND S. MISHRA, *On the approximation of functions by tanh neural networks*, Neural Networks, 143 (2021), pp. 732–750.
- [39] —, *On the approximation of functions by tanh neural networks*, Neural Networks, 143 (2021), pp. 732–750.
- [40] T. DE RYCK, S. MISHRA, AND R. MOLINARO, *wpinns: Weak physics informed neural networks for approximating entropy solutions of hyperbolic conservation laws*, arXiv preprint arXiv:2207.08483, (2022).
- [41] E. DI NEZZA, G. PALATUCCI, AND E. VALDINOCI, *Hitchhiker’s guide to the fractional sobolev spaces*, Bulletin des sciences mathématiques, 136 (2012), pp. 521–573.
- [42] M. DISSANAYAKE AND N. PHAN-THIEN, *Neural-network-based approximations for solving partial differential equations*, communications in Numerical Methods in Engineering, 10 (1994), pp. 195–201.
- [43] J. DUCHI, E. HAZAN, AND Y. SINGER, *Adaptive subgradient methods for online learning and stochastic optimization.*, Journal of machine learning research, 12 (2011).
- [44] R. DURÁN AND E. LIEBERMAN, *On mixed finite element methods for the reissner-mindlin plate model*, Mathematics of computation, 58 (1992), pp. 561–573.
- [45] W. E, C. MA, AND L. WU, *The barron space and the flow-induced function spaces for neural network models*, Constructive Approximation, 55 (2022), pp. 369–406.
- [46] J. L. ELMAN, *Finding structure in time*, Cognitive science, 14 (1990), pp. 179–211.
- [47] R. FLETCHER, *Practical methods of optimization*, John Wiley & Sons, 2013.

- [48] G. B. FOLLAND, *Real analysis: modern techniques and their applications*, vol. 40, John Wiley & Sons, 1999.
- [49] K. FUKUMIZU AND S.-I. AMARI, *Local minima and plateaus in hierarchical structures of multilayer perceptrons*, *Neural networks*, 13 (2000), pp. 317–327.
- [50] S. GERSCHGORIN, *Über die abgrenzung der eigenwerte einer matrix*, *Izvestija Akademii Nauk SSSR, Serija Matematika*, 7 (1931), pp. 749–754.
- [51] V. GIRAULT AND P.-A. RAVIART, *Finite element methods for Navier-Stokes equations: theory and algorithms*, vol. 5, Springer Science & Business Media, 2012.
- [52] X. GLOROT AND Y. BENGIO, *Understanding the difficulty of training deep feedforward neural networks*, in *Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings*, 2010, pp. 249–256.
- [53] I. GOODFELLOW, J. POUGET-ABADIE, M. MIRZA, B. XU, D. WARDE-FARLEY, S. OZAIR, A. COURVILLE, AND Y. BENGIO, *Generative adversarial networks*, *Communications of the ACM*, 63 (2020), pp. 139–144.
- [54] A. L. GRAHAM, S. A. ALTOBELLI, E. FUKUSHIMA, L. A. MONDY, AND T. S. STEPHENS, *Note: NMR imaging of shear-induced diffusion and structure in concentrated suspensions undergoing Couette flow*, *J. Rheol.*, 35 (1991), pp. 191–201.
- [55] A. GRIEWANK, *On automatic differentiation, mathematical programming*, *Recent Developments and Applications*, 83 (1989).
- [56] J. GU, Z. WANG, J. KUEN, L. MA, A. SHAHROUDY, B. SHUAI, T. LIU, X. WANG, G. WANG, J. CAI, ET AL., *Recent advances in convolutional neural networks*, *Pattern recognition*, 77 (2018), pp. 354–377.
- [57] B. GUO AND C. SCHWAB, *Analytic regularity of stokes flow on polygonal do-*

