Physics Inspired Deep Learning Solution of PDEs with Application to Image Processing

By

Nadine YIZERE (nadine.yizere@aims.ac.rw)
African Institute for Mathematical Sciences (AIMS), Rwanda

Supervised by: Dr. rer. nat. habil. Abebe Geletu W. Selassie

German research chair, AIMS Rwanda

Co-supervised by: Dr Eunice Gandote

African Institute for Mathematical Sciences (AIMS), Rwanda

June 2023

AN ESSAY PRESENTED TO AIMS RWANDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF SCIENCE IN MATHEMATICAL SCIENCES

DECLARATION

This work was carried out at AIMS Rwanda in partial fulfillment of the requirements for a Master of Science Degree. I hereby declare that except where due acknowledgment is made, this work has never been presented entirely or in part for the award of a degree at AIMS Rwanda or any other University.

Student: Nadine YIZERE

[illegible]

Supervisor: Dr. rer. nat. habil. Abebe Geletu W. Selassie

Co-supervised by : Dr Eunice Gandote

[illegible]

ACKNOWLEDGEMENTS

I would like to start by expressing my deepest gratitude to God for his guidance and blessings throughout my academic journey. I am also immensely grateful to my supervisor, Dr. rer. nat. habil. Abebe Geletu W. Selassie, for his invaluable guidance, expertise, and continuous support. I am thankful to my co-supervisor, Dr. Eunice Sonagnon Edwige Gandote, for her insights and encouragement. Furthermore I would like to express my sincere appreciation to my parents for taking care of me and helping me to become who I am today. They have always loved me no matter what and made sacrifices for me. They have believed in what I can do and have supported me through everything. I am really thankful for everything they have done for me and for always being there for me. I am also thankful to my brothers and sisters for always loving, supporting, and being my friends. They have brought happiness and laughter into my life. I am grateful for their constant presence and for being there whenever I needed them. Their belief in my potential and their encouragement to strive for excellence have been instrumental in my journey, driving me to work hard and achieve my goals. Additionally, I extend my thanks to the entire AIMS community for providing an excellent learning and research environment and for supporting my academic activities.

DEDICATION

I dedicate this thesis to Almighty God and my beloved parents.

Abstract

Physics Inspired Deep Learning is among powerful approximation method for complex systems of partial differential equations. In this study, we use unsupervised learning to approximate the solution of the two-dimensional heat equation while considering physical laws that are presented in the equation. We have looked at different methods such as finite difference method, finite element method, and spectral collocation that are commonly used to solve heat equation. We compared the results from these methods to the ones we obtained using Physics Inspired Deep Learning. Additionally, we expanded our research to include image denoising using the supervised learning approach and the heat equation. We experimented with both isotropic and anisotropic diffusion methods to address noise reduction in images.The experimental results show that physics inspired deep learning solution of heat equation is more realistic compared to that of existing methods. Also anisotropic diffusion denoised image more accurately compared to that of isotropic diffusion.

Keywords: Physics inspired deep learning, image denoising, isotropic and anisotropic diffusion.

Contents

DECLARATION i

ACKNOWLEDGEMENT ii

DEDICATION iii

Abstract iv

1 Introduction 1

1.1 Motivation 1

1.2 Objectives 2

1.3 Thesis organization 2

2 State of the art 3

2.1 Artificial Neural Networks (ANNs) and Deep Learning 3

2.2 Physics Inspired Deep Learning (PIDL) 5

2.3 Other approximation methods for Partial differential equations 8

2.4 Partial Differential Equations and Image processing 9

3 Problem statement and Algorithms 11

3.1 Problem Statement and summary of contributions 11

3.2 Methodology and Algorithms 11

4 Implementation and Discussions 21

4.1 Implementations 21

4.2 Image denoising 24

4.3 Discussions 27

5 Summary and conclusions 30

5.1 Summary 30

5.2 Conclusion 30

List of figures

2.1 First mathematical model for biological neuron (Wu and Feng, 2018) 3
2.2 Feed forward network (Rathi, 2018) 5
2.3 General architecture of PIDL(Guo, Cao, Liu, and Gao, 2020) 7

3.1 Finite difference method (Mawuli, 2020) 13
3.2 Discretization of the domain (Zieli, 1992) 15

4.1 FDM and FEM. 21
4.2 Spectral collocation method. 22
4.3 Sample data points 23
4.4 Solution of diffusion equation with PIDL. 23
4.5 Variation of loss values 24
4.6 Gaussian distribution and Guassian noise (Lendave, 2021). 25
4.7 Adding noise to a clean image 25
4.8 Denoising image by traditional methods 26
4.9 Denoising image by anisotropic diffusion 26
4.10 Denoising image by PIDL 27
4.11 Denoising image by PIDL 27

1. Introduction

Partial Differential Equations (PDEs), are powerful mathematical tools with various applications in explaining physical laws, modeling engineering systems as well as performing digital image processing tasks (Ruthotto and Haber, 2020). Solving nonlinear PDEs analytically is a challenging task, unless using approximation methods to get their solutions.Traditional methods such as Finite Difference Method (FDM), Finite Element Method (FEM), and Spectral Collocation are dominant in solving both linear and nonlinear PDEs systems. However the efficiency in approximating nonlinear PDEs and overcoming the curse of dimensionality is still a challenge for these methods (Berg and Nyström, 2018).

