Institute for Advanced Studies
in Basic Sciences
Gava Zang, Zanjan, Iran

Solving Partial Differential Equations with Uncertainties Using Neural-Networks

Applied Mathematics - Numerical Analysis
Master's Thesis

Sajed Zarrinpour Nashroudkoli

Supervisor: Khadije Nedaiasl

Advisor: Parvin Razzaghi

Department of Mathematics

September 2020

Acknowledgments

Foremost, I would like to thank my supervisor, Dr. Khadije Nedaiasl for her continuous support during my researches. She did a great deal in encouraging me and helping me researching and writing this thesis.

Besides my supervisor, I would like to thank Dr. Parvin Razaghi and Dr. DehBozorgi. If they did not help me during my research, this thesis may never have been concluded. Also, I have special thanks to my friends, especially Maryam, who supported me in any kind of possible way and I'm sure without them I would never come to this point.

Last but not least, I would like to thank my parents, and my brothers, who helped me both financially and spiritually during my thesis work.

A part of this work had been done during the COVID-19 pandemic, when I forced to live with my brother Sobhan and his wife, Zohre. I have a special thanks for giving to them.

Thanks for all your encouragement.

Abstract

Towards modeling the real-world phenomenons with partial differential equations that involve uncertainties, one of the major difficulties is a set of phenomena known as the curse of dimensionality. Luckily, very often the variability of physical quantities derived from the model can be captured by a few features on the coefficient fields, with what so called model reduction techniques. On the other hand, neural networks are good at finding hidden maps on the data. For example, one can use neural-networks based methods to parametrize the physical quantity of interest as a function of input coefficients. In that case, the representability of such quantity can be justified by viewing the neural networks as performing time evolution to find the solution to the model. Indeed, in this thesis, we review a surrogate forward neural network model used to solve two notable partial differential equations in engineering and physics. Also, we explain possibilities that neural networks comes forward throw looking at the mathematical analysis of a well known method, namely finite element method.

Keywords: *Partial Differential Equations, Finite Difference Method, Finite Element Method, Uncertainty Quantification*

Contents

Chapter 1

Introduction

This chapter begins by introducing the basic concepts required for the rest of the thesis, such as uncertainty quantification and neural networks. It follows by introducing two partial differential equations that we want to solve. Furthermore, a brief report of the results is given. This chapter is concluded by an overview of the structure of the thesis.

1.1 Background & Motivation

To model real-world phenomenons, one makes assumptions to simplify his model. More often than not, the hidden parameters that take part in the model are unknown. In many other cases, the exact value of the inputs is unclear. These are examples of what called uncertainty. Usually, one wants to quantify these uncertainties to control them, if possible.

Often, models are consist of a single or a system of Partial Differential Equations (PDEs). Then it is beneficial to study methods to solve PDEs that contain uncertainties in their structures. However, incorporating uncertainty would increase both complexity and computational time.

Our interests come from modeling the mechanical behavior of soft tissue, breast in

particular, under compression. Due to its importance in diagnosing and treatment of breast cancer, it is an exciting subject for many [1, 2, 3, 4, 5, 6, 7]. Ultimately, we are after answering this question: 'How can we make it both patient-specific and efficient to use in actual clinics?'.

The first challenge is that different bodies have different properties, e.g., elasticity. To our knowledge, there is no definite way to determine the exact value of these properties yet. Then they can be thought of as uncertain coefficients in the model. The second challenge is that the results of such modeling must be computable in clinical time.

On the other hand, it is promising to use machine learning for numerical approximations [8, 9, 10, 11, 12, 13]. As an example, for being saved from the curse of dimensionality, one may compute the solution of a PDE on a mesh, then feed that answer to a neural network and use it to find the solution on a finer mesh. Another may find a surrogate forward model which describes a map between the coefficient field to the quantities of interests rather than solving the PDE itself.

To divide the problem into subproblems, we decide to consider solving PDEs with uncertainties using neural networks. In this thesis, we will study the use of such a method on two PDEs to cover both linear and nonlinear cases.

1.2 Partial Differential Equations

The idea behind the partial differential equation is simple; to describe the variation of a physical quantity that depends on multiple physical variables, one has to find its partial variation with respect to each physical variable while keeping others constant. Then, summing up all the variations shall give the overall variation of that quantity. Then, a partial differential equation is a mathematical equation that involves two or more independent variables, an unknown function (which depends on those variables), and

partial derivatives of the unknown function with respect to the independent variables. The order of a partial differential equation is the order of the highest derivative involved. A solution (or a particular solution) to a partial differential equation is a function that solves the equation or, in other words, turns it into an identity when substituted into the equation. A solution is called general if it contains all particular solutions of the equation concerned [14].
PDEs have been used to formulate the solution of physical problems involving functions of several variables mathematically such as the propagation of heat or sound, fluid flow, elasticity, electrostatics, electrodynamics, etc [14].

If all the terms of a PDE contain the dependent variable or its partial derivatives, then such a PDE is called inhomogeneous partial differential equation or homogeneous otherwise. The external forces to the PDE models have applied through the inhomogeneous term.

One needs additional information about the initial state of the model or change of it over the boundary called conditions to find the unique solution of a PDE. There are two types of conditions, initial conditions, and boundary conditions. An initial condition expresses the value of the solution or its derivatives at an initial point in time. A boundary condition, on the other hand, defines the value of the solution or its derivatives at the boundary of the domain of the problem.
Boundary conditions can be categories into three categories; *Dirichlet* boundary condition defines the value of the exact solution. *Neumann* boundary condition describes the values of the derivatives of the exact solution. Finally, *Robin* boundary condition describes both the solution and the derivatives. Note that the Robin boundary condition can formulate as a linear combination of the Dirichlet and Neumann boundary conditions.

1.3 Uncertainty Quantification

As mentioned in Section (1.1), one aspect of this work is Uncertainty Quantification (UQ). This section gives an overview of what is 'uncertainty quantification' and how can it be applied to models.

"UQ is the end-to-end study of the reliability of scientific inference... one is interested in relationships between pieces of information, not the 'truth' of those information/assumptions... UQ cannot tell you that your model is 'right' or 'true', but only that, if you accept the validity of the model (to some quantified degree), then you must logically accept the validity of certain conclusions (to some quantified degree) "[15]. There are two types of uncertainty; uncertainty about an inherently variable phenomenon known as 'Aleatoric' and uncertainty emerging from the lack of knowledge known as 'Epistemic'. Epistemic uncertainty concerns the correctness of the structure of the model itself. Aleatoric uncertainty, on the other hand, concerns the correct values of the parameters of the model. In a broad view, a common UQ task is seeking one or more of these objectives

i. The *forward propagation* or *push-forward* problem;

ii. The *reliability* or *certification* problem;

iii. The *prediction* problem;

iv. The *inverse* problem;

v. The *model reduction* or *model calibration* problem;

UQ in applications often involves the study of PDEs with the random coefficient field. To understand the behavior of a system in the presence of uncertainties, one can extract PDE-derived physical quantities as functionals of their coefficient fields. Even with a suitable discretization of the PDE domain, and the range of the random

variables, this can potentially need solving PDE an exponential number of times numerically. Fortunately, in most PDE applications these functionals often depend only on a few characteristic "features" of the coefficient fields, enabling them to be determined from solving PDE a limited number of times.

1.4 Artificial Neural Network

Inspired by how the brain works, in 1943, Warren McCulloch and Walter Pitts [16] wrote a paper on how neurons might work. They modeled a simple neural network with electrical circuits. Although the study of the human brain is thousands of years old, this was the beginning of a new field that evolve into what we know today as Artificial Neural Networks (ANN).

The short history of the development of deep neural networks goes as follow:

"In the late 1940s, D. O. Hebb created a learning hypothesis that became known as Hebbian learning [17]. The first functional networks with many layers called 'Group Method of Data Handling' were created by Ivakhnenko and Lapa in 1965 [18, 19, 20]. The basics of continuous backpropagation [18, 21, 22] were derived in the context of control theory by Kelley [23] in 1960 and by Bryson in 1961, using principles of dynamic programming [24].

In 1970, Seppo Linnainmaa published the general method for Automatic Differentiation (AD) of discrete connected networks of nested differentiable functions [25, 26]. Geoffrey Hinton et al. (2006) proposed learning a high-level representation using successive layers of binary or real-valued latent variables with a restricted Boltzmann machine [27] to model each layer. In 2012, Ng and Dean created a network that learned to recognize higher-level concepts, such as cats, only from watching unlabeled images [28]. Unsupervised pre-training and increased computing power from Graphics Processing

Units (GPU) and distributed computing allowed the use of larger networks, particularly in the image and visual recognition problems, which became known as "deep learning". Neural Networks, as we know them today, are network or circuit of real biological neurons or their artificial counterparts. In the latter case, we call it artificial neural network. In this thesis, we are working strictly on ANN and we might use the term Neural Network (NN) for simplicity.

ANNs are composed of artificial neurons that hold the biological concept of neurons that are receiving input, combine it with their internal state and turn on or off. They do that using weights, bias and activation function.

The weights have used to model the connections of the biological neurons. A **weight** is "a parameter associated with a connection from one neuron M, to another neuron N. It corresponds to a synapse in a biological neuron, and it determines how much notice the neuron N pays to the activation it receives from neuron M. If the weight is positive, the connection is called excitatory, while if the weight is negative, the connection is called inhibitory" [29].

"In feedforward and some other neural networks, each hidden unit, and each output unit is connected via a trainable weight to a unit (the **bias** unit) that always has an activation level of 1. This gives a trainable threshold equal to the value of the weight from the bias unit to each hidden or output unit" [29]. Moreover, **activation function** "is the function that describes the output behavior of a neuron. Most network architecture starts by computing the weighted sum of the inputs (that is, the sum of the product of each input with the weight associated with that input). This quantity, the total net input, is then usually transformed in some way, using what is sometimes called a squashing function. The simplest squashing function is a step function: if the total net input is less than 0 (or more generally, less than some threshold T) then the output of the neuron is 0, otherwise, it is 1. A common squashing function is a logistic function. In summary, the activation function is the result of applying a squashing

function to the total net input" [29]. The important characteristic of the activation function is that it provides a smooth, differentiable transition as input values change, i.e. small changes in input would produce small changes in output.

Mathematically, we can model a neuron as follow:

$$\psi(XW + b). \tag{1.1}$$

Here, X is the input vector, W is the weight matrix, b is the bias and finally, ψ is the activation function. With this notation, the final output of our feedforward network would be:

$$\Phi(X, W^1, \ldots, W^K) = \psi_k(\psi_{k-1}(\ldots \psi_2(\psi_1(XW^1 + b^1)W^2 + b^2) \ldots W^{K-1} + b^{K-1})W^K + b^K). \tag{1.2}$$

The initial inputs of the ANN are external data, such as images and documents. The ultimate output is to accomplish the task, such as recognizing an object within the image.

Two general types of learning exist; learning from the data associated with labels which are called supervised learning and learning from unlabeled data which is called unsupervised learning. In this thesis, we would do supervised learning.

In general, the neural network would go through a process that we call learning. Learning, in case of supervised learning is to minimize a function like $\mathscr{L}(Y, \Phi(X, W^1, \ldots, W^K))$ (i.e. **loss** function e.g. mean square error) that defines a relation between the network output $\Phi(X, W^1, \ldots, W^K)$ and the expected network output Y, which is the labels associated with the data:

$$\min_{\{W^k\}_{k=1}^K} \mathscr{L}(Y, \Phi(X, W^1, \ldots, W^K)). \tag{1.3}$$

If the network were unable to minimize the loss function, we say that **underfitting**

happened.

To ensure that the network learned correctly, we have to test it against a fresh dataset which it hasn't see before, which called the test dataset. What we expect is an acceptable result from our network over this dataset concerning some metrics (e.g mean absolute error) which if satisfied, we say that the network learned successfully. If not, we would say that **overfitting** happened, which means that the network was unable to generalize the hidden relation of the training dataset (i.e. the provided data set in learning phase) to the test dataset though it has perfect results on the training dataset itself.

The learning process begins by feeding a **batch** of train data to the network. Next, the network would try to adjust it's weighted so that it minimizes loss function and after that, the next batch would feed the network. Each passes through all of the train data would be called an **epoch**. The whole learning process would often consist of many epochs. But one must be careful about overfitting the data with too many epochs.

Now, we would like to discuss the questions which this thesis is based upon. 'Why one may interested in solving a PDE, especially in the presence of uncertainty, using neural network, and how?' To highlight the important aspect of the idea of using a neural network for solving PDEs, consider these questions. Would it be nice if we can solve our problem on a bigger mesh to train our network and then use it to compute the solution over a finer mesh? How about solving problems with high dimensions in which the classical numerical methods such as finite element method and finite difference method suffer the curse of dimensionality? As we would see in the next section, computing the basis function over nodal points is the source of heavy computational cost in the finite element method. What if we would be able to bypass the need to compute them? These are just some of the applications which one may find when importing neural network as a tool in numerical analysis.

There are two different choices to generate data sets when dealing with PDEs. First,

solve the PDE over a bigger mesh to obtain the train data set and then solving the PDE over a finer mesh to generate test dataset. Second, picking a percentage of the data points concerning normal distribution over the domain. Moreover, to approach the problem, methods can be categorized into three categories. First, we know the PDE and its coefficients. What we seek is to solve the PDE directly using neural networks. In that regard, one may use the Ritz variational formulation of the PDE to define the lost function and uses the Stochastic Gradient Decent (SGD) optimization algorithm as the optimizer. This way, a numerical approximation of the solution can be calculated much like any other numerical methods [30], or else, one may use a coarse grid to train the network and then use the trained model on a fine grid mesh [31]. Second, the PDE is known but the coefficients are unknown [32, 33]. Finally, neither PDE nor it's coefficients are known [34].

Consider some PDEs with the random coefficient field, we are interested in finding our physical quantities as functionals which depend only on a few characteristic "features" of the coefficient fields through our neural network. A dimension reduction technique based on neural network representation has used to do regression.

The fundamental task of regression seeks to find a function h_θ parameterized by a parameter vector $\theta \in \mathbb{R}^p$ such that

$$f(\theta) \approx h_\theta(a), a \in \mathbb{R}^q \tag{1.4}$$

However, choosing a sufficiently large class of approximation functions without the issue of over-fitting remains a delicate business. For example when choosing the set of basis $\{\phi_k(a)\}$ such that $f(a) = \sum_k \beta_k \phi_k(a)$ in linear regression.

A key advantage of using neural network is that it bypasses the traditional need to handcraft basis for spanning $f(a)$ as in linear regression. Instead, it directly learns an approximation that satisfies (1.4) in a data-driven way. More precisely, we want to learn

$f(a)$ that maps the random coefficient vector a in a PDE to some physical quantities described by the PDE.
The approach is conceptually simple, consisting of the following steps:

i. Sample the random coefficients (a in (1.4)) of the PDE from a user-specified distribution. For each set of coefficients, solve the deterministic PDE to obtain the physical quantity of interest ($f(a)$ in (1.4)).

ii. Use a neural network as the surrogate model $h_\theta(a)$ in (1.4) and train it using the previously obtained samples.

iii. Validate the surrogate forward model with more samples.

We note that our work is a report of [33]. In this thesis, the function that we want to parameterize is over the coefficient field of the PDE. It would be beneficial to mention other works which try to solve deterministic PDE numerically using a neural network [30, 35, 36, 37, 38], and [39] where a deterministic PDE is solved as a stochastic control problem using neural network.

1.5 An Overview of Classical Numerical Methods

One of the popular numerical method for solving PDEs is Finite Difference Method (FDM). It is based on the idea of substituting the derivatives in the equation with their approximations. To do so, one may define a mesh, write down the value of each partial derivative on each nodal point of the mesh and then calculating a numerical approximation for the solution of the partial differential equation. The precision of the method has a direct relation with the order of the precision which we chose to approximate the derivatives in and it can be computed through the Taylor expansion. Though the FDM

is a good method and it is widely used, it has its downsides too. An important one is that the FDM cannot be used on complex domains [40, 41, 42].