- mains in countably weighted sobolev spaces*, Journal of Computational and Applied Mathematics, 190 (2006), pp. 487–519.
- [58] R. HAMPTON, A. MAMMOLI, A. GRAHAM, N. TETLOW, AND S. ALTOBELLI, *Migration of particles undergoing pressure-driven flow in a circular conduit*, J. Rheol., 41 (1997), pp. 621–640.
- [59] J. HE, L. LI, J. XU, AND C. ZHENG, *Relu deep neural networks and linear finite elements*, Journal of Computational Mathematics, 38 (2020), pp. 502–527.
- [60] K. HE, X. ZHANG, S. REN, AND J. SUN, *Deep residual learning for image recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
- [61] P. HESSARI AND B.-C. SHIN, *The least-squares pseudo-spectral method for navier–stokes equations*, Computers & Mathematics with Applications, 66 (2013), pp. 318–329.
- [62] K. HORNIK, M. STINCHCOMBE, AND H. WHITE, *Multilayer feedforward networks are universal approximators*, Neural networks, 2 (1989), pp. 359–366.
- [63] K. HORNIK, M. STINCHCOMBE, H. WHITE, AND P. AUER, *Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives*, Neural computation, 6 (1994), pp. 1262–1275.
- [64] A. HOWARD, J. DONG, R. PATEL, M. D’ELIA, P. STINIS, AND M. MAXEY, *Machine learning methods for particle stress development in suspension poiseuille flows*, Under review, (2023).
- [65] T. J. HUGHES, *The finite element method: linear static and dynamic finite element analysis*, Courier Corporation, 2012.
- [66] T. J. HUGHES AND L. P. FRANCA, *A mixed finite element formulation for reissner-*

- mindlin plate theory: Uniform convergence of all higher-order spaces*, Computer Methods in Applied Mechanics and Engineering, 67 (1988), pp. 223–240.
- [67] 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.*, in AAAI Spring Symposium: MLPS, 2021, pp. 2002–2041.
- [68] A. D. JAGTAP, K. KAWAGUCHI, AND G. E. KARNIADAKIS, *Adaptive activation functions accelerate convergence in deep and physics-informed neural networks*, Journal of Computational Physics, 404 (2020), p. 109136.
- [69] X. JIN, S. CAI, H. LI, AND G. E. KARNIADAKIS, *Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations*, Journal of Computational Physics, 426 (2021), p. 109951.
- [70] F. JOHANSSON ET AL., *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 0.14)*, February 2010. <http://code.google.com/p/mpmath/>.
- [71] G. E. KARNIADAKIS, I. G. KEVREKIDIS, L. LU, P. PERDIKARIS, S. WANG, AND L. YANG, *Physics-informed machine learning*, Nature Reviews Physics, 3 (2021), pp. 422–440.
- [72] E. KHARAZMI, Z. ZHANG, AND G. E. KARNIADAKIS, *Variational physics-informed neural networks for solving partial differential equations*, arXiv preprint arXiv:1912.00873, (2019).
- [73] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).
- [74] V. A. KONDRAT’EV, *Boundary value problems for elliptic equations in domains*

- with conical or angular points*, Trudy Moskovskogo Matematicheskogo Obshchestva, 16 (1967), pp. 209–292.
- [75] T. KOSSACZKÁ, M. EHRHARDT, AND M. GÜNTHER, *Deep fdm: Enhanced finite difference methods by deep learning*. 04 2023.
- [76] V. KOZLOV, V. G. MAZ'YA, AND J. ROSSMANN, *Spectral problems associated with corner singularities of solutions to elliptic equations*, no. 85, American Mathematical Soc., 2001.
- [77] I. E. LAGARIS, A. LIKAS, AND D. I. FOTIADIS, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE transactions on neural networks, 9 (1998), pp. 987–1000.
- [78] D. LEIGHTON AND A. ACRIVOS, *The shear-induced migration of particles in concentrated suspensions*, Journal of Fluid Mechanics, 181 (1987), p. 415–439.
- [79] M. LESHNO, V. Y. LIN, A. PINKUS, AND S. SCHOCKEN, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, Neural networks, 6 (1993), pp. 861–867.
- [80] K. LEVENBERG, *A method for the solution of certain non-linear problems in least squares*, Quarterly of applied mathematics, 2 (1944), pp. 164–168.
- [81] R. J. LEVEQUE, *Finite volume methods for hyperbolic problems*, vol. 31, Cambridge university press, 2002.
- [82] ———, *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, SIAM, 2007.
- [83] Z. LI, N. B. KOVACHKI, K. AZIZZADENESHELI, K. BHATTACHARYA, A. STUART, A. ANANDKUMAR, ET AL., *Fourier neural operator for parametric partial differential equations*, in International Conference on Learning Representations, 2020.