Recent progress in machine learning have shown the potential of deep learning to solve complex problems including nonlinear PDEs systems. However, many deep learning algorithms do not take into account the physics information contained in a PDE, and large amount of data are required to train the model effectively.To make better use of deep learning models, researchers are now turning to Physics-Inspired Deep Learning (PIDL).This approach incorporates the laws of physics, its principles and constraints into deep learning models. PIDL algorithm uses collocation points to solve PDEs in contrast to traditional mesh-based methods.The use of collocation points enhances the accuracy of the predicted solution which contribute in getting efficient results compared to the other deep learning algorithms. In particular PIDL predict accurately the solutions of PDEs with fewer training samples compared to other deep learning algorithm (Raissi, Perdikaris, and Karniadakis, 2017). PIDL approximates PDEs problems by training a neural network to minimize a loss function.The loss function has two main terms namely, mean square error for the initial and boundary conditions and mean square error for the residue function of the PDE (Blechschmidt and Ernst, 2021). In this study we will apply physics inspired deep Leaning to solve both linear and nonlinear PDEs and how they are applied in denoising images.

1.1 Motivation

PDEs are used in image processing tasks such as denoising image, improving quality of an image, image segmentation and etc.The fundamental notion was that images consist of data on various spatial scales, hence one could not predetermine one scale.This challenge can be removed by mathematical models, particularly PDEs, (Bredies, Lorenz, et al., 2018) which was the reason behind utilizing PDEs in denoising image tasks. Given that images are displayed on discrete grids particularly digital images, discretization techniques should be used to get accurate estimates to the PDEs presenting the image (Mikula, 2002). However many traditional approaches like FDM, FEM and spectral collocation were used to approximate PDEs but the curse of dimensionality is still a challenge. Deep learning techniques, specifically PIDL methods have shown great promise in solving PDEs.This new algorithm incorporates laws of physics, its principles and constraints to guide the neural network architecture and improve the accuracy of the solutions. Nevertheless in physics there is sometimes less amount of training data which can also be a challenge for some deep learning algorithms. PIDL algorithms use less training data and improve the accuracy of the solutions which is the main motivation for this study. By employing PIDL, we will construct a data-driven approach to approximate PDEs and utilize the obtained solutions to denoise images. Our objective is to enhance the accuracy and efficiency of image denoising tasks by leveraging the field of deep learning and incorporating algorithms based on PDEs.

1.2 Objectives

Objectives of this thesis are:

- to study the existing methods for numerical solution of PDEs such as FDM, FEM, FVM and spectral collocation.
- to study the concept of PIDL and its role in approximating PDEs for denoising images.
- to solve heat equation using PIDL technique and explore how it is applied to image denoising task.
- to demonstrate how PIDL can be used to solve denoising image problems and compare it with non physics inspired convolutional neural network.

1.3 Thesis organization

The outline of this thesis is organized as follows. Section 1 provides the introduction, motivation and the purpose of the study. Section 2 discusses the state of the art. Section 3 contains the problem statement and algorithms. Section 4 presents the implementation and discussions. Finally, Section 5 is dedicated to the summary of the results and conclusions.

2. State of the art

Deep learning is one of the effective methods used to solve complex problems in applied mathematics. It refers to a neural network with multiple hidden layers.The objective of using PIDL is to add physical constraints identified in PDE as a regularization term to enhance neural network performance.This method is special since it trains models with less input (Blechschmidt and Ernst, 2021). PIDL is a technique that can be used to approximate PDEs that are often used for image denoising. While deep neural network algorithms like Convolutional Neural Networks (CNN) are commonly used for this task, PIDL provides an alternative method for PDE-based image denoising, as described in the work by (Chambolle, 1994). In the next sections we will discuss the background of artificial neural networks. We will also discuss PIDL and the existing approximation methods for solving PDEs. At the end we will talk about PDEs and image processing in particular image denoising task.

2.1 Artificial Neural Networks (ANNs) and Deep Learning

Research about neural networks dated back to 1940 but during that time significant findings were still limited. In 1943 American psychologist Warren Mcculloch and mathematician Walter Pitts have proposed a model known as M-P.This model aims to describe behaviors of neurons in the brain using mathematical equation as shown in equations (2.1.1) and (2.1.2).The figure 2.1 demonstrates the process of M-P model.