Finite Element Method (FEM), uses more complex mesh structures and it can be used on more complex domains. It is based on the variational formulation of the equation. There are two kinds of finite element methods: **Ritz** formulation in which we would write an equivalent minimization problem and we would solve this new problem instead, and **Galerkin** formulation in which we would use the basics of integration by part and Green's identities to reduce the order of the differential equation and then we try to solve this new problem. Either way, we would write an equivalent problem to the desired differential equation using basis functions. After we achieved the variational formulation, we would approximate its solution by reducing the infinite dimension of the solution space to a finite dimension. Hence, we call the method, **finite element method** [43, 44].

The finite element method achieved a great deal of success in general and is very popular among mathematicians and engineers, but there is some downside to it as well. We cannot use it for high dimensional problems, which often in literature referred to as *the curse of dimensionality*. Moreover, refinement of the mesh would result in increasing the nodal points count which means an increase in computational cost. Finally, we can explore more domains with the finite element method, yet the domain has to be Lipschitz. We would study finite element method in more details in Section (2.2) [45, 46]. For the enthusiastic reader, here is a list of some other numerical methods: wavelet method [47, 48], finite volume method [49, 50] and meshfree method [51, 52].

1.6 Problem Statement and Contributions

As we saw in Section (1.2), PDEs come to life to express the relations between changing quantities. In this section, we discuss a bit about the problems which we want to solve. Suppose we have an inhomogeneous media (an inhomogeneous media is a medium in which all properties of interest are not the same at any point). Also, suppose that we have different values for conductance for different materials, which composed our media, modeled by $a(x)$. Given the fixed direction vector $\xi \in \mathbb{R}^d$, we are interested to know the effective conductance through the media in that direction. Mathematically, we want to find the solution to the following minimization problem:

$$A_{\text{eff}}(a) = \min_{u(x)} \int_{[0,1]^d} a(x)||\nabla u(x) + \xi||_2^2 \mathrm{d}x. \tag{1.5}$$

Now, for the second problem, we want to know the ground state energy of an electron along with its spatial distribution. Ground state energy is the state which electron wants to be in when the temperature drops to zero. Mathematically, it is the smallest eigenvalue of the problem of the form

$$Hu = Eu, \tag{1.6}$$

which in case of multi electrons, is

$$Eu(r) := [-\Delta + v(r) + \int w(r - r^*)|u(r^*)^2|dr^*]u(r). \tag{1.7}$$

In (1.6), H is a Hamiltonian operator which is the sum of the kinetic energies of all the particles, plus the potential energy of the particles associated with the system.

Considering multiple particles, our second equation

$$-\Delta u(x) + a(x)u(x) + \sigma u(x)^3 = E_0 u(x),\ x \in [0,1]^d,\ s.t. \int_{[0,1]^d} u(x)^2 \mathrm{d}x = 1, \qquad (1.8)$$

can be derived from (1.6). In (1.8), $-\Delta u(x)$ represents the kinetic energy, $a(x)$ is the potential.

The main contributions of this work are:

i. Providing theoretical guarantees on the neural network representation of $f(a)$ in (1.4) through explicit construction for the parametric PDE problems under study;

ii. Showing that even a rather simple neural network architecture can learn a good representation of $f(a)$ in (1.4) through training.

1.7 Results

1.7.1 Effective Conductance in Inhomogeneus Media

In 1D, mean squared error measured to be 2.5793×10^{-5} on the training set and 5.20×10^{-6} on test set. Finally, the predicted effective conductance by the network was 0.76800650 with the error $\|mean(\mathscr{A}_{\text{eff-predicted}}) - mean(\mathscr{A}_{\text{eff-exact}})\| = 1.02100 \times 10^{-3}$.

1.7.2 Nonlinear Shrödinger Equation

In 2D, the mean squared error measured to be 0.0173 on the training set and 0.01425044 on the test data set. Finally, the mean of predicted ground state energy by the network was 10.17474556 which has 7.235×10^{-5} L^2-distance from the mean of the normalized ground state energy provided.

1.8 Thesis Structure

The rest of this thesis is structured as follows. In Chapter 2, the mathematical preliminaries are presented. In Chapter 3, we propose our discretization and neural model for the problems. In Chapter 4 we use the proposed model in Chapter 3 to compute the solution and report the results and conclude.

Chapter 2

Mathematical Preliminaries

In this chapter, the required mathematical background to understand the rest of the thesis has discussed. It begins with functional analysis. Then, It goes over the mathematics of the finite element method to demonstrate the necessity of developing new methods that can perform better than our classical methods. This chapter has been concluded by the universal approximation theorem that ensures the existence of a neural network that approximates the solution.

2.1 Functional Analysis

This section begins by introducing measures and follows by Lebesgue integration and functional spaces. This section has heavily influenced by [15]. Interested readers may go further by reading [15, 53].

2.1.1 Measure Spaces

Sample spaces, which are abstract sets, are building blocks of measure and probability theory. One can distinguish certain subsets of these sample spaces as being 'measurable'

and assign to each of them a numerical notion of 'size'.

Definition 2.1. A *measurable space* is a pair $(\mathscr{X}, \mathscr{F})$, where

i. $\mathscr{X}$ is a set, called the sample space;

ii. $\mathscr{F}$ is a $\sigma - algebra$ on $\mathscr{X}$, i.e. collection of subsets of $\mathscr{X}$ containing ϕ and closed under countable applications of the operations of union, intersection, and complementation relative to $\mathscr{X}$; elements of $\mathscr{F}$ are called measurable sets or events.

Definition 2.2. i. A *signed measure* (or *charge*) on a measurable space $(\mathscr{X}, \mathscr{F})$ is a function $\mu : \mathscr{F} \to \mathbb{R} \bigcup \{\pm\infty\}$ that takes at most one of the two infinite values, has $\mu(\phi) = 0$, and, whenever $E_1, E_2, \cdots \in \mathscr{F}$ are pairwise disjoint with union $E \in \mathscr{F}$, then $\mu(E) = \sum_{n\in N} \mu(E_n)$. In the case that $\mu(E)$ is finite, we required that the series $\sum_{n\in N} \mu(E_n)$ converges absolutely to $\mu(E)$.

ii. A *measure* is a signed measure that does not take negative values.

Remark 1. The triple $(\mathscr{X}, \mathscr{F}, \mu)$ is called a signed measure space or measure space as appropriate. The sets of all signed measures and measures on $(\mathscr{X}, \mathscr{F})$ are denoted $\mathscr{M}_{\pm}(\mathscr{X}, \mathscr{F})$ and $\mathscr{M}_{+}(\mathscr{X}, \mathscr{F})$ respectively.

Definition 2.3. Let $(\mathscr{X}, \mathscr{F}, \mu)$ be a measure space.

i. If $N \subseteq \mathscr{X}$ is a subset of a measurable set $E \in \mathscr{F}$ such that $\mu(E) = 0$, then N is called a μ-null set.

ii. If the set of $x \in \mathscr{X}$ for which some property $P(x)$ does not hold is μ-null, then P is said to hold μ-almost everywhere (or, when μ is a probability measure, μ-almost surely).

Definition 2.4. Let $(\mathscr{X}, \mathscr{F})$ and $(\mathscr{Y}, \mathscr{G})$ be measurable spaces. A function $f : \mathscr{X} \to \mathscr{Y}$ generates a $\sigma - algebra$ on $\mathscr{X}$ by

$$\sigma(f) := \sigma([f \in E] | E \in \mathscr{G}),$$

and f is called a *measurable function* if $\sigma(f) \subseteq \mathscr{F}$. That is, f is measurable if the pre-image $f^{-1}(E)$ of every $\mathscr{G}$-measurable subset E of Y is an $\mathscr{F}$-measurable subset of $\mathscr{X}$. A measurable function whose domain is a probability space is usually called a *random variable*.

2.1.2 Lebesgue Integration

The integration of a measurable function with respect to a (signed or non-negative) measure is referred to as *Lebesgue integration*. Despite many technical details in its construction, it is the integral of choice for most applications; it extends the simple Riemann integral of functions of a single real variable, can handle worse singularities than the Riemann integral, has better convergence properties, and also naturally captures the notion of the expected value in probability theory.
The construction of the Lebesgue integral has accomplished in three steps:

Step i. Calculating Lebesgue integral over simple functions; the integral is defined for simple functions, which are analogous to step functions from elementary calculus, except that their plateaus are not intervals in $\mathbb{R}$ but measurable events in the sample space.

Definition 2.5. Let $(\mathscr{X}, \mathscr{F}, \mu)$ be a measure space. The indicator function $\mathbb{I}_E$ of a set

$E \in \mathscr{F}$ is the measurable function defined by

$$\mathbb{I}_E := \begin{cases} 1, & \text{if } x \in E \\ 0, & \text{if} x \notin E. \end{cases}$$

A function $f : \mathscr{X} \to \mathbb{K}$ is called simple if

$$f = \sum_{i=1}^{n} \alpha_i \mathbb{I}_{E_i}$$

for some scalars $\alpha_1, \ldots, \alpha_n \in \mathbb{K}$ and some pairwise disjoint measurable sets $E_1, \ldots, E_n \in \mathscr{F}$ with $\mu(E_i)$ finite for $i = 1, \ldots, n$. The Lebesgue integral of a simple function $f := \sum_{i=1}^{n} \alpha_i \mathbb{I}_{E_i}$ is defined to be

$$\int_{\mathscr{X}} f \mathrm{d}\mu := \sum_{i=1}^{n} \alpha_i \mu(E_i).$$

Step ii. Calculating the integral of non-negative measurable functions using simple functions.

Definition 2.6. Let $(\mathscr{X}, \mathscr{F}, \mu)$ be a measure space and let $f : \mathscr{X} \to [0, +\infty]$ be a measurable function. The Lebesgue integral of f is defined to be

$$\int_{\mathscr{X}} f \mathrm{d}\mu := \sup \left\{ \int_{\mathscr{X}} \phi \mathrm{d}\mu \middle| \begin{array}{l} \phi : \mathscr{X} \to \mathbb{R} \text{ is a simple function, and} \\ 0 \leq \varphi(x) \leq f(x) \text{ for } \mu\text{-almost all } x \in \mathscr{X} \end{array} \right\}$$

Step iii. The integral of a real- or complex-valued function is defined through integration of positive and negative real and imaginary parts, with care being taken to avoid the undefined expression '$\infty - \infty$':

Definition 2.7. Let $(\mathscr{X}, \mathscr{F}, \mu)$ be a measure space and let $f : \mathscr{X} \to \mathbb{R}$ be a measurable function. The Lebesgue integral of f is defined to be

$$\int_{\mathscr{X}} f \mathrm{d}\mu := \int_{\mathscr{X}} f_+ \mathrm{d}\mu + \int_{\mathscr{X}} f_- \mathrm{d}\mu$$

provided that at least one of the integrals on the right-hand side is finite. The integral of a complex-valued measurable function $f : \mathscr{X} \to \mathbb{C}$ is defined to be

$$\int_{\mathscr{X}} f \mathrm{d}\mu := \int_{\mathscr{X}} (\mathrm{Re} f) \mathrm{d}\mu + \int_{\mathscr{X}} (\mathrm{Im} f) \mathrm{d}\mu.$$

It is worthy to define the spaces of Lebesgue-integrable functions which are ubiquitous in analysis.

Definition 2.8. Let $(\mathscr{X}, \mathscr{F}, \mu)$ be a measure space. For $1 \leq p \leq \infty$, the L^p space (or Lebesgue space) is defined by

$$L^p(\mathscr{X}, \mu; \mathbb{K}) := \{f : \mathscr{X} \to \mathbb{K} | f \text{ is measurable and } \|f\|_{L^p(\mu)} \text{ is finite}\}.$$

For $1 \leq p \leq \infty$, the norm is defined by the integral expression

$$\|f\|_{L^p(\mu)} := \left(\int_{\mathscr{X}} |f(x)|^p \mathrm{d}\mu(x) \right)^{1/p}. \tag{2.1}$$

To be more precise, $L^p(\mathscr{X}, \mu; \mathbb{K})$ is the set of equivalence classes of such functions, where functions that differ only on a set of μ-measure zero are identified.

Example 2.9. A particular important case corresponds to taking $p = 2$, then the inner product would be

$$\langle u, v \rangle := \big(\int_{\Omega} |u(x)|^2 \mathrm{d}x \big)^{1/2}.$$

Clearly, $\|u\|_{L^2(\Omega)} = \langle u, u \rangle^{1/2}$.

2.1.3 Sobolev Spaces

Definition 2.10. A *multi-index* α is an n-tuple $\alpha = (\alpha_1, \ldots, \alpha_n)$, used to concisely denote the partial differential operator

$$D^\alpha(u) := \frac{\mathrm{d}^{|\alpha|}}{\mathrm{d}x_1^{\alpha_1} \ldots \mathrm{d}_x^{\alpha_n}}(u). \tag{2.2}$$

We define $|\alpha| = \alpha_1 + \cdots + \alpha_n$ to be the *degree* of α.

Definition 2.11. For two multi-indices α, β we define the following:

i. $\alpha \leq \beta$ if $\alpha_i \leq \beta_i$, for all $1 \leq i \leq n$

ii. If $\alpha \leq \beta$, we define $\alpha - \beta := \gamma$, where $\gamma = (\alpha_1 - \beta_1, \ldots, \alpha_n - \beta_n)$

iii. $\alpha! = \alpha_1! \ldots \alpha_n!$

Notation 2. For the remainder of this subsection, let Ω be a bounded open subset of $\mathbb{R}^n$. In general, Ω would be the domain of the PDE which we want to solve.

Definition 2.12. Weak Derivative. Let $\Omega \subset \mathbb{R}^n$, $u, v \in L^p(\Omega)$, and α be a multi-index. Then v is the weak α-th partial derivative of u if:

$$\int_\Omega u D^\alpha \phi \mathrm{d}x = (-1)^{|\alpha|} \int_\Omega v \phi \mathrm{d}x$$

for all $\phi \in \mathscr{C}_0^\infty$.

Definition 2.13. Let $1 \leq p \leq \infty$ and k be a non-negative integer. The *Sobolev Space* $\mathscr{W}^{k,p}(\Omega)$ consists of all functions $u : \Omega \to \mathbb{R}, u \in L^p_{loc}(\Omega)$ such that each weak derivative $D^\alpha u$ with $|\alpha| \leq k$ exists and belongs to L^p. That is,

$$\mathscr{W}^{k,p}(\Omega) = \{u \in L^p_{loc}(\Omega) | \|D^\alpha u\|_{L^p(\Omega)}, \forall |\alpha| \leq k\} \tag{2.3}$$

Similarly, the space $\mathscr{W}^{k,p}_{loc}(\Omega)$ consists of the functions u as above for which $D^\alpha u$ with $|\alpha| \leq k$ exists and belongs to $L^p_{loc}(\mathscr{V})$, where $\mathscr{V}$ is an arbitrary compact subset of Ω.

Remark 3. Note that $\mathscr{W}^{k,p}(\Omega) = \{u \in L^p_\Omega \mid D^\alpha u \in L^p_\Omega, \, |\alpha| \leq k\}$.

Theorem 2.14. *Assume $u, v \in \mathscr{W}^{k,p}(\Omega)$, $|\alpha| \leq k$. Then*

i. $D^\alpha(\Omega) \in \mathscr{W}^{k-|\alpha|,p}(\Omega)$ and $D^\beta(D^\alpha u) = D^\alpha(D^\beta u) = D^{\alpha+\beta}u$ for all multi-indices α, β satisfying $|\alpha| + |\beta| \leq k$.

ii. For each $\lambda, \mu \in \mathbb{R}$, $\lambda u + \mu v \in \mathscr{W}^{k,p}(\Omega)$ and $D^\alpha(\lambda u + \mu v) = \lambda D^\alpha u + \mu D^\alpha v$, $|\alpha| \leq k$.