- [84] K. LIPNIKOV, G. MANZINI, AND M. SHASHKOV, *Mimetic finite difference method*, Journal of Computational Physics, 257 (2014), pp. 1163–1227.
- [85] M. LIU, Z. CAI, AND J. CHEN, *Adaptive two-layer relu neural network: I. best least-squares approximation*, Computers & Mathematics with Applications, 113 (2022), pp. 34–44.
- [86] S. LOMHOLT AND M. R. MAXEY, *Force-coupling method for particulate two-phase flow: Stokes flow*, Journal of Computational Physics, 184 (2003), pp. 381–405.
- [87] L. LU, P. JIN, G. PANG, Z. ZHANG, AND G. E. KARNIADAKIS, *Learning nonlinear operators via deeponet based on the universal approximation theorem of operators*, Nature Machine Intelligence, 3 (2021), pp. 218–229.
- [88] M. LYON AND L. LEAL, *An experimental study of the motion of concentrated suspensions in two-dimensional channel flow. part 2. bidisperse systems*, Journal of Fluid Mechanics, 363 (1998), pp. 57–77.
- [89] D. W. MARQUARDT, *An algorithm for least-squares estimation of nonlinear parameters*, Journal of the society for Industrial and Applied Mathematics, 11 (1963), pp. 431–441.
- [90] W. MCLEAN, *Strongly elliptic systems and boundary integral equations*, Cambridge university press, 2000.
- [91] B. MEURIS, S. QADEER, AND P. STINIS, *Machine-learning-based spectral methods for partial differential equations*, Scientific Reports, 13 (2023), p. 1739.
- [92] R. M. MILLER AND J. F. MORRIS, *Normal stress-driven migration and axial development in pressure-driven flow of concentrated suspensions*, Journal of non-newtonian fluid mechanics, 135 (2006), pp. 149–165.
- [93] S. MISHRA AND R. MOLINARO, *Estimates on the generalization error of physics-*