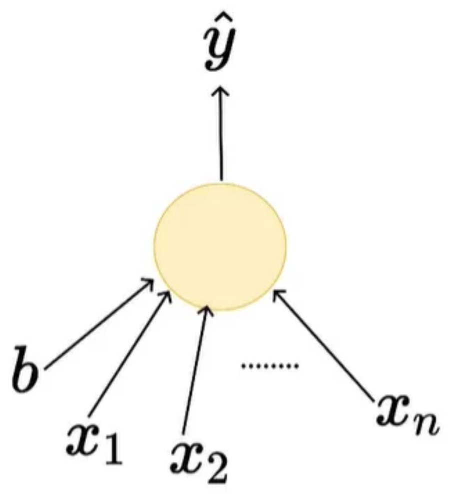


Figure 2.1: First mathematical model for biological neuron (Wu and Feng, 2018)

$$\hat{Y} = \sum_{i=1}^{n} w_i x_i \geq b \tag{2.1.1}$$

$$\mathcal{L} = \sum_{i} (Y_i - \hat{Y}_i)^2 \tag{2.1.2}$$

The constant b in (2.1.1) represents the bias. In the M-P model, since the output is binary, we need to avoid negative values, when the predicted value is 1 and the true value is 0. To overcome this, we use the square of the difference between the true and predicted. Numerous problems arise when the inputs are non-Boolean and functions are not linearly separable. Through research efforts, these limitations were gradually addressed (Chandra and First, 2018). ANNs are now defined as a series of models used in machine learning to imitate the appealing structural features of the nervous system and identify patterns in observations. The beauty of this strategy is that we do not tell the computer what to do instead, it makes observations based on the data set provided and provides the appropriate solutions. ANNs can be used to mine and fit complex relationships between data because it is made up of numerous connected artificial neurons. Additionally, the weights assigned to the connections between various neurons reflect the level of an effect that one neuron has over the others (Guo, Cao, Liu, and Gao, 2020). ANNs map an input X to an output Y. These models were frequently employed in supervised learning where the inputs and outputs are represented in the same datasets. The solution y is not known when modeling PDEs or Ordinary Differential Equations (ODEs). Hence we are unable to simply train a neural network to map the inputs x to the outputs we want. The equations can be rewritten and equate them to zero, hence are utilized as loss functions. Since the objectives of a neural network is to minimize its loss function, once it reduces the above loss function, the model will then learn how to solve the equation.

It sounds interesting to learn automatically from data but this is possible for few particular problems. However we did not know how to train neural networks to perform better than traditional techniques until 2006. The same year, new methods for learning, the so-called deep neural networks were discovered. Deep neural networks are ways of increasing number of hidden layers within the networks to improve performance. These now provides exceptional performance on a variety of significant problems in computer vision, speech recognition, and natural language processing (Nielsen, 2015). The most basic design of an ANN is the feed-forward structure, which consists of an input layer, an output layer, and one or more hidden layers, with information moving from lower to higher layers within the network. Figure 2.2 shows feed forward network.

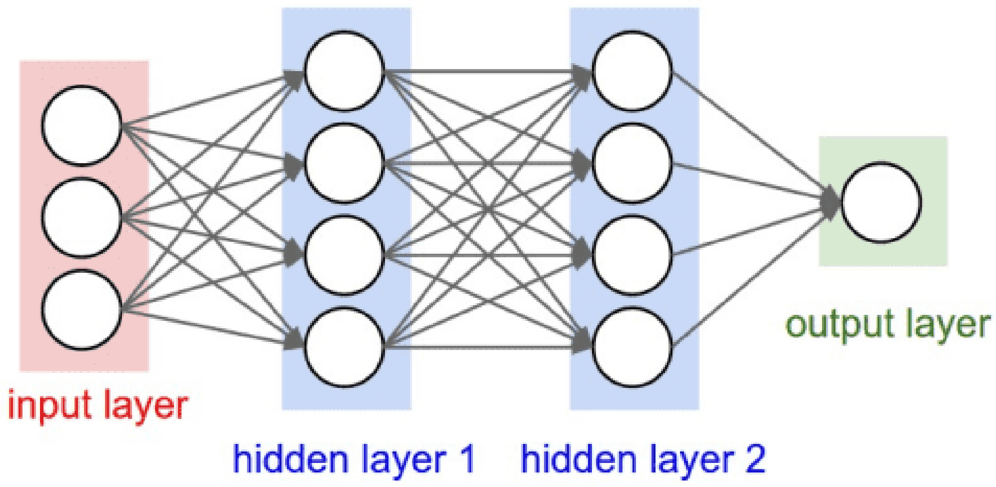


Figure 2.2: Feed forward network (Rathi, 2018)