Definition 2.15. We define the following norm on $u \in \mathscr{W}^{k,p}(\Omega)$:

$$\|u\|_{\mathscr{W}^{k,p}(\Omega)} := \begin{cases} \left(\sum_{|\alpha|\leq k} \int_\Omega |D^\alpha u|^p \mathrm{d}x\right)^{1/p}, & \text{if } 1 \leq p \leq \infty \\ \sum_{|\alpha|\leq k} \operatorname*{ess\,sup}_\Omega |D^\alpha u|, & \text{if} p = \infty. \end{cases}$$

The following theorem states that Sobolev spaces are complete:

Theorem 2.16. *Let $\Omega \in \mathbb{R}^n$ be an open bounded set and $1 \leq p \leq \infty$. Then*

i. the space $\mathscr{W}^{k,p}(\Omega)$ is a Banach space with respect to the norm $\|\cdot\|_{\mathscr{W}^{k,p}}$

ii. the space $H^1(\Omega) := \mathscr{W}^{1,2}(\Omega)$ is a Hilbert space with inner product

$$\langle u, v\rangle := \int_\Omega uv dx + \sum_{i=1}^N \int_\Omega \frac{\partial u}{\partial x_i}\frac{\partial v}{\partial x_i} dx.$$

Lemma 4. Given $u \in \mathscr{W}^{k,p}(\Omega)$ and $v \in \mathscr{C}^\infty_0(\Omega)$, $uv \in \mathscr{W}^{k,p}(\Omega)$.

2.2 Finite Element Method

This section would review the finite element method. Here, The goal is to highlight the limitations of this method. For interested readers, we suggest reading finite element resources [43, 44]. Moreover, one may be interested in using parallel programming to enhance finite element methods performance [54]. However, one needs to question the nature of the method itself to find a way to improve it.

Notation 5. Let Ω be an open set in $\mathbb{R}^n$ and let $k \in \mathbb{N}$. We denote by $\mathscr{C}^k(\Omega)$ the set of all continuous real-valued functions defined on Ω such that $D^\alpha u$ is continuous on Ω for all $\alpha = (\alpha_1, \ldots, \alpha_n)$ with $|\alpha| \leq k$. Assuming that Ω is a bounded open set, $\mathscr{C}^k(\Omega)$ will denote the set of all u in $\mathscr{C}^k(\Omega)$ such that $D^\alpha u$ can be extended from Ω to a continuous function on $\bar{\Omega}$, the closure of the set Ω, for all $\alpha = (\alpha_1, \ldots, \alpha^n)$, $|\alpha| \leq k$. $\mathscr{C}^k(\bar{\Omega})$ can be equipped with the norm

$$\|u\|_{\mathscr{C}^k(\bar{\Omega})} := \sum_{|\alpha| \leq k} \sup_{x \in \Omega} |D^\alpha u(x)|.$$

In particular when $k = 0$ we shall write $\mathscr{C}(\bar{\Omega})$ instead of $\mathscr{C}^0(\bar{\Omega})$ to denote the set of all continuous functions defined on $\bar{\Omega}$; in this case,

$$\|u\|_{\mathscr{C}(\bar{\Omega})} = \sup_{x \in \Omega} |u(x)| = \max_{x \in \bar{\Omega}} |u(x)|.$$

Similarly, if $k = 1$,

$$\|u\|_{\mathscr{C}(\bar{\Omega})} = \sum_{|\alpha| \leq 1} \sup |D^\alpha u(x)| = \sup_{x \in \Omega} |u(x)| + \sum_{j=1}^{n} \sup_{x \in \Omega} |\frac{\partial u}{\partial x_j}(x)|.$$

Definition 2.17. The support of a continuous function u defined on an open set $\Omega \subset \mathbb{R}^n$ is defined as the closure in Ω of the set $x \in \Omega : u(x) \neq 0$. We shall write *supp* u for the

support of u. Thus, support of function u is the smallest closed subset of Ω such that $u = 0$ in $\Omega \setminus \text{supp } u$.

We denote by $\mathscr{C}_0^k(\Omega)$ the set of all u contained in $\mathscr{C}^k(\Omega)$ whose support is a bounded subset of Ω. Let $\mathscr{C}_0^\infty = \cap_{k\geq 0}\mathscr{C}_0^k(\Omega)$.

Lemma 6. *(The Cauchy-Schwarz inequality).* Let u and v belong to L^2_Ω; then $uv \in L^1_\Omega$ and

$$| \langle u, v \rangle | \leq \|u\|_{L^2_\Omega} \|v\|_{L^2_\Omega}.$$

Corollary 7. *(The triangle inequality).* Let u and v belong to L^2_Ω; then $u + v \in L^2_\Omega$, and

$$\|u + v\|_{L^2_\Omega} \leq \|u\|_{L^2_\Omega} + \|v\|_{L^2_\Omega}.$$

Definition 2.18. Let $\mathscr{X}$ be a linear space with norm $\|\cdot\|_{\mathscr{X}}$. $\mathscr{X}$ is called a *Banach space* if, whenever $\{u_m\}_{m=1}^\infty$ is a sequence of elements of $\mathscr{X}$ such that

$$\lim_{n,m\to\infty} \|u_n - u_m\|_{\mathscr{X}} = 0,$$

there exists $u \in \mathscr{X}$ such that $\lim_{m\to\infty} \|u - u_m\|_{\mathscr{X}} = 0$ (i.e. the sequence $\{u_m\}_{m=1}^\infty$ converges to u in $\mathscr{X}$). Furthermore, such a sequence is called a *Cauchy sequence*.

Remark 8. Every Banach space equipped with an inner product is a *Hilbert* space. Note that if $\mathscr{H}$ is Hilbert and $u \in \mathscr{H}$, then $\|u\|_{\mathscr{H}} = \langle u, u \rangle^{1/2}$

Remark 9. The space L^p_Ω with $p \in [1, \infty]$ is a Banach space. In particular, L^2_Ω is a Hilbert space.

Remark 10. The Hilbert space $\mathscr{H}_0^1(\Omega)$ has significant importance in our context. It defines as follows:

$$\mathscr{H}_0^1(\Omega) = \left\{ u \in L^2(\Omega) | \frac{\partial u}{\partial x_i} \in L^2(\Omega),\ i = 1, \ldots, n,\ u = 0 \text{ on } \partial\Omega \right\}.$$

2.2.1 Weak Solutions to Elliptic Problems

Let Ω be a bounded open set in $\mathbb{R}^n$, and consider the linear second-order partial differential equation

$$-\sum_{i,j=1}^{n} \frac{\partial}{\partial x_j}\left(a_{i,j}(x)\frac{\partial u}{\partial x_i}\right) + \sum_{i=1}^{n} b_i(X)\frac{\partial u}{\partial x_i} + c(x)u = f(x),\ x \in \Omega \tag{2.4}$$

$$u = 0 \text{ on } \partial\Omega \tag{2.5}$$

where $\partial\Omega$ indicates the boundary of the domain Ω,

$$a_{i,j} \in \mathscr{C}^1(\bar{\Omega}), \quad i,j = 1,\ldots,n; \quad b_i \in \mathscr{C}(\bar{\Omega}), \quad i = 1,\ldots,n; \quad c \in \mathscr{C}(\bar{\Omega}), \quad f \in \mathscr{C}(\bar{\Omega})$$

and

$$\sum_{i,j=1}^{n} a_{i,j}(x)\zeta_i\zeta_j \geq \tilde{c}\sum_{i=1}^{n}\zeta_i^2, \quad \forall \zeta = (\zeta_1,\ldots,\zeta_n) \in \mathbb{R}^n, \quad x \in \bar{\Omega}, \tag{2.6}$$

here $\tilde{c}$ is a positive constant independent of x and ζ. The condition in (2.6) is usually referred to as *uniform ellipticity* and (2.4) is called an elliptic equation. Here, as 2.5 stated, we took the boundary condition as Dirichlet boundary condition for simplicity.

Definition 2.19. A **classical solution** of (2.4) is any function $u \in \mathscr{C}^2(\Omega) \cap \mathscr{C}(\bar{\Omega})$ which satisfies (2.4) and (2.5).

According to the theory of partial differential equations, the equation (2.4) with boundary condition (2.5) has a unique classical solution, provided that $a_{i,j}, b_i, c, f$ and $\partial\Omega$ are sufficiently smooth. However, in many applications one has to consider equations where these smoothness requirements are violated, and for such problems the classical theory is inappropriate.

Example 2.20. Poissons equation with zero Dirichlet boundary condition on $\Omega =$

$(1,1)^n$ in $\mathbb{R}^n$:

$$\begin{cases} -\Delta u = \operatorname{sgn}(\frac{1}{2} - |x|), & x \in \Omega, \\ u = 0, & x \in \partial\Omega, \end{cases} \qquad (\star)$$

This problem does not have a classical solution, $u \in \mathscr{C}^2(\Omega) \cap \mathscr{C}(\bar{\Omega})$, for otherwise Δu would be a continuous function on Ω, which is not possible because $\operatorname{sgn}(\frac{1}{2}|x|)$ is not continuous on Ω.

In order to overcome the limitations of the classical theory and to be able to deal with partial differential equations with non-smooth data, the idea is to generalise the notion of solution by weakening the differentiability requirements on u.

Definition 2.21. Let $a_{i,j} \in L^\infty(\Omega)$, $i,j = 1,\dots,n$, $b_i \in L^\infty(\Omega)$, $i = 1,\dots,n, c \in L^\infty(\Omega)$, and let $f \in L^2(\Omega)$. A function $u \in \mathscr{H}_0^1(\Omega)$ satisfying

$$\sum_{i,j=1}^{n} \int_\Omega a_{i,j}(x) \frac{\partial u}{\partial x_i} \frac{\partial v}{\partial x_j} \mathrm{d}x + \sum_{i=1}^{n} \int_\Omega b_i(x) \frac{\partial u}{\partial x_i} v \mathrm{d}x + \int_\Omega c(x) u v \mathrm{d}x = \int_\Omega f(x) v(x) \mathrm{d}x, \ \forall v \in \mathscr{H}_0^1(\Omega) \qquad (2.7)$$

is called a **weak solution** of (2.4), (2.5). All partial derivatives in 2.7 should be understood as weak derivatives.

Clearly, if u is a classical solution of (2.4), (2.5), then it is also a weak solution of (2.4), (2.5). However, the converse is not true. If (2.4), 2.5 has a weak solution, this may not be smooth enough to be a classical solution. Before giving an example of such a situation, it would be convenient to have a tool to show the existence of a unique solution to the PDEs.

Theorem 2.22. *(Lax & Milgram theorem). Suppose that $\mathscr{V}$ is a real Hilbert space equipped with norm $\|\cdot\|_{\mathscr{V}}$. Let $a(\cdot,\cdot)$ be a bilinear functional on $\mathscr{V} \times \mathscr{V}$ such that:*

i. $\exists c_0 > 0$ *s.t.* $\forall v \in \mathscr{V} : a(v,v) \geq c_0 \|v\|_{\mathscr{V}}^2$,

ii. $\exists c_1 > 0$ *s.t.* $\forall c, w \in \mathscr{V}$ *:* $|a(w,v)| \leq c_1 \|w\|_{\mathscr{V}} \|v\|_{\mathscr{V}}$,

iii. let $l(\cdot)$ *be a linear functional on* $\mathscr{V}$ *such that :* $\exists c_2 > 0$ *s.t.* $\forall v \in \mathscr{V}$ *:* $|l(v)| \leq c_2 \|v\|_{\mathscr{V}}$.

Then, there exists a unique $u \in \mathscr{V}$ *such that*

$$a(u,v) = l(v), \quad \forall v \in \mathscr{V}.$$

Remark 11. If

$$c(x) - \frac{1}{2} \sum_{i=1}^{n} \frac{\partial b_i}{\partial x_i} \geq 0, \; x \in \bar{\Omega},$$

then has a unique solution $u \in \mathscr{H}_0^1$.

The first step in the construction of a finite element method for an elliptic boundary value problem, (e.g. (2.4), (2.5)) is to convert it into its weak formulation:

$$\text{find } u \in \mathscr{V} \text{ such that } a(u,v) = l(v) \; \forall v \in \mathscr{V}. \tag{P}$$

where $\mathscr{V}$ is the solution space (e.g. $\mathscr{H}_0^1(\Omega)$ for the homogeneous Dirichlet boundary value problem), $a(\cdot,\cdot)$ is a bilinear functional on $\mathscr{V} \times \mathscr{V}$, and $l(\cdot)$ is a linear functional on $\mathscr{V}$. The second step in the construction is to replace $\mathscr{V}$ in (P) by a finite-dimensional subspace $\mathscr{V}_h \subset \mathscr{V}$ which consists of continuous piecewise polynomial functions of a fixed degree associated with a subdivision of the computational domain; then consider the following approximation of (P):

$$\text{find } u_h \in \mathscr{V}_h \text{ such that } a(u_h, v_h) = l(v_h) \; \forall v_h \in \mathscr{V}_h. \tag{P_h}$$

Suppose, for example, that

$$\dim \mathscr{V}_h = N(h) \text{ and } \mathscr{V}_h = \text{span}\{\phi_1, \ldots, \phi_{N(h)}\}$$

where the (linearly independent) basis functions $\phi_i, i = 1, \ldots, N(h)$, have "small" support. Expressing the approximate solution u_h in terms of the basis functions, ϕ_i, we can write

$$u_h(x) = \sum_{i=1}^{N(h)} U_i \phi_i(x), \qquad (\star\star)$$

where $U_i, i = 1, \ldots, N(h)$, are to be determined. Thus, (P_h) can be written as follows:

$$\text{find } (U_1, \ldots, U_{N(h)}) \in \mathbb{R}^{N(h)} \text{ s.t. } \sum_{i=1}^{N(h)} a(\phi_i, \phi_j) U_i = l(\phi_i),\ j = 1, \ldots, N(h). \qquad (P_h')$$

This is a system of linear equations for $U = (U_1, \ldots, U_{N(h)})^T$, with the matrix of the system $A = (a(\phi_i, \phi_j))$ of size $N(h) \times N(h)$. Because the ϕ_i's have small support, $a(\phi_i, \phi_j) = 0$ for most pairs of i and j, so the matrix A is sparse (in the sense that most of its entries are equal to 0); this property is crucial from the point of efficient solution - in particular, fast iterative methods are available for sparse linear systems. Once $(P_h{}')$ has been solved for $U = (U_1, \ldots, U_{N(h)})^T$, the expansion $(\star\star)$ provides the required approximation to u. The matrix A is called the stiffness matrix.

2.2.2 The Self-Adjoint Elliptic Problem

In the special case, when the boundary value problem is self-adjoint, i.e.,

$$a_{i,j} = a_{j,i},\ i, j = 1, \ldots, n,\ x \in \bar{\Omega},$$

and

$$b_i \equiv 0,\ i = 1, \ldots, n,\ x \in \bar{\Omega},$$

the bilinear functional $a(\cdot, \cdot)$ is symmetric in the sense that

$$a(v, w) = a(w, v), \quad \forall v, w \in \mathscr{H}_0^1(\Omega).$$

Thus, consider

$$-\sum_{i,j=1}^{n}\frac{\partial}{\partial x_j}\left(a_{i,j}(x)\frac{\partial u}{\partial x_i}\right)+c(x)u=f(x),\ x\in\Omega u=0 \text{ on } \partial\Omega \tag{2.8}$$

with $a_{i,j}(x)$ satisfying the ellipticity condition (2.6); $a_{i,j}=a_{j,i}$, $c(x)\geq 0$, $x\in\bar{\Omega}$. If we define the quadratic functional $J:\mathscr{H}_0^1(\Omega)\to\mathbb{R}$ by

$$J(v)=\frac{1}{2}a(v,v)-l(v),\ v\in\mathscr{H}_0^1(\Omega).$$

then we can restate (2.8) as a minimization problem.