- informed neural networks for approximating a class of inverse problems for pdes*, IMA Journal of Numerical Analysis, 42 (2022), pp. 981–1022.
- [94] H. K. MOFFATT, *Viscous and resistive eddies near a sharp corner*, Journal of Fluid Mechanics, 18 (1964), pp. 1–18.
- [95] J. F. MORRIS AND F. BOULAY, *Curvilinear flows of noncolloidal suspensions: The role of normal stresses*, Journal of rheology, 43 (1999), pp. 1213–1237.
- [96] J. M. MYRE, E. FRAHM, D. J. LILJA, AND M. O. SAAR, *Tnt: a solver for large dense least-squares problems that takes conjugate gradient from bad in theory, to good in practice*, in 2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW), IEEE, 2018, pp. 987–995.
- [97] P. R. NOTT AND J. F. BRADY, *Pressure-driven flow of suspensions: simulation and theory*, Journal of Fluid Mechanics, 275 (1994), pp. 157–199.
- [98] J. A. OPSCHOOR, P. C. PETERSEN, AND C. SCHWAB, *Deep relu networks and high-order finite element methods*, Analysis and Applications, 18 (2020), pp. 715–770.
- [99] —, *Deep relu networks and high-order finite element methods*, Analysis and Applications, 18 (2020), pp. 715–770.
- [100] S. PAKRAVAN, P. A. MISTANI, M. A. ARAGON-CALVO, AND F. GIBOU, *Solving inverse-pde problems with physics-aware neural networks*, Journal of Computational Physics, 440 (2021), p. 110414.
- [101] J. PARK AND I. W. SANDBERG, *Universal approximation using radial-basis-function networks*, Neural computation, 3 (1991), pp. 246–257.
- [102] A. PASZKE, S. GROSS, F. MASSA, A. LERER, J. BRADBURY, G. CHANAN, T. KILLEEN, Z. LIN, N. GIMELSHEIN, L. ANTIGA, A. DESMAISON, A. KOPF, 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 Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.
- [103] P. PETERSEN, M. RASLAN, AND F. VOIGTLAENDER, *Topological properties of the set of functions generated by neural networks of fixed size*, Foundations of computational mathematics, 21 (2021), pp. 375–444.
 - [104] R. J. PHILLIPS, R. C. ARMSTRONG, R. A. BROWN, A. L. GRAHAM, AND J. R. ABBOTT, *A constitutive equation for concentrated suspensions that accounts for shear-induced particle migration*, Physics of Fluids A: Fluid Dynamics, 4 (1992), pp. 30–40.
 - [105] M. L. PISCOPO, M. SPANNOVSKY, AND P. WAITE, *Solving differential equations with neural networks: Applications to the calculation of cosmological phase transitions*, Physical Review D, 100 (2019), p. 016002.
 - [106] M. RAISSI, P. PERDIKARIS, AND G. E. KARNIADAKIS, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational physics, 378 (2019), pp. 686–707.
 - [107] D. RAY AND J. S. HESTHAVEN, *An artificial neural network as a troubled-cell indicator*, Journal of computational physics, 367 (2018), pp. 166–191.
 - [108] J. RICHARDSON AND W. ZAKI, *The sedimentation of a suspension of uniform spheres under conditions of viscous flow*, Chemical Engineering Science, 3 (1954), pp. 65–73.
 - [109] S. H. RUDY, S. L. BRUNTON, J. L. PROCTOR, AND J. N. KUTZ, *Data-driven discovery of partial differential equations*, Science advances, 3 (2017), p. e1602614.
 - [110] S. A. SAUTER AND C. SCHWAB, *Boundary element methods*, in Boundary Element Methods, Springer, 2010, pp. 183–287.

- [111] C. SCHWAB AND C. SCHWAB, *p-and hp-finite element methods: Theory and applications in solid and fluid mechanics*, Oxford University Press, 1998.
- [112] A. SHAULY, A. WACHS, AND A. NIR, *Shear-induced particle resuspension in settling polydisperse concentrated suspension*, International journal of multiphase flow, 26 (2000), pp. 1–15.
- [113] J. W. SIEGEL AND J. XU, *Approximation rates for neural networks with general activation functions*, Neural Networks, 128 (2020), pp. 313–321.
- [114] J. SIRIGNANO AND K. SPILIOPOULOS, *Dgm: A deep learning algorithm for solving partial differential equations*, Journal of computational physics, 375 (2018), pp. 1339–1364.
- [115] —, *Dgm: A deep learning algorithm for solving partial differential equations*, Journal of computational physics, 375 (2018), pp. 1339–1364.
- [116] B. SNOOK, J. E. BUTLER, AND É. GUAZZELLI, *Dynamics of shear-induced migration of spherical particles in oscillatory pipe flow*, J. Fluid Mech, 786 (2016), pp. 128–153.
- [117] R. STENBERG, *A new finite element formulation for the plate bending problem*, Asymptotic methods for elastic structures, (1995), pp. 209–221.
- [118] R. STENBERG AND M. SURI, *An hp error analysis of mitc plate elements*, SIAM journal on numerical analysis, 34 (1997), pp. 544–568.
- [119] J. C. STRIKWERDA, *Finite difference schemes and partial differential equations*, SIAM, 2004.
- [120] T. STROUBOULIS, K. COPPS, AND I. BABUŠKA, *The generalized finite element method*, Computer methods in applied mechanics and engineering, 190 (2001), pp. 4081–4193.
- [121] N. SUKUMAR, N. MOËS, B. MORAN, AND T. BELYTSCHKO, *Extended finite*