Fundamental equations that govern this feed forward network, are as follow:

$$\begin{aligned} a_i &= \sigma(b_i + \sum_i^n w_{ij}x_i) \\ a &= \sigma(wx + b), \text{vector form} \\ u &= \sigma(Va + b') \end{aligned}$$

where x_i represents the inputs and in this study inputs are x, y, t. u represents the output, σ stands for activation function, a represents output of first hidden layer, w, V, b and b' represent weights and biases respectively. Feed forward neural network is not enough to get better efficient result when training deep learning model. Methods like back propagation, data augmentation and variable learning rate can be used to increase the efficiency of the model. In addition, efficiency of deep learning neural networks is enhanced by the so-called physical neural networks (PNN) according to (Wright, Onodera, Stein, Wang, Schachter, Hu, and McMahon, 2022). A significant challenge for deep learning system comes when predicting complicated physical processes. Pure data-driven deep learning models have limited generalizability, large training costs, and error accumulation. A more promising approach is to apply prior knowledge of physics to scientific deep learning models, or simply to use PIDL (Liu, Sun, and Wang, 2022).

2.2 Physics Inspired Deep Learning (PIDL)

PIDL is one of machine learning methods that aims to combine the principles of physics with deep learning techniques. PIDL approaches have been applied to a wide range of applied mathematics problems like inverse and forward problems using unsupervised and supervised learning.The essential idea behind PIDL is to use physical laws as constraints on the neural network architecture and training process. For example, the heat equation, which governs the diffusion of heat through a material, can be used to impose constraints on the neural network architecture.These constraints can help ensure that the neural network solutions satisfy the underlying physical laws and can improve the accuracy and robustness of the model. When generating data for the training model, it is better to use monte-carlo methods because they are computationally efficiency in particular for PDEs which contains second derivatives (Sirignano and Spiliopoulos, 2018). PIDL is applied by using the neural network and directly approximate the solution of a PDE (Raissi, Perdikaris, and Karniadakis, 2017; Sirignano and Spiliopoulos, 2018).This is the supervised learning approach where the neural network receives as input the coordinates of a point in space and time, hence the outputs the solution value correspond to these coordinates. In order to reduce the difference between the predicted and actual solutions, the network is trained on a set of known PDE solutions.The neural network learns to generate solutions that are consistent with the underlying physics by including physical laws as constraints.

Another approach to implementing PIDL is to use the neural network to directly learn the fundamental physical principles, or simply unsupervised learning. In this method, the coordinates of a location in space and time are fed into a neural network, which subsequently produces the related time derivative.The network can be trained on a set of observable states and their time derivatives to discover the fundamental physical laws that govern the system. This technique has been successfully used to forecast the dynamics of simple physical systems, like a pendulum or a mass-spring system, from noisy observations.

PIDL is preferable over traditional numerical methods for solving PDEs in several ways. First, especially for high-dimensional systems, PIDL can be orders of magnitude faster than traditional methods. Second, PIDL can offer more accurate solutions than traditional methods, particularly when the underlying physics is complex or unclear (Maziarz, Wang, Bao, and Zhang, 2021).The neural network may identify the physical variables that are most crucial for a certain situation, making the solutions of PIDL to be more interpretable.

PIDL has numerous advantages, yet it also has some drawbacks. One difficulty is the requirement for an enormous amount of high-quality training data, particularly for complex systems (Zhu and Zabaras, 2019).The problem of incorporating uncertainty and error estimates into PIDL models (Raissi, 2019) presents another difficulty. Last but not least, PIDL models are likely to be affected by over-fitting problem, particularly if the training data is sparse or the neural network architecture is poor.

PIDL is used to develop more effective and powerful models for dealing with complex problems such as nonlinear PDEs. In order to solve problems with limited data availability, such as imprecise measurements obtained from experiments, physics-informed neural networks (PINNs) were developed. PIDL can be grouped into two categories such as data-driven techniques and physics-based methods. Without explicitly outlining the underlying physics, data-driven methods employ machine learning techniques like deep neural networks to learn patterns from data. By creating a loss function that enforces these rules, physics-based approaches, on the other hand, include physical principles in the learning process. Combining both strategies and employing physics-based regularization to direct the learning of data-driven models is the most frequently employed approach in PIDL. To account for the lack of data, the algorithm is augmented with physical laws that govern the problem at hand.Typically, these laws are expressed through non-linear partial differential equations with parameters λ

$$\frac{\partial u}{\partial t} + \mathcal{N}[x : \lambda] = 0 \tag{2.2.1}$$

for $x, y \in \Omega$ and $t \in T$.

The solution $u(x, t)$ depends on time $t \in [0, T]$ and spartial variables x. $\mathcal{N}[x : \lambda]$ represent a non-linear differential operator (Kollmannsberger, D'Angella, Jokeit, Herrmann, Kollmannsberger, D'Angella, Jokeit, and Herrmann, 2021). Figure 2.3 shows a sample architecture of PIDL approach.