Lemma 12. Let $u\in\mathscr{H}_0^1(\Omega)$ be the (unique) weak solution to (P) with $\mathscr{V}=\mathscr{H}_0^1(\Omega)$ and suppose that $a(\cdot,\cdot)$ is a symmetric bilinear functional on $\mathscr{H}_0^1(\Omega)$; then u is the unique minimizer of $J(\cdot)$ over $\mathscr{H}_0^1(\Omega)$.

Lemma 13. Let $u\in\mathscr{H}_0^1(\Omega)$ minimizes $J(\cdot)$ over $\mathscr{H}_0^1(\Omega)$; then u is the (unique) solution of the problem (P) with $\mathscr{V}=\mathscr{H}_0^1(\Omega)$. The problem (P) is called the **Euler-Lagrange equation** for this minimization problem.

As we would see in (2.23), convexity is very important in our context. Indeed, it is easy to show that $J(\cdot)$ is convex, i.e.

$$J((1-\theta)v+\theta w)\leq(1-\theta)J(v)+\theta J(w),\ \forall\theta\in[0,1],\quad \forall v,w\in\mathscr{H}_0^1(\Omega),$$

which follows from the identity

$$(1-\theta)J(v)+\theta J(w)=J((1-\theta)v+\theta w)+\frac{1}{2}\theta(1-\theta)a(v-w,v-w)$$

and the fact that $a(v-w,v-w)\geq 0$. Also, note that if u minimises $J(\cdot)$, then, $J(\cdot)$ has a stationary point at u.

One can see that following problems are equivalent:

$$\text{find } u \in \mathscr{H}_0^1(\Omega) \text{ such that } a(u,v) = l(v)\ \forall v \in \mathscr{H}_0^1(\Omega), \tag{W}$$

$$\text{find } u \in \mathscr{H}_0^1(\Omega) \text{ such that } J(u) \leq J(v)\ \forall v \in \mathscr{H}_0^1(\Omega). \tag{M}$$

Given that $\mathscr{V}$ is a certain finite-dimensional subspace of $\mathscr{H}_0^1(\Omega)$ which consists of continuous piecewise polynomiyals of a fixed degree, then the finite element approximation of (W) and (M) would be as follows respectively:

$$\text{find } u_h \in \mathscr{V}_h \text{ such that } a(u_h,v_h) = l(v_h)\ \forall v_h \in \mathscr{V}_h, \tag{W_h}$$

$$\text{find } u_h \in \mathscr{V}_h \text{ such that } J(u_h) \leq J(v_h)\ \forall v_h \in \mathscr{V}_h. \tag{M_h}$$

2.3 Neural Network

Universal approximation theorem is our main tool for construction of neural networks which can be used to solve PDEs. It gives us information needed to ensure that our network would converge to the solution of our problem. It states that a feedforward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of $\mathbb{R}^n$, under mild assumptions on the activation function. George Cybenko in 1989 proved the theorem for *sigmoid* activation functions [55]. Later, in [56] it has been shown that the class of deep neural networks is a universal approximation if and only if the activation function is not polynomial. It had shown in 1991 by Kurt Hornik that it is not the specific choice of the activation function, but rather the multilayer feedforward architecture itself which gives neural networks the potential of being universal approximators [57]; the output units are always assumed to be linear.

feedforward networks with a single hidden layer are universal approximators. However, the width of such networks has to be exponentially large. In 2017 Lu et al. proved universal approximation theorem for width-bounded deep neural networks [58]. In particular, they showed that width $n+4$ networks with ReLU activation functions can approximate any *Lebesgue integrable function* on n-dimensional input space with respect to L^1 distance if network depth is allowed to grow which can be achieved with *Residual Networks* [59, 60]. They also showed the limited expressive power if the width is less than or equal to n.

A later result of their work showed that ReLU networks with width $n+1$ is sufficient to approximate any continuous function of n-dimensional input variables [61]. [62] is a good starting point to read and enthusiastic readers may find [63] quite interesting.

Theorem 2.23. ***(Universal Approximation Theorem)***.

i. ***(Unbounded Width case)*** *Let $\varphi : \mathbb{R} \to \mathbb{R}$ be a non-constant, bounded, and continuous function (called the activation function). Let I_m denote the m-dimensional unit hypercube $[0,1]^m$. The space of real-valued continuous functions on I_m is denoted by $C(I_m)$. Then, given any $\varepsilon > 0$ and any function $f \in C(I_m)$, there exist an integer N, real constants $v_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}^m$ for $i = 1, \ldots, N$, such that we may define:*

$$F(x) = \sum_{i=1}^{N} v_i \varphi\big(w_i^T x + b_i\big)$$

as an approximate realization of the function f; that is,

$$|F(x) - f(x)| < \varepsilon$$

for all $x \in I_m$. In other words, functions of the form $F(x)$ are dense in $C(I_m)$. This still holds when replacing I_m with any compact subset of $\mathbb{R}^m$.

ii. ***(Bounded Width Case)*** *The universal approximation theorem for width-bounded networks can be expressed mathematically as follows:*

For any Lebesgue-integrable function $f : \mathbb{R}^n \to \mathbb{R}$ *and any* $\epsilon > 0$*, there exists a fully-connected ReLU network* $\mathcal{A}$ *with width* $d_m \leq n + 4$*, such that the function* $F_{\mathcal{A}}$ *represented by this network satisfies*

$$\int_{\mathbb{R}^n} |f(x) - F_{\mathcal{A}}(x)| \mathrm{d}x < \epsilon.$$

Here is a list of some of important results and works from literature:

i. Multilayer feedforward networks are universal approximators [64].

ii. Neural network with unbounded activation functions is universal approximator [65].

iii. Optimal approximation of piecewise smooth functions using deep ReLU neural networks [66].

iv. On the approximation by neural networks with bounded number of neurons in hidden layers [67].

v. Resnet with one-neuron hidden layers is a universal approximator [60].

Another important tool for us which enables us to set an error bound for our network is residual connections [59].

Chapter 3

Mathematical Models

In this chapter, we will introduce two PDEs which we want to solve as an example. To maintain generality we choose one linear, inhomogeneous elliptic equation - effective conductance in a non-homogeneous media - and one nonlinear inhomogeneous Schrödinger equation with random background potential.

3.1 Effective Coefficients for Inhomogeneous Elliptic Equation

Let

$$\mathscr{A} = \{a \in L^\infty([0,1]^d) | \lambda_1 \geq a(x) \geq \lambda_0 > 0\}, \tag{3.1}$$

for some fixed constants λ_0 and λ_1. Then the effective conductance/coefficient in a non-homogeneous media can be expressed as a functional $A_{\text{eff}} : \mathscr{A} \to \mathbb{R}$ defined by

$$A_{\text{eff}}(a) = \min_{u(x)} \int_{[0,1]^d} a(x) ||\nabla u(x) + \xi||_2^2 \, \mathrm{dx}, \tag{3.2}$$

where ξ is a fix direction in $\mathbb{R}^d$ with $||\xi||_2 = 1$ ($||.||_2$ is the Euclidean norm). For $x \in [0,1]^d$ the minimizer $u_a(x)$ of the variational problem (3.2) would satisfy the following elliptic partial differential equation:

$$-\nabla \cdot (a(x)(\nabla u(x) + \xi)) = 0. \tag{3.3}$$

To confirm that (3.3) and (3.2) are equivalent we define $\tilde{J}(\lambda)$ as $\tilde{J}(\lambda) = J(u_a + \lambda\phi)$ where $J(u) = \int_{[0,1]^d} a(x)||\nabla u(x) + \xi||_2^2 \,\mathrm{dx}$ for $\lambda \in \mathbb{R}$ and $\phi \in C_0^\infty([0,1]^d)$. Then

$$\begin{aligned}\tilde{J}(\lambda) &= \int_{[0,1]^d} a(x)||\nabla u_a(x) + \lambda\nabla\phi + \xi||_2^2 \,\mathrm{dx} \\ &= \int_{[0,1]^d} a(x)[< \nabla u_a(x) + \xi, \nabla u_a(x) + \xi > +2\lambda < \nabla u_a(x) + \xi, \nabla\phi > +\lambda^2 < \nabla\phi, \nabla\phi >] \,\mathrm{dx}\,.\end{aligned} \tag{3.4}$$

The idea here is to use the derivative of $\tilde{J}(\lambda)$ to find the minimum of it. Since we define $\tilde{J}(\lambda)$ such that the minimum of it would be the same as A_{eff}. Lets calculate the derivative of $\tilde{J}(\lambda)$:

$$\tilde{J}'(\lambda) = \int_{[0,1]} a(x)[2 < \nabla u_a(x) + \xi, \nabla\phi > +2\lambda < \nabla\phi, \nabla\phi >] \,\mathrm{dx}\,. \tag{3.5}$$

Due to the definition of $\tilde{J}'(\lambda)$, one can see that $\tilde{J}'(0) = 0$, so:

$$\begin{aligned}\tilde{J}'(0) &= 2\int_{[0,1]} a(x)(\nabla u_a(x) + \xi) \cdot \nabla\phi \,\mathrm{dx} \\ &= 2[a(x)(\nabla u_a(x) + \xi) \cdot \phi - \int_{[0,1]} \nabla \cdot (a(x)(\nabla u_a(x) + \xi)) \cdot \phi \,\mathrm{dx}] = 0.\end{aligned} \tag{3.6}$$

Since ϕ is a compact support function on $[0,1]$ the $\phi = 0$ out of the boundary:

$$\tilde{J}'(0) = \int_{[0,1]} -\nabla \cdot (a(x)(\nabla u_a(x) + \xi)) \cdot \phi \,\mathrm{dx} = 0. \tag{3.7}$$

Furthermore, $\phi \in c_0^\infty$, which means for any $\psi \in L^2([0,1]^d)$ we can find a sequence like $\{\phi_i\}_{i=0}^\infty$ which converges to that ψ and these together yells that :
$-\nabla \cdot (a(x)(\nabla u_a(x) + \xi)) \overset{\text{a.e}}{=} 0$.

Notation 14. To simplify finite difference notations in higher dimensions, suppose:

i. $\{e_k\}_{k=1}^d$ is the the canonical basis in $\mathbb{R}^d$,

ii. $i \in (i_1, \ldots, i_d) | 1 \leq i_1, \ldots, i_d \leq n$.

Then $u_{i \pm e_k}$ denotes $u_{i_1,\ldots,i_{k\pm1},\ldots,i_d}$.

With this notation, to write the finite difference schema one begins by approximating the first and second derivatives:

$$\frac{\partial}{\partial x_k} a(x) = \frac{a_{i+\frac{1}{2}e_k} - a_{i-\frac{1}{2}e_k}}{h} \tag{3.8}$$

$$\frac{\partial}{\partial x_k} u(x) = \frac{u_{i+e_k} - u_i}{h} \tag{3.9}$$

$$\frac{\partial^2}{\partial x_k^2} u(x) = \frac{u_{i+e_k} - 2u_i + u_{i-e_k}}{h^2} \tag{3.10}$$

where $a_{i+\frac{1}{2}e_k} = \frac{a_{i+e_k}+a_i}{2}$ and $a_{i-\frac{1}{2}e_k} = \frac{a_{i-e_k}+a_i}{2}$. Now Lets rewrite (3.3) :

$$\begin{aligned} -\nabla \cdot (a(x)(\nabla u(x) + \xi)) &= -\nabla \cdot (a(x)\nabla u(x) + a(x)\xi) \\ &= -\nabla a(x)\nabla u(x) - a(x)\nabla^2 u(x) - \nabla a(x)\xi \\ &= -\sum_{k=1}^d \frac{\partial}{\partial x_k} a(x) \frac{\partial}{\partial x_k} u(x) - \sum_{k=1}^d a(x) \frac{\partial^2}{\partial x_k^2} u(x) - \sum_{k=1}^d \frac{\partial}{\partial x_k} a(x)\xi_k = 0. \end{aligned}$$

Next, we discretize the domain using a uniform grid with step size $h = \frac{1}{n}$ and grid points denoted by $x_i = i \times h$, where the multi-index $i \in \{(i_1, \cdots, i_d) | 1 \leq i_1, \cdots, i_d \leq n\}$ and then we can discretize the former equatoin:

$$
\begin{aligned}
&-\sum_{k=1}^{d} \frac{a_{i+\frac{1}{2}e_k} - a_{i-\frac{1}{2}e_k}}{h} \cdot \frac{u_{i+e_k} - u_i}{h} - \sum_{k=1}^{d} a_{i-\frac{1}{2}e_k} \frac{u_{i+e_k} - 2u_i + u_{i-e_k}}{h^2} - \sum_{k=1}^{d} \xi_k \frac{a_{i+\frac{1}{2}e_k} - a_{i-\frac{1}{2}e_k}}{h} \\
&= -\sum_{k=1}^{d} \frac{a_{i+\frac{1}{2}e_k} u_{i+e_k} - a_{i+\frac{1}{2}e_k} u_i - a_{i-\frac{1}{2}e_k} u_{i+e_k} + a_{i-\frac{1}{2}e_k} u_i + a_{i-\frac{1}{2}e_k} u_{i+e_k} - 2a_{i-\frac{1}{2}e_k} u_i + a_{i-\frac{1}{2}e_k} u_{i-e_k}}{h^2} \\
&\qquad - \sum_{k=1}^{d} \xi_k \frac{a_{i+\frac{1}{2}e_k} - a_{i-\frac{1}{2}e_k}}{h} \\
&\quad = -\sum_{k=1}^{d} \frac{a_{i+\frac{1}{2}e_k}[u_{i+e_k} - u_i] - a_{i-\frac{1}{2}e_k}[u_i - u_{i-e_k}]}{h^2} - \sum_{k=1}^{d} \xi_k \frac{a_{i+\frac{1}{2}e_k} - a_{i-\frac{1}{2}e_k}}{h} \\
&= \sum_{k=1}^{d} \frac{-a_{i+\frac{1}{2}e_k} u_{i+e_k} + [a_{i+\frac{1}{2}e_k} + a_{i-\frac{1}{2}e_k}] u_i - a_{i-\frac{1}{2}e_k} u_{i-e_k}}{h^2} - \sum_{k=1}^{d} \xi_k \frac{a_{i+\frac{1}{2}e_k} - a_{i-\frac{1}{2}e_k}}{h} = 0.
\end{aligned}
$$

Which can be written as $(L_a U)_i = (b_a)_i$ where

$$
(L_a u)_i := \sum_{k=1}^{d} \frac{-a_{i+\frac{1}{2}e_k} u_{i+e_k} + (a_{i-\frac{1}{2}e_k} + a_{i+\frac{1}{2}e_k}) u_i - a_{i-\frac{1}{2}e_k} u_{i-e_k}}{h^2} \tag{3.11}
$$

$$
(b_a)_i := \sum_{k=1}^{d} \frac{\xi_k (a_{i+\frac{1}{2}e_k} - a_{i-\frac{1}{2}e_k})}{h}. \tag{3.12}
$$