- element method for three-dimensional crack modelling*, International journal for numerical methods in engineering, 48 (2000), pp. 1549–1570.
- [122] M. SURI, I. BABUŠKA, AND C. SCHWAB, *Locking effects in the finite element approximation of plate models*, Mathematics of computation, 64 (1995), pp. 461–482.
 - [123] J. TOMPSON, K. SCHLACHTER, P. SPRECHMANN, AND K. PERLIN, *Accelerating eulerian fluid simulation with convolutional networks*, in International Conference on Machine Learning, PMLR, 2017, pp. 3424–3433.
 - [124] R. VAN DER SMAN AND H. VOLLEBREGT, *Effective temperature for sheared suspensions: a route towards closures for migration in bidisperse suspension*, Advances in colloid and interface science, 185 (2012), pp. 1–13.
 - [125] R. VERFÜRTH, *A posteriori error estimation and adaptive mesh-refinement techniques*, Journal of Computational and Applied Mathematics, 50 (1994), pp. 67–83.
 - [126] H. VOLLEBREGT, R. VAN DER SMAN, AND R. BOOM, *Model for particle migration in bidisperse suspensions by use of effective temperature*, Faraday discussions, 158 (2012), pp. 89–103.
 - [127] 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, 384 (2021), p. 113938.
 - [128] S. WRIGHT, J. NOCEDAL, ET AL., *Numerical optimization*, Springer Science, 35 (1999), p. 7.
 - [129] J. YANG, K. MITTAL, T. DZANIC, S. PETRIDES, B. KEITH, B. PETERSEN, D. FAISSOL, AND R. ANDERSON, *Multi-agent reinforcement learning for adaptive mesh refinement*, arXiv preprint arXiv:2211.00801, (2022).
 - [130] X. YANG, S. ZAFAR, J.-X. WANG, AND H. XIAO, *Predictive large-eddy-simulation*

- wall modeling via physics-informed neural networks*, Physical Review Fluids, 4 (2019), p. 034602.
- [131] D. YAROTSKY, *Error bounds for approximations with deep relu networks*, Neural Networks, 94 (2017), pp. 103–114.
- [132] K. YEO AND M. R. MAXEY, *Simulation of concentrated suspensions using the force-coupling method*, Journal of computational physics, 229 (2010), pp. 2401–2421.
- [133] K. YEO AND M. R. MAXEY, *Numerical simulations of concentrated suspensions of monodisperse particles in a Poiseuille flow*, J. Fluid Mech., 682 (2011), pp. 491–518.
- [134] B. YU ET AL., *The deep ritz method: a deep learning-based numerical algorithm for solving variational problems*, Communications in Mathematics and Statistics, 6 (2018), pp. 1–12.
- [135] X. YU AND C.-W. SHU, *Multi-layer perceptron estimator for the total variation bounded constant in limiters for discontinuous galerkin methods*, La Matematica, 1 (2022), pp. 53–84.
- [136] Y. ZANG, G. BAO, X. YE, AND H. ZHOU, *Weak adversarial networks for high-dimensional partial differential equations*, Journal of Computational Physics, 411 (2020), p. 109409.
- [137] M. D. ZEILER, *Adadelata: an adaptive learning rate method*, arXiv preprint arXiv:1212.5701, (2012).
- [138] H. ZHANG, Y. N. DAUPHIN, AND T. MA, *Fixup initialization: Residual learning without normalization*, in International Conference on Learning Representations, PMLR, 2019.
- [139] J. ZHOU, G. CUI, S. HU, Z. ZHANG, C. YANG, Z. LIU, L. WANG, C. LI, AND M. SUN, *Graph neural networks: A review of methods and applications*, AI open, 1 (2020), pp. 57–81.