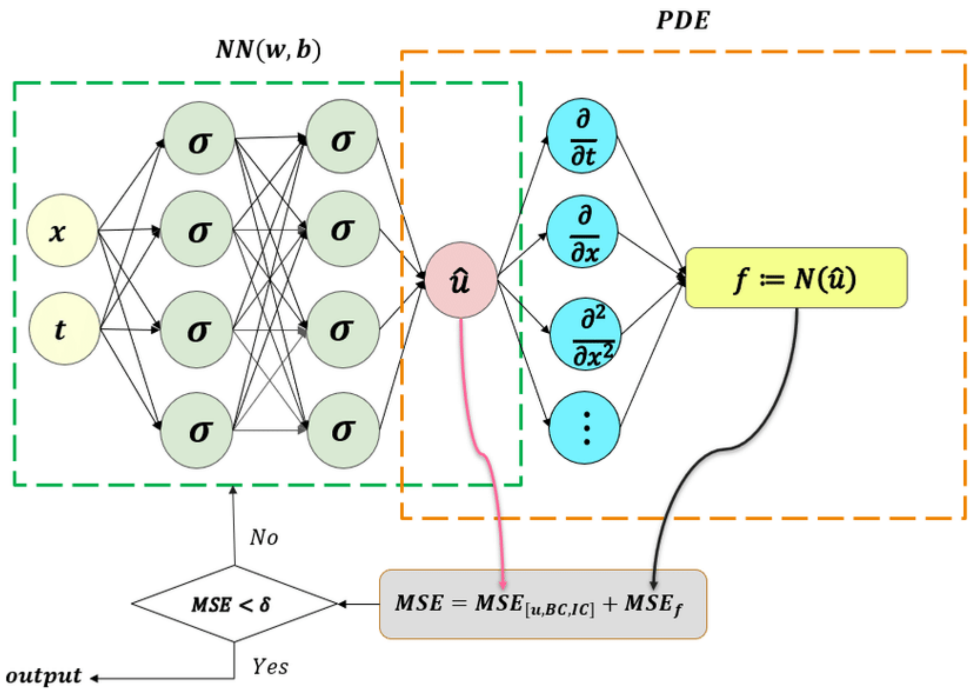


Figure 2.3: General architecture of PIDL(Guo, Cao, Liu, and Gao, 2020)

Traditional neural networks are completely dependent on a data-driven methodology that ignores the physical principles that are present in the data for example partial differential equation. This method requires a large amount of training data to train the neural networks and produce an accurate model. However PIDL only incorporate physical information into the network to ensure that the network output satisfy the associated partial differential equations. Specifically, the model is built to take physical laws into account during training by adding regularization about partial differential equations to the loss function.This process reduces the amount of data needed for training and its speed (Guo, Cao, Liu, and Gao, 2020). PIDL is suitable for approximating PDEs over domain Ω as shown in equation (3.2.1) as well as obtaining parameters from training data which is known as inverse problem.The loss function minimized by network of PIDL is a sum of two terms namely loss function corresponding to initial and boundary conditions.The second term requires the neural network to satisfy constraints of the partial differential equation, which connects the physical information components of the neural network.

$$L(\theta) = MSE_{u,Bc,Ic} + MSE_f \tag{2.2.2}$$

From equation (2.2.2), each term is defined as

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} \|\hat{u}^i - u(x_u^i, t_u^i)\|^2 \tag{2.2.3}$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} \|f(x_f^i, t_f^i)\|^2 \tag{2.2.4}$$

where $u(x_u^i, t_u^i)$ represents the training data from initial and boundary conditions while $u(x_f^i, t_f^i)$ represents the training data in the space-time domain. In other words they are randomly distributed internal domain collocation points (Raissi, Perdikaris, and Karniadakis, 2017).The advantage of PIDL over classical numerical approximations is that they are sufficiently adaptable to be applied to a wide range of complicated PDEs. In contrast traditional numerical approximation needs to be customized to a particular PDEs. According to (Guo et al., 2020), PIDL is among branches of deep learning that were discovered to overcome the challenges like insufficiency large amount of data. Particularly PIDL has come up to optimize the system using few data and achieve accurate results and lack of interpretability.The basic theory behind PIDL training is that it can be an unsupervised technique that does not require data with labels or supervised learning depending to the nature of the problem. Moreover this is a mesh-free approach of approximating complex systems of nonlinear PDEs. It transforms directly governing equations of the system into a loss function optimization problem.The PIDL algorithm is based on the mathematical descriptions of the problems that are stored in the loss or neural network.The accuracy of the model's predictions is impacted by the training data, specifically the data used to train the model, which are referred to as training points. (Cuomo, Di Cola, Giampaolo, Rozza, Raissi, and Piccialli, 2022).The problem of denoising images has been a classical challenge for researchers for several decades. Previously, filters were utilized to diminish the noise in images, which worked well for images with a moderate amount of noise. However, the downside was that applying filters could also introduce blurriness into the image. Additionally, when the noise level was high, the resultant image could become excessively blurry, causing crucial details in the image to be lost.The need for a more effective solution to this problem lead to creation of multiple deep learning models including PIDL that significantly outperform the conventional denoising filters (Karniadakis, Kevrekidis, Lu, Perdikaris, Wang, and Yang, 2021).