Next, to summarize later in matrix form, consider $d = 1$ (for $d > 1$ the process is the same, only the algorithm in computing coefficients matrix need more explanations which addressed in NLSE section by details). Begin by defining matrices and set the

boundary condition to Dirichlet boundary condition $u_{|_{\partial\Omega}} = 0$,

$$L_a = \frac{1}{h^2}\begin{bmatrix} a_{1-\frac{1}{2}e_1} + a_{1+\frac{1}{2}e_1} & -a_{1+\frac{1}{2}e_1} & & & & & \\ -a_{2-\frac{1}{2}e_1} & a_{2-\frac{1}{2}e_1} + a_{2+\frac{1}{2}e_1} & -a_{2+\frac{1}{2}e_1} & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & -a_{n-1-\frac{1}{2}e_1} & a_{n-1-\frac{1}{2}e_1} + a_{n-1+\frac{1}{2}e_1} & -a_{n-1+\frac{1}{2}e_1} & & \\ & & & -a_{n-\frac{1}{2}e_1} & a_{n-\frac{1}{2}e_1} + a_{n+\frac{1}{2}e_1} & & \end{bmatrix}_{n\times n},$$

$$b_a = \frac{1}{h}\begin{bmatrix} \xi_1(a_{1+\frac{1}{2}e_1} - a_{1-\frac{1}{2}e_1}) \\ \xi_1(a_{2+\frac{1}{2}e_1} - a_{2-\frac{1}{2}e_1}) \\ \vdots \\ \xi_1(a_{n+\frac{1}{2}e_1} - a_{n-\frac{1}{2}e_1}) \end{bmatrix}_{n\times 1}, U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}_{n\times 1}.$$

Then we can write $L_aU = b_a$. However, our boundary condition is not Dirichlet. To apply periodic boundary condition, one needs to patch some elements to L_a and U as follows:

$$L_a^{\text{patched}} = \frac{1}{h^2}\begin{bmatrix} a_{1-\frac{1}{2}e_1} + a_{1+\frac{1}{2}e_1} & -a_{1+\frac{1}{2}e_1} & & & & -a_{1-\frac{1}{2}e_1} \\ & & \ddots & \ddots & \ddots & \\ -a_{n+\frac{1}{2}e_1} & & & -a_{n-\frac{1}{2}e_1} & a_{n-\frac{1}{2}e_1} + a_{n+\frac{1}{2}e_1} & \end{bmatrix}_{n\times n},$$

$$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}_{n\times 1},$$

with $a_{1-\frac{1}{2}e_1} = \frac{a_n+a_1}{2}$ and $a_{n+\frac{1}{2}e_1} = \frac{a_1+a_n}{2}$. The iterative schema needed to generate the data set with sufficient stopping condition is as follow :

Algorithm 1 Calculate U_a

$U^0 \leftarrow 0$;

while (Stop condition is not satisfied) **do**

Solve $L_a^{\text{patched}} U = b_a$ for U with $U^0 \leftarrow U^i$;

$U^{i+1} \leftarrow U^i$;

end while

$U_a \leftarrow U^i$;

Plugging the u_a obtained from algorithm (1) to

$$A_{\text{eff}}(a) = h^d(u_a^T L_a^{\text{patched}} u_a - 2u_a^T b_a + a^T 1) \tag{3.13}$$

will give us $\mathscr{A}_{\text{eff}}$. To generalize this idea to higher dimensions, one can write L_a^{patched} and U for each dimension, then concatenating L_a^{patched} horizontally and U vertically gives required matrices to proceed with the rest of the algorithm. Note that in this case, for b_a we will compute the sum of b_a's in each dimension. In 1D, since we have the analytic solution for effective conductance as defined in (4.1), then we can write $\mathscr{A}_{ex} - \mathscr{A}_{app} < \epsilon$ where $\mathscr{A}_{app}$ is the approximation which can be obtained via substituting current U^{i+1} in (3.13).

3.1.1 Theoretical Justification of Deep Neural Network Representation

Tough everything we said till now together would show that we can solve PDEs with neural networks, here, our primary goal is not that. Instead, we will prove that we can represent a map from coefficient field to our quantities of interest via convolutional neural networks.

The main idea is to view the solution u of the PDE as being obtained via time evolution,

where each layer of the NN corresponds to the solution at discrete time step. In other words, mapping the input a from the first layer to last layer in the NN resembles the time-evolution of a PDE with discrete time-steps. We focus here on the case of solving elliptic equations with inhomogeneous coefficients. Similar line of reasoning can be used to demonstrate the representability of the ground state-energy E_0 as a function of a using an NN.

Notation 15. Let

$$\mathcal{E}(u;a) := \frac{h^d}{2}(u^T L_a u - 2u^T b_a + a^T 1). \tag{3.14}$$

One can see immediately that $\mathscr{A}_{\text{eff}} = 2\min_{u\in\mathbb{R}^{n^d}} \mathcal{E}(u;a)$.

Due to the variational characterization of (3.14), we can minimize $\mathcal{E}(u;a)$ over the solution space, using steepest decent:

$$u^{m+1} = u^m - \Delta t \frac{\partial \mathcal{E}(u;a)}{\partial u} = u^m - \Delta t(L_a u^m - b_a), \tag{3.15}$$

where Δt is a step size (called learning rate in machine learning context) chosen sufficiently small to ensure decent of the energy. The optimization problem is convex due to the ellipticity assumption of the coefficient field a in (3.1) with Lipshitz continuous gradient (which ensures $u^T L_a u > 0$ except for $u = 1$). Therefore the iterative schema converges to the minimizer with proper choice of the step size for any initial condition. Thus we can choose $u^0 = 0$.

At each step, we need to approximate the map from u^m to u^{m+1} in (3.15) using NN approximation. Hence, the process of time-evolution is similar to applying noisy gradient decent on $\mathcal{E}(u;a)$. More precisely, after performing a step of gradient decent update, the NN approximation incur noise to the update, i.e.

$$v^0 = u^0 = 0, \quad u^{m+1} = v^m - \Delta t \nabla \mathcal{E}(v^m;a), \quad v^{m+1} = u^{m+1} + \Delta t \gamma^{m+1}. \tag{3.16}$$

Here γ^{m+1} is the error for each layer of the NN in approximating each exact time-evolution iteration u^{m+1}. To be sure, instead of u^m, the object that is evolving in the NN as m changes is v^m.

Assumption 16. We assume $a \in \mathscr{A} = a \in \mathbb{R}^n | a_i \in [\lambda_0, \lambda_1], \forall i$ with $\lambda_0 > 0$. Under this assumption $\lambda_a := \|L_a\|_2$ and $\mu_a := \frac{1}{\|L_a^\dagger\|_2}$ satisfy

$$\lambda_a = \mathcal{O}(\lambda_1 h^{d-2}), \quad \mu_a = \Omega(\lambda_0 h^d). \tag{3.17}$$

Here for matrices $\|\cdot\|_2$ denotes the spectral norm, $h = 1/n$.

Assumption 17. We assume the NN results an approximation error term γ^{m+1} with properties

$$\left\|\gamma^{m+1}\right\|_2 \leq c\|\nabla \mathcal{E}(v^m; a)\|_2, \tag{3.18}$$

$$\mathbf{1}^T \gamma^{m+1} = 0, \tag{3.19}$$

for $m = 0, \ldots, M-1$ when approximating each step of time-evolution.

Lemma 18. The iteration in (3.16) satisfies

$$\mathcal{E}(v^{m+1}; a) - \mathcal{E}(v^m; a) \leq -\frac{\Delta t}{2}\|\nabla \mathcal{E}(v^m; a)\|_2^2, \tag{3.20}$$

id $\Delta t \leq \delta$, $\delta = (1 - \frac{1}{2(1-c)})\frac{2}{\lambda_a'}$ with $\lambda_a' = (1 + \frac{c^2}{1-c})\lambda_a$. Furthermore,

$$\frac{\Delta t}{2}\sum_{m=0}^{M-1}\left\|\mathcal{E}(v^{m+1}; a)\right\|_2^2 \leq \mathcal{E}(v^0; a) - \mathcal{E}(v^M; a) \leq \mathcal{E}(v^0; a) - \mathcal{E}(u^*). \tag{3.21}$$

Proof. From Lipshitz property of $\nabla\mathcal{E}(u;a)$,

$$\begin{aligned}
\mathcal{E}(v^{m+1};a)-\mathcal{E}(v^m;a) &\le \langle\nabla\mathcal{E}(v^m;a),v^{m+1}-v^m\rangle+\frac{\lambda_a}{2}\big\|v^{m+1}-v^m\big\|_2^2\\
&=\langle\nabla\mathcal{E}(v^m;a),v^m-\Delta t(\nabla\mathcal{E}(v^m;a)+\gamma^{m+1})-v^m\rangle\\
&+\frac{\lambda_a}{2}\big\|v^m-\Delta t(\nabla\mathcal{E}(v^m;a)+\gamma^{m+1})\big\|_2^2\\
&=-\Delta t(1-\frac{\Delta t\lambda_a}{2})\|\nabla\mathcal{E}(v^m;a)\|_2^2\\
&+\Delta t(1-\frac{\Delta t\lambda_a}{2})\left\langle\gamma^{m+1},\nabla\mathcal{E}(v^m;a)\right\rangle+\frac{\lambda_a\Delta t^2}{2}\|\gamma^m\|_2^2\\
&\le-\Delta t(1-\frac{\Delta t\lambda_a}{2})\|\nabla\mathcal{E}(v^m;a)\|_2^2\\
&+c\Delta t(1-\frac{\Delta t\lambda_a}{2}+\frac{c\Delta t\lambda_a}{2})\|\nabla\mathcal{E}(v^m;a)\|_2^2\\
&=-\Delta t\big((1-c)-(1-c+c^2)\frac{\Delta t\lambda_a}{2}\big)\|\nabla\mathcal{E}(v^m;a)\|_2^2\\
&=-\Delta t(1-c)\big(1-\frac{1-c+c^2}{1-c}\frac{\Delta t\lambda_a}{2}\big)\|\nabla\mathcal{E}(v^m;a)\|_2^2\\
&=-\Delta t(1-c)(1-\frac{\Delta t\lambda_a'}{2})\|\nabla\mathcal{E}(v^m;a)\|_2^2.
\end{aligned}$$

Letting $\Delta t\le\big(1-\frac{1}{2(1-c)}\big)\frac{2}{\lambda_a'}$, we get

$$\mathcal{E}(v^{m+1};a)-\mathcal{E}(v^m;a)\le-\frac{\Delta t}{2}\|\nabla\mathcal{E}(v^m;a)\|_2^2 \tag{3.22}$$

Summing the left hand side and the right hand side gives (3.21). this concludes the lemma. □

Theorem 3.1. *If Δt satisfies the the condition in Lemma 18, given any $\epsilon>0$, $|\mathcal{E}(v^M;a)-\mathcal{E}(v;a)|\le\epsilon$ for $M=\mathcal{O}((\frac{\lambda_1^2}{\lambda_0}+\lambda_1)\frac{n^2}{\epsilon})$.*

Proof. Since by convexity

$$\mathcal{E}(u^*;a)-\mathcal{E}(v^m;a)\ge\langle\nabla\mathcal{E}(v^m;a),u^*-v^m\rangle, \tag{3.23}$$

along with Lemma 18,

$$
\begin{aligned}
\mathcal{E}(u^*;a) &\le \mathcal{E}(u^*;a) + \langle \nabla\mathcal{E}(v^m;a), v^m - u^*\rangle - \frac{\Delta t}{2}\|\nabla\mathcal{E}(v^m;a)\|_2^2 \\
&= \mathcal{E}(u^*) + \frac{1}{2\Delta t}(2\Delta t\,\langle \nabla\mathcal{E}(v^m;a), v^m - u^*\rangle - \Delta t^2\|\nabla\mathcal{E}(v^m;a)\|_2^2 \\
&+ \|v^m - u^*\|_2^2 - \|v^m - u^*\|_2^2) \\
&= \mathcal{E}(u^*;a) + \frac{1}{2\Delta t}(\|v^m - u^*\|_2^2 - \|v^m - \Delta t\nabla\mathcal{E}(v^m;a) - u^*\|_2^2) \\
&= \mathcal{E}(u^*;a) + \frac{1}{2\Delta t}(\|v^m - u^*\|_2^2 - \left\|v^m - \Delta t\gamma^{m+1} - u^*\right\|_2^2) \\
&= \mathcal{E}(u^*;a) + \frac{1}{2\Delta t}(\|v^m - u^*\|_2^2 - \left\|v^{m+1} - u^*\right\|_2^2 \\
&+ 2\Delta t\,\langle \gamma^{m+1}, v^{m+1} - u^*\rangle - \Delta t^2\left\|\gamma^{m+1}\right\|_2^2) \\
&= \mathcal{E}(u^*;a) + \frac{1}{2\Delta t}(\|v^m - u^*\|_2^2 - \left\|v^{m+1} - u^*\right\|_2^2 + \Delta t^2\left\|\gamma^{m+1}\right\|_2^2 \\
&+ 2\Delta t\,\langle \gamma^{m+1}, v^m - u^*\rangle - 2\Delta t\,\langle \gamma^{m+1}, \nabla\mathcal{E}(v^m;a)\rangle) \\
&\le \mathcal{E}(u^*;a) + \frac{1}{2\Delta t}(\|v^m - u^*\|_2^2 - \left\|v^{m+1} - u^*\right\|_2^2 + \Delta t^2\left\|\gamma^{m+1}\right\|_2^2 \\
&+ 2\Delta t\left\|\gamma^{m+1}\right\|_2(\|v^m - u^*\|_2^2 + \|\nabla\mathcal{E}(v^m;a)\|_2)) \\
&\le \mathcal{E}(u^*;a) + \frac{1}{2\Delta t}(\|v^m - u^*\|_2^2 - \left\|v^{m+1} - u^*\right\|_2^2 + \Delta t^2\left\|\gamma^{m+1}\right\|_2^2 \\
&+ 2\Delta t(1 + \frac{2}{\mu_a})\left\|\gamma^{m+1}\right\|_2\|\nabla\mathcal{E}(v^m;a)\|_2) \\
&\le \mathcal{E}(u^*;a) + \frac{1}{2\Delta t}(\|v^m - u^*\|_2^2 - \left\|v^{m+1} - u^*\right\|_2^2 + c^2\Delta t^2\|\nabla\mathcal{E}(v^m;a)\|_2^2 \\
&+ 2c(1 + \frac{2}{\mu_a})\Delta t\|\nabla\mathcal{E}(v^m;a)\|_2^2).
\end{aligned}
$$

The last inequality follows from (3.17), which implies $\|L_a u\|_2 \ge \mu_a\|u\|_2$ if $u^T\mathbf{1} = 0$. More precisely, the fact that $v^0 = 0$, $\nabla\mathcal{E}(u)^T\mathbf{1} = 0$ (follows from the form of L_a and b_a defined in (3.11, 3.12)), and $(\gamma^m)^T\mathbf{1} = 0\ \forall m$(due to the assumption in (3.18)) implies $(v^m)^T\mathbf{1} = 0$, hence $\frac{\mu_a}{2}\|v^m - u^*\|_2 \le \|\triangle\mathcal{E}(v^m;a) - \triangle\mathcal{E}(u^*;a)\|_2 = \|\triangle\mathcal{E}(v^m;a)\|_2$.