2.3 Other approximation methods for Partial differential equations

Solving nonlinear PDEs analytically is still a challenge in the field of applied mathematics (Beck, Hutzenthaler, Jentzen, and Kuckuck, 2020).Traditional approximation methods such as spectral Galerkin methods, FDM, FEM, sparse grid approximation methods, and standard nested Monte Carlo method succeed when approximating nonlinear PDEs. However they are not able to overcome the curse of dimensionality.The number of computational operations of the employed approximation scheme grows exponentially in the PDEs of dimension $d \in N$.

More research have been explored to find out which approximation method can overcome the curse of dimensionality in solving PDEs. In 1990, the first proposal of deep learning-based approximation methods for PDEs have come in case of low-dimensional PDEs (Beck, Hutzenthaler, Jentzen, and Kuckuck, 2020). Since 1990, many deep learning algorithms were developed such as, physics informed neural networks, spectral PIDL, deep Galerkin method (Beck, Hutzenthaler, Jentzen, and Kuckuck, 2020), etc.The primary motivation behind developing these algorithms is to design machine learning techniques that are more easily understandable. Additionally, these methods should be capable of withstanding imperfect data, such as missing or inaccurate values and anomalies, while providing precise and physically coherent predictions, even for generalization tasks. (Karniadakis, Kevrekidis, Lu, Perdikaris, Wang, and Yang, 2021).The primary objective of this study is to investigate the use of PIDL, and comparison with other algorithms that are used to solve PDEs. Furthermore, the approximated solution obtained from this approach will be utilized in denoising image.

denoising image is the process of modifying a digital image, especially by using a computer to eliminate noise and other anomalies (Mikula, 2002).This task is mainly performed by most of machine learning methods. However the existing methods like CNNs and recurrent neural networks require plenty of data to train the model. It has been discovered that PDEs can interpret well image processing tasks such as image segmentation, denoising, registration, and reconstruction, when we use efficient approximating methods (Ruthotto and Haber, 2020). Solving PDEs, one needs to apply novel algorithms and perspectives to perform denoising image by leveraging PIDL as a competitive approximation method. PIDL utilizes a smaller quantity of training data compared to convolutional neural networks or recurrent neural networks. Our investigation involved examining both linear and non-linear PDEs with low dimension using the methods mentioned earlier. Nonetheless, addressing high-dimensional PDEs numerically presents a challenge in improving PDEs-based algorithms for digital image processing (Blechschmidt and Ernst, 2021). To address this issue, we can leverage the Feynman-Kac formula to solve high-dimensional PDEs, which is analogous to approximating Backward Stochastic Differential Equations (BSDE) using Deep Learning approaches like PIDL.

2.4 Partial Differential Equations and Image processing

Image is defined as a two dimensional function say $f(x, y)$ with x and y being plane coordinates. Image content can be gray-values that ranging from 0 (black) to 255 (white) or color values that are vectors known as (r,g,b) (Bredies, Lorenz, et al., 2018). It can also be defined by discrete domain

$$\Omega = \{1, \cdots, N\}\times, \cdots, \times\{1, \cdots, N_d\} \tag{2.4.1}$$

or by continuous domain $\Omega \in \mathbb{R}^d$

$$\Omega = [0, a_1]\times, \cdots, \times[0, a_d] \tag{2.4.2}$$

In simple words, a digital image is a representation of an image using a discrete domain, as opposed to a continuous image which uses a continuous domain (Bredies, Lorenz, et al., 2018).The fundamental notion is that images consist of data on various spatial scales, hence one could not predetermine one scale.That challenge can be removed by mathematical models, particularly PDEs. In the area of image processing based on PDEs, the heat or diffusion equation is important. equation (Bredies, Lorenz, et al., 2018). Linear diffusion equation, has been used for a various image-processing operations, including image denoising and smoothing. When performing the same task of smoothing and denoising images, the results of the linear diffusion equation are equivalent to those of the Gaussian kernel. However, one major drawback of the linear diffusion method is that it uniformly eliminates both local signal features and noise. Perona and Malik have address this limitation by using nonlinear diffusion process.The gradient of the signal was chosen to be a decreasing function of the P-M diffusion coefficient, which helps to control the image smoothing and denoising tasks (Gilboa, Sochen, and Zeevi, 2002). PDEs have been widely used for image denoising due to their ability to smooth the image while preserving its edges. However, traditional PDE-based denoising methods suffer from limitations such as the difficulty of parameter selection and the loss of fine details in the image. In recent years, deep learning methods such as PIDL, have emerged as a promising alternative for image denoising. It uses the laws of physics as constraints to train deep neural networks for various tasks, including image denoising.The motivation behind PIDL is that the laws of physics represent powerful prior knowledge that can help to overcome some of the limitations of traditional deep learning methods. PIDL method for image denoising is unique from the fact that the network it uses less training data and provide accurate results unlike traditional learning methods which requires large amount of data for training. With PIDL we have the so-called diffusion-based-network, which incorporate diffusion equation to a convolutional neural network (CNN) to learn the diffusion process from the data and applies it to the noisy image to denoise it.The results show that DiffusionNet outperforms traditional PDE-based denoising methods in terms of both quantitative metrics and visual quality (Salamat, Missen, and Surya Prasath, 2021).