Reorganizing our last inequality, we get

$$\begin{aligned}&\mathcal{E}(v^m;a)-\mathcal{E}(u^*;a)\\&\leq\frac{1}{2\Delta t}\left(\|v^m-u^*\|_2^2-\left\|v^{m+1}-u^*\right\|_2^2+c\Delta t(c\Delta t+2(1+\frac{2}{\mu_a}))\|\nabla\mathcal{E}(v^m;a)\|_2^2\right)\end{aligned}\tag{3.24}$$

Summing both left and right hand sides results in

$$\begin{aligned}\mathcal{E}(v^M;a)-\mathcal{E}(u^*;a)&\leq\frac{1}{M}\sum_{m=0}^{M-1}\mathcal{E}(v^{m+1};a)-\mathcal{E}(u^*;a)\\&\leq\frac{1}{M}\left[\left(\frac{\|v^0-u^*\|_2^2}{2\Delta t}\right)+\frac{2c}{\Delta t}\left(c\Delta t+2(1+\frac{2}{\mu_a})\right)(\mathcal{E}(v^0;a)-\mathcal{E}(u^*;a))\right]\end{aligned}\tag{3.25}$$

where the second inequality follows from (3.21). In order to derive a bound for $\|v^0-u^*\|_2^2$, we appeal to strong convexity property of $\mathcal{E}(u;a)$:

$$\mathcal{E}(v^0;a)-\mathcal{E}(u^*;a)\geq\langle\nabla\mathcal{E}(u^*;a),v^0-u^*\rangle+\frac{\mu_a}{2}\left\|v^0-u^*\right\|_2^2=\frac{\mu_a}{2}\left\|v^0-u^*\right\|_2^2$$

for $\langle\mathbf{1},v^0-u^*\rangle=0$. The last equality follows from the optimality of u^*. Then

$$\mathcal{E}(v^M;a)-\mathcal{E}(u^*;a)\leq\frac{1}{M}\left[\left(\frac{1}{\mu_a\Delta t}\right)+\frac{2c}{\Delta t}\left(c\Delta t+2(1+\frac{2}{\mu_a})\right)\right](\mathcal{E}(v^0;a)-\mathcal{E}(u^*;a)).\tag{3.26}$$

Since $\mathcal{E}(v^0;a)=h^d\frac{a^T\mathbf{1}}{2}=\mathcal{O}(\lambda_1)$, along with $\lambda_a=\mathcal{O}(\lambda_1h^(d-2))$ and $\mu_a=\Omega(\lambda_0h^d)$, we establish the claim. □

Theorem (3.1) will immediately results the following theorem:

Theorem 3.2. *Fix an error tolerance $\epsilon>0$, there exists a neural-network $h_\theta(\cdot)$ with $\mathcal{O}(n^d)$ hidden nodes per-layer and $\mathcal{O}((\frac{\lambda_1}{\lambda_0}+1)\frac{n^2}{\epsilon})$ layers such that for any $a\in\mathscr{A}=$*

$a \in \mathbb{R}^n | a_i \in [\lambda_0, \lambda_1], \forall i$, *we have*

$$|h_\theta(a)\mathscr{A}_{\mathit{eff}}(a)| \leq \epsilon\lambda_1.$$

Note that due to the ellipticity assumption $a \in \mathscr{A}$*, the effective conductivity is bounded from below by* $\mathscr{A}_{\mathit{eff}}(a) \geq \lambda_0 \geq 0$*. Therefore the theorem immediately implies a relative error bound*

$$\frac{|h_\theta(a) - \mathscr{A}_{\mathit{eff}}(a)|}{\mathscr{A}_{\mathit{eff}}(a)} \leq \epsilon\frac{\lambda_1}{\lambda_0}.$$

3.2 None Linear Schrödinger Equation with Inhomogeneous Background Potential

Consider following differential equation:

$$-\Delta u(x) + a(x)u(x) + \sigma u(x)^3 = E_0 u(x), x \in [0,1]^d, s.t. \int_{[0,1]^d} u(x)^2 dx = 1. \tag{3.27}$$

In this eigen value problem, given background potential $a(x)$, we are interested to find the smallest eigen value, *the ground state energy*, E_0. Here, $-\Delta u(x)$ represents the kinetic energy and $\sigma u(x)$ represents interaction between particles of the same kind. Let $\sigma = 2$ and thus, consider a defocusing cubic Schrödinger equation which can be understood as a model for solition in nonlinear photonics or Bose-Einstein condensate with inhomogeneous media. Similar to the previous section, we would like to solve the discretized version of equation (3.27) as follow

$$(Lu)_i + a_i u_i + \sigma u_i^3 = E_0 u_i, \sum_{i=1}^{n^d} u_i^2 h^d = 1 \tag{3.28}$$

where

$$(lu)_i := \sum_{k=1}^{d} \frac{-u_{i+e_k} + 2u_i - u_{i-e_k}}{h^2}.$$

To solve this equation, we use newton homotopy method as follow. Consider the sequence of NLSE, $\{(Lu)_i + a_i u_i + s u_i^3 = E_0 u_i\}_{i \in \mathbb{I}}$, with the normalization constraint on u, $h^d \langle u_i, u_i \rangle = 1$ with $s = s_1 \dots s_k$ where $0 < s_1 < s_2 < \cdots < s_k = \sigma$. Consider 2D case. When $s = 0$, the coefficient matrix can be calculated as follows:

Algorithm 2 Coefficient matrix

$i, j \leftarrow 1$;
Devide x_1 to p part and x_2 to q part;
for $r \in \{1, ..., p * q\}$ **do**
    if $mod(r, q) == 0$ **then**
        $j = q$;
    else
        $j = mod(r, q)$;
    end if
    $i = floor((r - 1)/p) + 1$;
    if $i > p$ **then**
        $i = mod(i, p)$;
    end if
    if $j > q$ **then**
        $j = mod(j, q)$;
    end if

16: **if** $i == p$ **then**

17: $A(r, j) = \frac{1}{h^2}$;

18: **else**

19: $A(r, ip + j) = \frac{1}{h^2}$;

20: **end if**

21: **if** $i == 1$ **then**

22: $A(r, (p-1)p + j) = \frac{1}{h^2}$;

23: **else**

24: $A(r, (i-2)p + j) = \frac{1}{h^2}$;

25: **end if**

26: **if** $j == q$ **then**

27: $A(r, (i-1)p + 1) = \frac{1}{h^2}$;

28: **else**

29: $A(r, (i-1)p + (j+1)) = \frac{1}{h^2}$;

30: **end if**

31: **if** $j == 1$ **then**

32: $A(r, (i-1)p + q) = \frac{1}{h^2}$;

33: **else**

34: $A(r, (i-1)p + (j-1)) = \frac{1}{h^2}$;

35: **end if**

36: $A(r, r) = -a_{i,j} - \frac{4}{h^2}$;

37: **end for**

From algorithm (2), one can find starting values E_0^0 and U^0. Note that, for $s = 0$, it is an standard eigenvalue problem; $AU = E_0 U$. Also,for $s_i > 0$, there is an additional update operation on the diagonal elements, that is, $A(r, r) = A(r, r) + s_i U_{i,j}^2$. Note that the normalization condition had been applied in the method implicitly. Hence, what

the algorithm computes is $E \times h^2$ so there remains another operation $E = E/h^2$. For each S_i, $i > 0$, inverse power method used to solve the NLSE and E_0, V_0 obtained with $s = s_i$ will be used to warm start for the next iteration. The procedure will proceed till $s_k = \sigma$. We used step size equals to 0.4 for that purpose.

In the rest of this thesis, we use algorithm (1) and equation (3.27) as our numerical schema for generating the Data set needed for training and testing the Neural-Network as will be discussed in details in Next Chapter.

Chapter 4

Results

In his chapter, we present the results for the neural schemes as derived in this thesis.

4.1 Neural Network Architecture

The structure of the network is shown in figure (4-2). The input to the NN is an n^d matrix representing the coefficient field $a \in \mathbb{R}^{n^d}$ on grid points, and the out put of the network will be $\mathscr{A}_{\text{eff}}$. The main part of the network are convolutional layers with ReLU activation function. This extracts the relevant features around each grid points. To incorporate the translational symmetry property of the solution, which is a previous knowledge, one can use a sum-pooling layer followed by a linear map. More precisely, let $d = 2$, $a_{i,j}^{\tau_1\tau_2} := a_{(i+\tau_1)(j+\tau_2)}$ where the additions are done on $\mathbb{Z}_n$. The output of the convolutional layer gives basis functions that satisfy

$$\tilde{\phi}_{k,i,j} = \tilde{\phi}_{k(i-\tau_1)(j-\tau_2)}(a), \qquad k = 1, \dots, \alpha, \quad i, j = 1, \dots, n, \quad \forall \tau_1, \tau_2 = 1, \dots, n.$$

When using the architecture in figure (4-1) for any τ_1, τ_2,

$$\begin{aligned} f(a^{\tau_1\tau_2}) &= \sum_{k=1}^{\alpha} \beta_k \sum_{i,j=1}^{n} \left(\tilde{\phi}_{kij}(a^{\tau_1\tau_2}) \right) \\ &= \sum_{k=1}^{\alpha} \beta_k \sum_{i,j=1}^{n} \left(\tilde{\phi}_{k(i-\tau_1)(j-\tau_2)}(a) \right) \\ &= \sum_{k=1}^{\alpha} \beta_k \phi_k(a), \\ \phi_k &:= \sum_{i,j=1}^{n} \tilde{\phi}_k ij, \end{aligned}$$

where β_k's are the weights of the last densely connected layer. The summation over i, j comes from the sum-pooling operation. Therefore, the later shows that the translational symmetry of f is preserved.

Note that, due to the periodic boundary condition, the input has to extend periodic too.

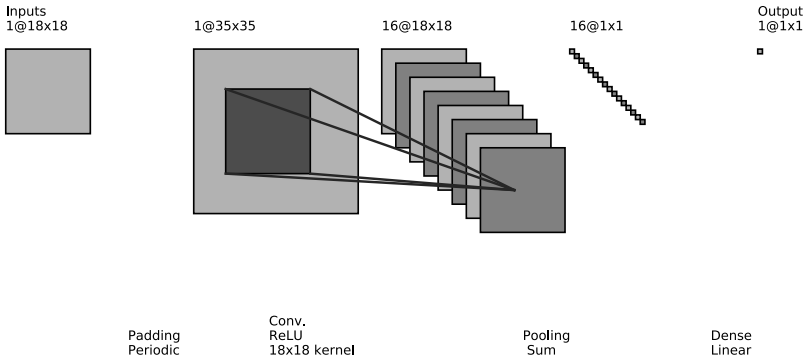


Figure 4-1: Single convolutional layer.

We solved 1D effective conductance. In 1D, the effective conductance can be expressed analytically as the harmonic mean of a_i's as follow:

$$A_{\text{eff}}(a) = \big(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{a_i}\big)^{-1}. \tag{4.1}$$

Full network schema can be found in figure (4-2). The schema can be divided into three parts. First part is consist of some convolutional layers with size 1 kernel window. Note that the last layer in this part has the channel size 1, since the output of that layer must be a vector of size n. The channel size for the rest of them is chosen to be 16. The plot of the output of this layer can be found in figure (4-3).

Second part is consist of a sum-pooling layer with size n window which produces an scalar.

Although the layers in third part are essentially densely-connected layers, but we taken them to be convolutional layers to reflect the symmetry between the first and third parts. Moreover, The input of this part is an scalar.

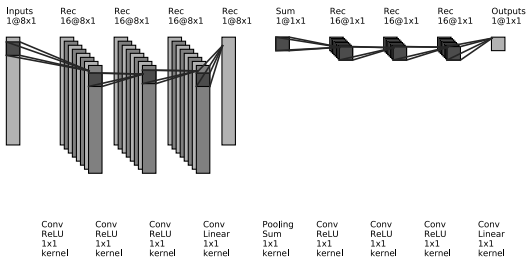


Figure 4-2: Neural network architecture for approximating A_{eff} in 1D.

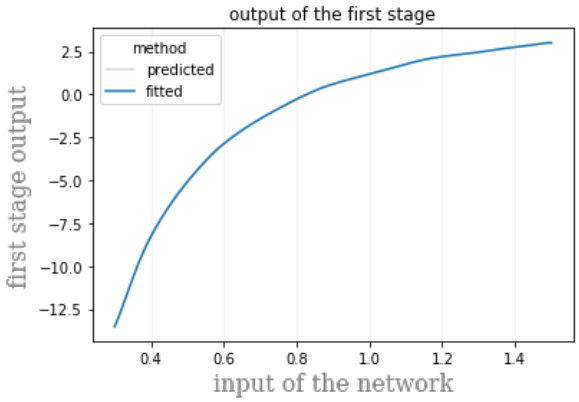


Figure 4-3: Output of the first stage of the neural network in approximating A_{eff} in 1D.

4.2 Effective Conductance in Inhomogeneus Media

We let $d = 1$. We let $a_i \sim \mathscr{U}[0.3, 1.5]$, giving an effective conductance of 0.77 ± 0.13 for $n = 8$. with $\alpha = 16$, we got 2.5793×10^{-5} for the minimum of the loss function (which was mean squared error), and 3.6×10^{-3} for our metric (which was the mean absolute error) on the training set. Moreover, minimum of the loss function on the test set was 5.20×10^{-6} and the metric reported to be 1.74210×10^{-3}. Finally, the predicted effective conductance by the network was 0.76800650 which is acceptable, $\|mean(A_{\text{eff-predicted}}) - mean(A_{\text{eff-exact}})\| = 1.02100 \times 10^{-3}$, and the sum of the error committed over all samples was $\|mean(A_{\text{eff-predicted}} - A_{\text{eff-exact}})\| = 3.4214979 \times 10^{-1}$. More details of the distribution of the error per sample can be found in Figures (4-4) and (4-5).

4.3 Nonlinear Shrödinger Equation

We let $d = 2$, the same network were used except that this time we used a convolutional 2D layer instead of convolutional 1D, and the pooling is also a 2D global pooling layer. 4786 sample were used to train, 861 samples were used to validate, and 1013 sample were

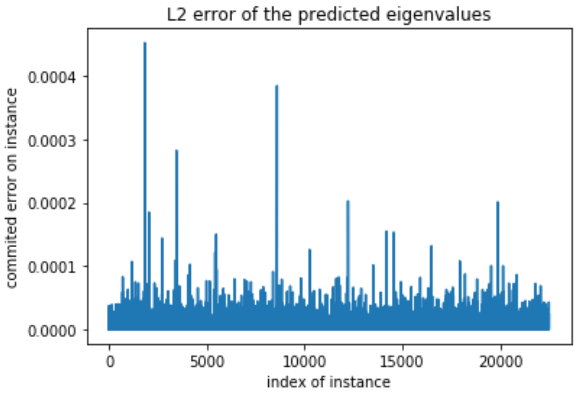


Figure 4-4: Committed error per sample over the prediction set

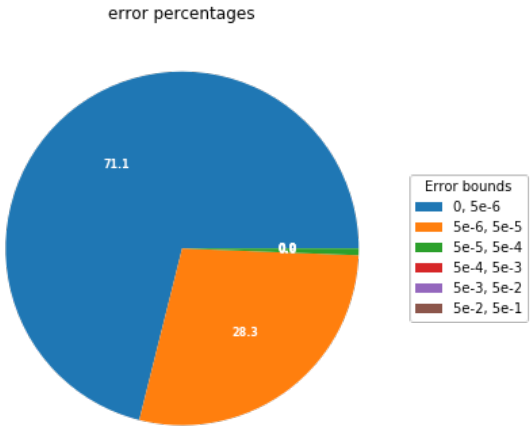


Figure 4-5: Committed error per sample over the prediction set distribution

used for predictions. Minimum of loss function on the train data, validation data, and prediction sets reported to be 0.0173, 0.0140, and 0.01425044 respectively. Moreover, the metric on on those sets reported to be 0.1048, 0.0934, 0.09431260. The mean of predicted ground state had been 10.17474556 which has the L^2-error 7.235×10^{-5} from the mean of the labels, and also is in agreement with the [33]. The number of parameters for our model was 1714, which is higher due to more layer. Figures (4-6) and (4-7) shows more details of committed errors over the prediction set.

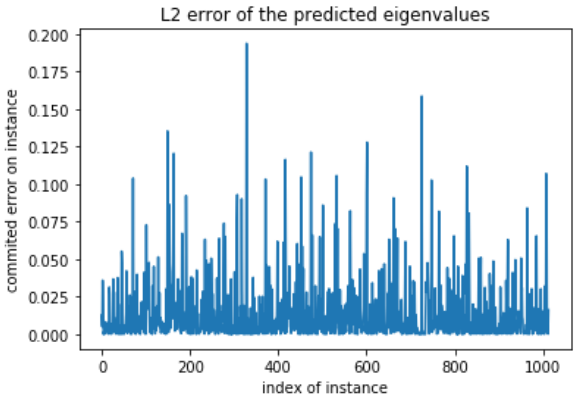


Figure 4-6: Committed error per sample over the prediction set

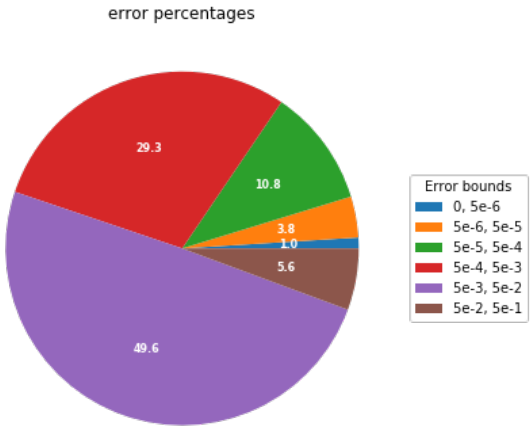


Figure 4-7: Committed error per sample over the prediction set

4.4 Conclusion

We pinpoint the bottle necks of the existed methods. We used neural networks to build a surrogate forward model to solve elliptic partial differential equations. Overall, neural networks seem to be promising for solving partial differential equations.

4.5 Future Work

One of the problems with neural networks is that they have problems with large labels. On the other hands, sometimes, we have large values as the solutions of the PDEs. For example, the ground state energy can be large. We used Z-score normalization. Will this potentially cause error? Is there any better way to deal with these large labels? If not, is there any error bound for this imposed error?

Bibliography

[1] Hamed Khatam, Gregory P Reece, Michelle C Fingeret, Mia K Markey, and Krishnaswamy Ravi-Chandar. In-vivo quantification of human breast deformation associated with the position change from supine to upright. *Medical engineering & physics*, 37(1):13–22, 2015.

[2] Christine Tanner, Andreas Degenhard, Julia Anne Schnabel, Andrew D Castellano-Smith, Carmel Hayes, Luke I Sonoda, Martin O Leach, D Rodney Hose, Derek LG Hill, and David John Hawkes. Comparison of biomechanical breast models: A case study. In *Medical Imaging 2002: Image Processing*, volume 4684, pages 1807–1818. International Society for Optics and Photonics, 2002.

[3] Lianghao Han, John H Hipwell, Christine Tanner, Zeike Taylor, Thomy Mertzanidou, Jorge Cardoso, Sebastien Ourselin, and David J Hawkes. Development of patient-specific biomechanical models for predicting large breast deformation. *Physics in Medicine & Biology*, 57(2):455, 2011.

[4] Fred S Azar, Dimitris N Metaxas, and Mitchell D Schnall. Methods for modeling and predicting mechanical deformations of the breast under external perturbations. *Medical image analysis*, 6(1):1–27, 2002.

[5] Fred S Azar, Dimitris N Metaxas, and Mitchell D Schnall. A deformable finite element model of the breast for predicting mechanical deformations under external perturbations. *Academic Radiology*, 8(10):965–975, 2001.

[6] A Pérez Del Palomar, B Calvo, J Herrero, J López, and M Doblaré. A finite element model to accurately predict real deformations of the breast. *Medical engineering & physics*, 30(9):1089–1097, 2008.

[7] Francisco Martínez-Martínez, María J Rupérez-Moreno, Marcelino Martínez-Sober, JA Solves-Llorens, Delia Lorente, AJ Serrano-López, Sandra Martínez-Sanchis, C Monserrat, and José David Martín-Guerrero. A finite element-based

machine learning approach for modeling the mechanical behavior of the breast tissues under compression in real-time. *Computers in biology and medicine*, 90:116–124, 2017.

[8] Nicholas Geneva and Nicholas Zabaras. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, 2020.

[9] Nejib Smaoui and Suad Al-Enezi. Modelling the dynamics of nonlinear partial differential equations using neural networks. *Journal of Computational and Applied Mathematics*, 170(1):27–58, 2004.

[10] Isaac E Lagaris, Aristidis C Likas, and Dimitris G Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–1049, 2000.

[11] Christian Beck, E Weinan, and Arnulf Jentzen. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *Journal of Nonlinear Science*, 29(4):1563–1619, 2019.

[12] Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

[13] Daniel R Parisi, Marıa C Mariani, and Miguel A Laborde. Solving differential equations with unsupervised neural networks. *Chemical Engineering and Processing: Process Intensification*, 42(8-9):715–721, 2003.

[14] Ioannis P Stavroulakis and Stepan A Tersian. *Partial differential equations: An introduction with Mathematica and MAPLE*. World Scientific Publishing Company, 1999.

[15] T. J. Sullivan. Introduction to uncertainty quantification. 2015.

[16] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[17] Donald O Hebb. *The organization of behavior*. na, 1949.

[18] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[19] Alekseui Grigorevich Ivakhnenko and Valentin Grigor'evich Lapa. *Cybernetic predicting devices*. CCM Information Corporation, 1973.

[20] Alekseĭ Ivakhnenko. Cybernetics and forecasting techniques.

[21] Stuart E Dreyfus. Artificial neural networks, back propagation, and the kelley-bryson gradient procedure. *Journal of guidance, control, and dynamics*, 13(5):926–928, 1990.

[22] Eiji Mizutani, Stuart E Dreyfus, and Kenichi Nishio. On derivation of mlp backpropagation from the kelley-bryson optimal-control gradient formula and its application. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 2, pages 167–172. IEEE, 2000.

[23] Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960.

[24] Arthur E Bryson. A gradient method for optimizing multi-stage allocation processes. In *Proc. Harvard Univ. Symposium on digital computers and their applications*, volume 72, 1961.

[25] Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki*, pages 6–7, 1970.

[26] Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, 16(2):146–160, 1976.

[27] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

[28] Quoc V Le. Building high-level features using large scale unsupervised learning. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8595–8598. IEEE, 2013.

[29] Bill Wilson. The machine learning dictionary: http://www.cse.unsw.edu.au/ billw/dictionaries/mldict.html. 1998 - 2012.

[30] E Weinan and Bing Yu. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[31] Min Wang, Siu Wun Cheung, Eric T Chung, Yalchin Efendiev, Wing Tat Leung, and Yating Wang. Prediction of discretization of gmsfem using deep learning. *Mathematics*, 7(5):412, 2019.

[32] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. arxiv. *arXiv preprint arXiv:1711.10561*, 2017.

[33] Yuehaw Khoo, Jianfeng Lu, and L. I. U. Ying. Solving pde problems with uncertainty using neural-networks. 2018.

[34] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, 19(1):932–955, 2018.

[35] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving for high-dimensional committor functions using artificial neural networks. *Research in the Mathematical Sciences*, 6(1):1, 2019.

[36] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[37] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. *arXiv preprint arXiv:1710.09668*, 2017.

[38] Keith Rudd and Silvia Ferrari. A constrained integration (cint) approach to solving partial differential equations using artificial neural networks. *Neurocomputing*, 155:277–285, 2015.

[39] Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[40] George E Forsythe and Wolfgang R Wasow. Finite difference methods. *Partial Differential*, 1960.

[41] Gordon D Smith, Gordon D Smith, and Gordon Dennis Smith Smith. *Numerical solution of partial differential equations: finite difference methods*. Oxford university press, 1985.

[42] Keith W Morton and David Francis Mayers. *Numerical solution of partial differential equations: an introduction*. Cambridge university press, 2005.

[43] Endre Süli. Lecture notes on finite element methods for partial differential equations. *Mathematical Institute, University of Oxford*, 2012.

[44] Claes Johnson. *Numerical solution of partial differential equations by the finite element method*. Courier Corporation, 2012.

[45] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, Perumal Nithiarasu, and JZ Zhu. *The finite element method*, volume 3. McGraw-hill London, 1977.

[46] Robert D Cook et al. *Concepts and applications of finite element analysis*. John wiley & sons, 2007.

[47] Wolfgang Dahmen, Andrew Kurdila, and Peter Oswald. *Multiscale wavelet methods for partial differential equations*. Elsevier, 1997.

[48] Ülo Lepik. Numerical solution of differential equations using haar wavelets. *Mathematics and computers in simulation*, 68(2):127–143, 2005.

[49] Fadl Moukalled, L Mangani, Marwan Darwish, et al. *The finite volume method in computational fluid dynamics*, volume 113. Springer, 2016.

[50] Randall J LeVeque et al. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002.

[51] Gui-Rong Liu and Moubin B Liu. *Smoothed particle hydrodynamics: a meshfree particle method*. World scientific, 2003.

[52] Gui-Rong Liu. *Meshfree methods: moving beyond the finite element method*. CRC press, 2009.

[53] Walter Rudin. Functional analysis. 1991. *Internat. Ser. Pure Appl. Math*, 1991.

[54] Craig C Douglas, Gundolf Haase, and Ulrich Langer. *A tutorial on elliptic PDE solvers and their parallelization*, volume 16. Siam, 2003.

[55] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[56] Bao-Ming Li, Zheng-Mei Mao, Min Wang, and Zhen-Tong Mei. Alpha-2 adrenergic modulation of prefrontal cortical neuronal activity related to spatial working memory in monkeys. *Neuropsychopharmacology*, 21(5):601–610, 1999.

[57] Josef Hofbauer and Karl Sigmund. Adaptive dynamics and evolutionary stability. *Applied Mathematics Letters*, 3(4):75–79, 1990.

[58] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pages 6231–6239, 2017.

[59] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[60] Hongzhou Lin and Stefanie Jegelka. Resnet with one-neuron hidden layers is a universal approximator. In *Advances in neural information processing systems*, pages 6169–6178, 2018.

[61] Boris Hanin and Mark Sellke. Approximating continuous functions by relu nets of minimal width. *arXiv preprint arXiv:1710.11278*, 2017.

[62] Domonkos Tikk, László T Kóczy, and Tamás D Gedeon. A survey on universal approximation and its limits in soft computing techniques. *International Journal of Approximate Reasoning*, 33(2):185–202, 2003.

[63] Anastasis Kratsios. Universal approximation theorems. *arXiv preprint arXiv:1910.03344*, 2019.

[64] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[65] Sho Sonoda and Noboru Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233–268, 2017.

[66] Philipp Petersen and Felix Voigtlaender. Optimal approximation of piecewise smooth functions using deep relu neural networks. *Neural Networks*, 108:296–330, 2018.

[67] Vugar E Ismailov. On the approximation by neural networks with bounded number of neurons in hidden layers. *Journal of Mathematical Analysis and Applications*, 417(2):963–969, 2014.

Chapter 5

Summary of the thesis in Persian

واژه‌نامه فارسی به انگلیسی

شبکه عصبی پیچشی ..... convolutional neural-network

یادگیری عمیق ..... deep learning

روش تفاضلات متناهی ..... finite difference method

روش عناصر متناهی ..... finite element method

محیط ناهمگن ..... inhomogeneous media

شبکه عصبی ..... neural-network

بعد فضائی ..... spatial dimension

عدم قطعیت ..... uncertainty

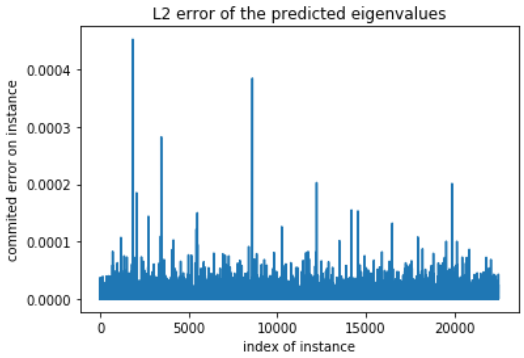
مقدار سنجی عدم قطعیت ..... uncertainty quantification

نتیجه‌گیری و کارهای پیش‌رو

همان‌گونه که از نتایج مشهود است، شبکه‌های عصبی توانایی بالایی در تقریب روابط پنان مابین داده‌ها دارند. همچنین سادگی روش، آن را به یک روش در دسترس تبدیل می‌کند. ضمن اینکه پس از طی مرحله آموزش، شبکه عصبی قادر است جواب مسئله را تقریبا به طور آنی ارائه کند. یکی از محدودیت‌های شبکه‌های عصبی در مورد اندازه مقیاس ورودی‌هاست: به این معنی که در صورتی که برچسب‌ها بسیار بزرگ باشند یا با فاصله بسیار از هم روند یادگیری با مشکل مواجه می‌شود. همان‌گونه که مشاهده می‌شود خطا در معادله شرودینگر به علت بزرگ بودن برچسب ها در مقایسه با ضرایب عدم قطعیت بیشتر است. ما از یک روش نرمال‌سازی برای نرمال‌سازی استفاده نمودیم. چه روش‌های دیگری برای حل این مسئله موجود است و آیا این نرمال سازی خود خطایی به مدل تحمیل می‌کند؟ کران این خطای تحمیلی چیست؟

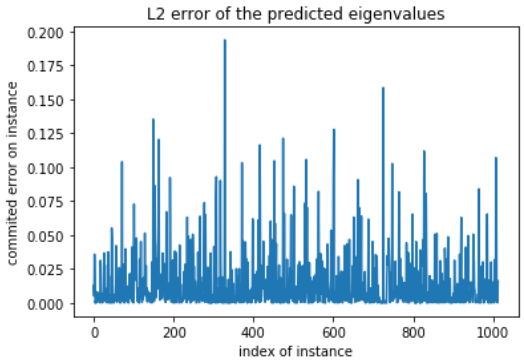
نتایج

مقادیر ضریب رسانایی مؤثر و انرژی حالت پایه به ترتیب ۰٫۷۶۸۰۰۶۵۰ و ۱۰٫۱۷۴۷۴۵۵۶ بدست آمده اند که خطای L^2 به ترتیب عبارت اند از $۱٫۰۲۱۰۰ \times ۱۰^{-۳}$ و $۷٫۲۳۵ \times ۱۰^{-۵}$. نمودار توزیع خطا بر حسب نمونه برای ضریب رسانایی مؤثر به شرح زیر است:



شکل ۱: خطای مرتکب شده روی مجموعه آزمون به تفکیک نمونه

همچنین، نمودار مشابه برای انرژی حالت پایه نیز به شرح است:



شکل ۲: خطای مرتکب شده روی مجموعه آزمون به تفکیک نمونه

روش پیشنهادی

معادله تعیین ضریب رسانش مؤثر تنها در یک بعد، و معادله شرودینگر در دو بعد حل خواهند شد. ابتدا معادلات فوق را با روش تفاضلات متناهی (و یا هر روش عددی دیگری) حل نموده و یک پایگاه داده می‌سازیم. برای اینکار، معادله تعیین ضرایب رسانش مؤثر روی یک شبکه نه نقطه‌ای متساوی‌الفاصله، با مقادیر ضرایب عدم قطعیت با توزیع نرمال $\mathscr{U}[۰/۳, ۱/۵]$، و معادله شرودینگر غیر خطی روی یک شبکه هشتاد و یک نقطه‌ای متساوی‌الفاصله (گسسته سازی نه نقطه‌ای هر کدام از ابعاد)با مقادیر ضرایب عدم قطعیت با توزیع نرمال $\mathscr{U}[۱, ۱۶]$ به تعداد نمونه‌های مورد نیاز حل می‌شوند. سپس درصدی از تکرارها (در اینجا هفتاد و پنج درصد) به عنوان داده برای مرحله آموزش و الباقی برای مرحله آزمون کنار گذاشته می‌شوند.