It is possible to incorporate Perona-Malik diffusion with PIDL method when performing image denoising task so as to get accurate results (Wen, Liu, and Xiao, 2020). Perona-Malik equation, is a PDE that is widely used for image denoising. We use a CNN to learn the diffusion process and applies it to the noisy image to denoise it.The results show how Perona-Malik diffusion performs better in terms of measurement accuracy and visual quality than other deep learning techniques and traditional PDE-based denoising techniques.

Recently (Zhang, Zuo, Chen, Meng, and Zhang, 2017) proposed a PIDL method for image denoising called the Navier-Stokes Net (NS-Net). It is based on the Navier-Stokes equations, which describe the motion of fluids in a medium. NS-Net uses a CNN to learn the flow field from the data and applies it to the noisy image to denoise it.The findings indicate that NS-Net performs better than conventional denoising methods based on PDE and other deep learning techniques, considering both quantitative measures and visual appearance. PIDL is a promising approach for image denoising. By incorporating the laws of physics as constraints, it has shown to outperform traditional PDE-based denoising methods and other deep learning methods. However, more research is needed to explore the full potential of this approach and to address some of its limitations. This is the main motivation of utilizing PDEs in denoising image using PIDL. In this study, we will apply PIDL to solve both linear and nonlinear PDEs and how they are applied in denoising image tasks.

3. Problem statement and Algorithms

3.1 Problem Statement and summary of contributions

Despite the success of deep neural networks in performing machine learning tasks such as computer vision and natural language processing, they are also applied to classical problems of applied mathematics to approximate solutions of PDEs. However lack of incorporation of physical laws in deep neural networks algorithms lead to less accurate predictions. Although they may provide a potential solution, but deep neural networks often requires a lot of training data, which is not always available for scientific problems. (Karniadakis, Kevrekidis, Lu, Perdikaris, Wang, and Yang, 2021).Therefore, this project aims to incorporate physical constraints in deep learning algorithms to build a model which is efficient in solving complex nonlinear PDEs. Existing approximation methods for solving PDEs numerically such as FDM, FEM and spectral collocation will be considered.Their performance will be compared with the deep learning model in order to explore both merits and drawbacks of each approach.We exploit the application of PDEs in digital image processing, in particular using the diffusion equation. We demonstrate how PIDL approach can be used to solve practical digital image processing problems and compare it with non-physics inspired convolutional neural network models. In addition we aim to find data driven solution of diffusion equation i.e instead of relying on a theoretical or mathematical model to describe the behavior of the system, the solution will be based on data from experiments or observations.

3.2 Methodology and Algorithms

We begin this section with defining diffusion equation as one example of PDEs used in image denoising tasks. We give an overview of traditional approximation techniques such as FDM, FEM, and spectral collocation for solving PDEs. Later on we build a deep learning model that combine all physical constraints, initial and boundary conditions of PDEs.

3.2.1 The diffusion equation.

Diffusion equation is a parabolic PDEs, which expresses the rate of change of diffusion of heat or temperature with respect to time. It is frequently utilized as a tool for image smoothing and noise reduction in digital image processing.The diffusion equation can be linear PDE or non linear PDE depending on the value of diffusion constant. Isotropic diffusion, which is also referred to as linear diffusion equation and it is characterized by a scalar diffusion value. On the other hand, anisotropic diffusion, also known as nonlinear diffusion, has a diffusion value which varies in terms of spatial variables. Isotropic diffusion equation can be expressed as follows with both initial and boundary conditions.