شبکه عصبی متشکل از سه بخش است. بخش اول و سوم قرینه یکدیگر و متشکل از لایه‌های پیچشی [۱] اند که به واسطه بخش دوم که یک استخر مجموع [۲] است به هم متصل شده‌اند. ورودی این شبکه برای رسانایی مؤثر و یک ماتریس برای معادله شرودینگر است. دقت شود که در حالت دو بعدی، قبل از لایه‌های پیچشی، ابعاد داده ورودی گسترش می‌یابد. این امر با توجه به اینکه شرط مرزی مسئله دوره‌ای است، به صورت کاشی کاری دوره‌ای انجام می‌شود. خروجی شبکه در هر دو حالت یک اسکالر است. که در مورد ضزیب رسانائی مؤثر، این اسکالر برابر ضریب رسانائی مؤثر در جهت ثابت ξ و در مورد معادله شرودینگر، برابر با سطح انرژی پایه است.

شبکه پس از چندین بار مرور داده ها در انتها ضرایب خود را به گونه ای تنظیم میکند که تابع هدفی که به آن معرفی کرده ایم را کمینه نماید. وقتی تابع مذکور به میزان کمینه خود برسد می‌گوییم آموزش شبکه به اتمام رسیده است. از این پس می‌توانیم با خوراندن ورودی جدید به شبکه از آن برای یافت جواب استفاده نماییم.

[1] convolutional layers
[2] sum-pooling

را به عنوان تقریبی از f ارائه دهیم که عبارت است از:

$$|F(x) - f(x)| < \varepsilon$$

که در آن $x \in I_m$ است. به عبارت دیگر، توابع به شکل $F(x)$ در $C(I_m)$ چگال‌اند.
این نتیجه به ازای هر زیر محموعه فشرده دیگری از $\mathbb{R}^m$ به جای I_m برقرار است.

۲. **(حالت کران‌دار)** در شبکه‌های کران‌دار، برای هرتابع انتگرال‌پذیر لبگ مانند $f : \mathbb{R}^n \to \mathbb{R}$ و هر $\epsilon > ۰$یک شبکه ReLu کامل $\mathcal{A}$ با عرض $d_m \leq n + ۴$، به گونه ای موجود است که $F_{\mathcal{A}}$ نمایش داده شده با این شبکه در رابطه

$$\int_{\mathbb{R}^n} |f(x) - F_{\mathcal{A}}(x)| \, \mathrm{d}x < \epsilon$$

صدق نماید.

شبکه‌های عصبی

شبکه عصبی از تعدادی واحد متصل به هم نام نورون تشکیل می‌شود. هر نورون دارای یک وضعیت داخلی است که در ترکیب با داده ورودی تغییر می‌کند و خروجی نورون را به حالت روشن یا خاموش تغیر می‌دهد. به عبارت ریاضی، هر نورون دارای ضرایب داخلی به نام وزن و بایاس است که به ترتیب با W و b نمایش داده می‌شوند. و خروجی نورون در این صورت با فرض اینکه X ورودی نورون باشد عبارت خواهد بود از $\phi(WX+b)$. در این صورت، می‌توان روند یادگیری یک شبکه عصبی را معادل با یک مسئله کمینه‌سازی در نظر گرفت. به شبکه رابطه‌ای روی داده های خروجی میدهیم تا آن را کمینه کند و ابزار شبکه برای کمینه سازی آن رابطه، تغییر وزن‌ها و بایاس‌های نورون‌های خود است. قضیه زیر که به قضیه تقریب جهانی مشهور است، این را بیان می‌کند که می‌توان تحت شرایطی از شبکه‌های عصبی برای تقریب جواب مسئله استفاده کرد:

قضیه. (قضیه تقریب جهانی)

۱. **(حالت نامتناهی)** فرض کنید $\varphi : \mathbb{R} \to \mathbb{R}$ یک تابع غیرثابت پیوسته بی کران باشد که آن را تابع فعال سازی می‌نامیم. فرض کنید I_m بیان‌گر ابر مکعب mـ بعدی $[۰,۱]^m$ باشد، و فضای توابع پیوسته حقیقی مقدار روی I_m با $C(I_m)$ نمایش داده شود. در این‌صورت، به ازای هر $\varepsilon > ۰$ دلخواه و هرتابع $f \in C(I_m)$، ثوابت حقیقی مانند $v_i, b_i \in \mathbb{R}$ و بردارهای $w_i \in \mathbb{R}^m$ برای $i = ۱, \ldots, N$وجود دارند، به طوری‌که می‌توانیم

$$F(x) = \sum_{i=۱}^{N} v_i \varphi\left(w_i^T x + b_i\right)$$

یافت ضریب رسانایی مؤثر در محیط ناهمگون

معادله دوم، معادله غیرخطی شرودینگر دو بعدی است. هدف از این معادله، یافت میزان انرژی حالت پایه الکترون با پتانسیل اولیه همراه با عدم قطعیت است. این معادله به صورت یک مسئله مقدار ویژه به صورت زیر تعریف می‌شود، که هدف ما در حل این مسئله یافتن کوچکترین مقدار ویژه آن است:

$$-\Delta u(x) + a(x)u(x) + \sigma u(x)^{۳} = E_{۰}u(x), x \in [۰, ۱]^{d}, s.t. \int_{[۰,۱]^{d}} u(x)^{۲}dx = ۱. \quad (۱)$$

تعریف مسئله

در این پایان نامه، هدف ما بررسی یک مدل میانبر برای حل مسائل معادلات دیفرانسیل با مشتقات جزئی به کمک شبکه‌های عصبی بوده است. به عبارت روشن‌تر یافت نگاشتی از فضای عدم قطعیت مسئله به فضای جواب. مسائلی که در این رساله برای حل انتخاب شده اند از این جهت حائز اهمیت بوده‌اند که هر دو حالت خطی و غیر خطی معادلات دیفرانسیل غیر همگن شامل عدم قطعیت را پوشش می‌دهند.

یافت ضریب رسانایی مؤثر در محیط ناهمگون

معادله اول، رسانایی مؤثر در یک جهت انتخاب شده درون یک ماده غیر همگون را توسط ضریب رسانش توصیف می‌کند. ماده غیر همگون، ماده‌ای است که در آن خصوصیات مورد توجه در تمامی نقاط یکسان نیستد. این امر ممکن است به دلایلی همچون جنس‌های گوناگون مواد تشکیل دهنده یا چگالی های متفاوت مربوط باشد. فرض ما بر آن است که ضریب رسانایی ماده در جهات متفاوت یکسان نباشد و این ضریب را با $a(x)$ نمایش می‌دهیم. با فرض انتخاب یک جهت دلخواه ثابت $\xi \in \mathbb{R}^d$، میزان ضریب رسانش در آن جهت مطلوب است. به عبارت دقیق‌تر، جواب معادله زیر مد نظر است:

$$A_{\text{eff}}(a) = \min_{u(x)} \int_{[۰,۱]^d} a(x)||\nabla u(x) + \xi||_۲^۲ \mathrm{d}x.$$

بایوپسی راهنمایی شده توسط MRI[1] بافت پستان توسط صفحه‌های سخت و غیرقابل انعطافی بی حرکت می‌شوند که منجر به فشرده شدن بافت نیز می‌شود. بنابراین، شکل، اندازه و مکان غده در این تصاویر متفاوت خواهد بود. این امر مقایسه تصاویر را با سختی بسیار همراه می‌کند. علاوه‌براین، برای برنامه‌ریزی پیش از جراحی، پزشک نیاز به دانستن مکان و اندازه دقیق غده دارد. بنابراین، نیاز به توسعه الگوریتم‌های ثبت غیرسخت[2] احساس می‌شود. روش‌هائی مبتنی بر روش المان‌های متناهی[3] برای حل این مسئله ارائه شده‌اند. اما مشکل عمده این روش‌ها هزینه محاسباتی بالای آنهاست. به طور متوسط اجرای یک شبیه سازی صد و بیست دقیقه به طول می‌انجامد که مقرون به صرفه نیست. ما به دنبال ارائه روشی برای کاهش این هزینه محاسباتی با استفاده از شبکه‌های عصبی و ارائه یک مدل مختص به بیمار در زمانی قابل قبول بودیم. این امر مستلزم در نظر گرفتن ضریب کشسانی بدن بیمار، که یک ضریب عدم قطعیت است، می‌باشد. از این رو، برآن شدیم که بدنبال حل عددی معادلات دیفرانسیل (بیضوی) به کمک شبکه‌های عصبی باشیم. در این رساله، بدنبال حل عددی معادلات دیفرانسیل با مشتقات جزئی بیضوی خطی و غیر خطی ناهمگن هستیم. روشی که ما در سدد معرفی آن هستیم، یک روش کاهش بعد برای محاسبه جواب بدون نیاز به حل مستقیم معادله دیفرانسیل است. ایده، استفاده از شبکه‌های عصبی برای یادگیری نگاشتی از دامنه ضرایب عدم قطعیت به فضای جواب بر اساس مجموعه داده‌ای از قبل محاسبه شده است.

[1] MRI-guided biopsy
[2] Non-rigid registration algorithm
[3] Finite element method

انگیزه و هدف

مدل‌سازی طبیعت همیشه با پارامترهایی همراه است که مقادیر آنها از کنترل ما خارج است. اما عموما ما درباره محدوده تغییرات این پارامترها اطلاعاتی داریم. به معادلاتی که شامل این‌گونه پارامترهایی هستند، معادلات با ضرایب عدم قطعیت گوییم. به طور مثال، در مورد حرکت مایعات، به طور مثال نفت، در سفره‌های زیر زمینی، برای بیان شیوه حرکت مایعات نیاز به دانستن مکان حفره‌ها داریم. این امر را می‌توان به صورت رسانایی مؤثر در حضور ناخالصی نیز در نظر گرفت. به عنوان مثالی دیگر، مسئله‌ای را مطرح می‌کنیم که نقطه شروع این رساله بوده است. برای تشخیص سرطان پستان روش‌های متعددی موجود است. از جمله آن‌ها می‌توان به تصویر برداری پستان با بازتابش اشعه ایکس (XRM)[1]، تصویربرداری با استفاده از ارتعاشات مغناطیسی (MRI)[2]، تصویربرداری فراصوت (US)[3]، توموسنتز دیجیتال (DBT)[4]، ماموگرافی انتشار پوزیترون (PET)[5] و توموگرافی فراصوت (UST)[6] اشاره کرد. هرکدام از این روش‌ها اطلاعات را به طرق مختلفی نمایش می‌دهند، به این معنا که غده‌ای که در یکی از این روش‌ها غیر قابل تشخیص است در روش دیگر قابل تشخیص است؛ غده‌ای که در یک روش بافت مشکوک معرفی می‌شود، در روش دیگر می‌تواند به عنوان غده‌ای سالم و طبیعی معرفی شود. و این موضوع باعث ایجاد مشکلات بسیاری در روند تشخیص و برنامه ریزی درمان می‌شود. در این مرحله، راه‌حلی که به ذهن می‌رسد، ترکیب نتایج حاصل از این روش‌ها برای بالابردن ضریب دقت است؛ لیکن مشکل دیگری مانع این‌کار می‌شود. بافت پستان بسیار کشسان است به راحتی تغییر فرم می‌دهد و هرکدام از این روش‌ها نیز به حالت خاصی از قرارگیری بیمار نیاز دارد. به طور مثال، طی MRI بیمار در حالت دمر قرار دارد ولی برای تصویربرداری فراصوت بیمار به پشت می‌خوابد. علاوه‌براین، در روش

[1] Projection X-ray mammography
[2] Magnetic resonance imaging
[3] Ultra sound
[4] Digital breast tomosynthesis
[5] Positron emission mammography
[6] Ultra sound tomography

مقدمه

مقدار سنجي عدم قطعيت (۱۵) در فيزيک و مهندسي اغلب شامل مطالعه معادلات ديفرانسيل با مشتقات جزئي با ميدان ضرايب تصادفي است. براي درک رفتار يک سيستم شامل عدم قطعيت، مي‌توان کميت‌هاي فيزيکي مشتق شده از معادلات ديفرانسيل توصيف کننده آن سيستم را به عنوان توابعي از ميدان ضرايب استخراج کرد. اما حتي با گسسته‌سازي مناسب روي دامنه معادله و برد متغييرهاي تصادفي، اين کار به طور ضمني به حل عددي معادله ديفرانسيل با مشتق جزئي به تعداد نمايي مي‌انجامد. يکي از روش‌هاي متداول براي مقدار سنجي عدم قطعيت روش نمونه برداري مونته کارلو است. گرچه اين روش در بسياري از موارد کاربردي است اما کميت اندازه‌گيري شده ذاتاً داراي پراش است. به‌علاوه اين روش قادر به پيدا کردن جواب‌هاي جديد در صورتي که قبلا نمونه‌گيري نشده باشند، نيست. روش گالرکين تصادفي با استفاده چند جمله‌اي‌هاي آشوب يک جواب تصادفي را روي فضاي متغييرهاي تصادفي بسط مي‌دهد و به اين طريق مسئله با بعد بالا را به تعدادي معادله ديفرانسيل با مشتقات جزئي معين تبديل مي‌کند. اين گونه روش‌ها به دقت زيادي درباره تعيين توزيع عدم قطعيت نيازمند هستند و از آن‌جا که پايه‌هاي استفاده شده مستقل از مسئله هستند، وقتي بعد متغييرهاي تصادفي بالا باشد هزينه محاسباتي بسيار زياد خواهد شد. هدف کار ما پارامتري کردن جواب يک معادله ديفرانسيل معيين به کمک شبکه‌هاي عصبي و سپس استفاده از روش‌هاي بهينه‌سازي براي يافتن جواب معادله است. در اين پايان‌نامه تابع مورد نظر براي پارامتري‌سازي روي ميدان ضرايب معادله ديفرانسيل با مشتقات جزئي تعريف شده است. در واقع ما به دنبال کاهش بعد مبتني بر نمايش شبکه عصبي براي حل معادلات ديفرانسيل با مشتقات جزئي همراه با عدم قطيت هستيم.

فهرست

چکیده

برای مدل‌سازی پدیده‌های واقعی با معادلات دیفرانسیل با مشتقات جزئی که شامل عدم قطعیت است، یکی از مشکلات وجود مجموعه‌ای از پدیده‌هاست که به عنوان مشکلات ابعاد بالا شناخته می‌شوند. خوشبختانه، اغلب تغییرات متغییرهای مدل میتوانند توسط تعداد کمی خصوصیات دامنه توسط روش‌های کاهش مدل، ثبت شوند. برای مثال، می‌توان با استفاده از روش‌های مبتنی بر شبکه‌های عصبی متغییرهای مورد نظر را به عنوان تابعی از ضرایب ورودی اندازه‌گیری کرد. در این‌صورت، نمایش پذیری متغیرها توسط چنین شبکه‌ای را می‌توان با دید شبکه عصبی به عنوان یک تحول زمانی برای پیداکردن جواب‌های مدل توجیه کرد. در این پایان نامه، ما یک روش میانبر برای پیدا کردن جواب‌های مدل روی دو معادله دیفرانسیل با مشتقات جزئی معروف در فیزیک و مهندسی را باربینی مینمائیم. همچنین، ما به سراغ بررسی یک روش عددی سنتی از نظر تئوری خواهیم رفت و از این طریق، احتمالات جدیدی برای استفاده از شبکه‌های عصبی در حل معادلات دیفرانسیل را مطرح خواهیم نمود.

واژه‌های کلیدی: *شبکه‌های عصبی، روش تفاصلات متناهی، روش المان‌های متناهی، معادلات دیفرانسیل با مشتقات جزئی، عدم قطعیت.*

وزارت علوم، تحقیقات و فناوری
دانشگاه تحصیلات تکمیلی علوم پایه
گاوازنگ - زنجان

حل عددي معادلات ديفرانسيل با مشتقات جزئی با ضرايب نامعين به کمک شبکه عصبی

پایان‌نامهٔ کارشناسی ارشد ریاضیات کاربردی ـ آنالیز عددی

ساجد زرین پور نشرودکلی

استاد راهنما: **دکتر خدیجه ندائی اصل**

استاد مشاور: **دکتر پروین رزاقی**

شهریور ۱۳۹۹