$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{3.2.1}$$

$$u(x, y, 0) = I_0(x, y) \tag{3.2.2}$$

$I_0(x, y)$ is an initial gray level image with noise. x and y represent spatial variables, $I(x, y)$ is the clean or ground truth image.

$$u(0, y, t) = f_1(y, t) \qquad \text{for } 0 \leq y \leq L_y \qquad \text{(left boundary)} \tag{3.2.3}$$

$$u(L_x, y, t) = f_2(y, t) \qquad \text{for } L_x \leq y \leq L_y \qquad \text{(right boundary)} \tag{3.2.4}$$

$$u(x, 0, t) = g_1(x, t) \qquad \text{for } 0 \leq x \leq L_x \qquad \text{(bottom boundary)} \tag{3.2.5}$$

$$u(x, L_y, t) = g_2(x, t) \qquad \text{for } L_y \leq x \leq L_x \qquad \text{(top boundary)} \tag{3.2.6}$$

$u(x, y, t)$ is the temperature distribution over the image plane, t is time, D is the thermal diffusivity. On the other hand anisotropic diffusion also knwon as Perona-Malik equation (Perona, Shiota, and Malik, 1994) is described by equation (3.2.7). Consider the domain Ω and family of gray level images $I(x, y, t)$ at time $t = 0$ where the input image is $I(x, y, 0)$.Therefore anisotropic diffusion is defined as:

$$\frac{\partial I}{\partial t} = div(c(x, y, t)\nabla I) = \nabla c.\nabla I + c(x, y, t)\nabla^2 I \tag{3.2.7}$$

In anisotropic diffusion, we have stopping functions that are used to control diffusion process.

$$g(b^2) = \frac{1}{(1 + (\frac{dt}{b})^2)} \tag{3.2.8}$$

From the fact that we have two dimensional PDE the domain of definition is square plate ω where g is known as the conductivity function, dt is the time step and D is the diffusion coefficient.The parameter dt helps to regulate the size of the time step during each diffusion iteration. When the time step is excessively large, it can cause excessive smoothing that erases crucial image features. Conversely, if the time step is too small, the diffusion process may take longer to converge, resulting in higher computational expenses.The parameter D aims to adjust the extent of diffusion at each gray-level in the image. A higher D value will result in less diffusion, allowing for the preservation of image edges and other high-frequency components. In contrast, a lower D value will result in more diffusion, producing smoother images.Thus, the D parameter enables us to strike a balance between image smoothing and maintaining its intricate details They use a non-uniform method that decreases the rate of diffusion in areas with a higher probability of being edges. In this study, we will use dirichlet boundary conditions as shown in equation (3.2.3) to (3.2.6).

3.2.2 Finite Difference Method.

The finite difference method (FDM) is a numerical technique used to approximate the solutions of differential equations, such as the diffusion equation, by discretizing the domain into a grid of finite points.The method is based on the principle of approximating the derivatives of the solution at each point using finite differences.To apply FDM to the diffusion equation, we first need to discretize the domain into a grid of points with a uniform spacing Δx, Δy in both the x and y directions. We then approximate the partial derivatives in the diffusion equation using the forward, backward and the central difference.

Consider $2D$ diffusion equation and assume that we have a rectangular domain with sides L_x and L_y and a grid with $N \times M$ points.The grid spacing is given by $\Delta x = L_x/(N-1)$ and $\Delta y = L_y/(M-1)$.The solution $u(x, y, t)$ at the grid point (i, j) is denoted by $u(i\Delta x, j\Delta y, k\Delta t)$, where i and j k are the indices of the grid points.

Suppose that

$$u(i\Delta x, j\Delta y, k\Delta t) = u^k_{i,j}$$

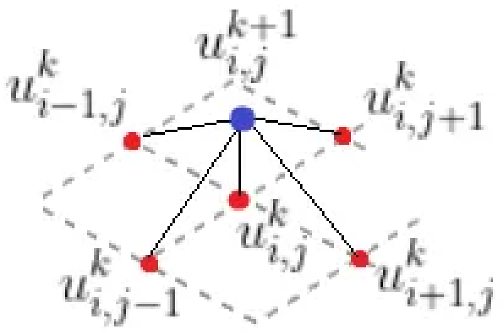
After applying forward difference at left side and central difference at right side, the diffusion equation becomes

$$\frac{u^{k+1}_{i,j} - u^k_{i,j}}{\Delta t} = D\left(\frac{u^k_{i+1,j} - 2u^k_{i,j} + u^k_{i-1,j}}{\Delta x^2} + \frac{u^k_{i,j+1} - 2u^k_{i,j} + u^k_{i,j-1}}{\Delta y^2}\right) \tag{3.2.9}$$

Assume that $\Delta x = \Delta y = 1$ and $\gamma = \dfrac{D\Delta t}{\Delta x^2}$. Then (3.2.9) becomes

$$u^{k+1}_{i,j} = \gamma\left(u^k_{i+1,j} + u^k_{i-1,j} + u^k_{i,j+1} + u^k_{i,j+1} + u^k_{i,j-1} - 4u^k_{i,j}\right) + u^k_{i,j}$$

Given that $x_i = i\Delta x$, $y_i = i\Delta y$ and $t_k = k\Delta t$. where u stands for unknown solution, t represents temporal variable, x and y are spatial variables, and D is diffusivity.The figure 3.1 below illustrates the steps involved in approximating the solution, making it easier to understand



.

Figure 3.1: Finite difference method (Mawuli, 2020)

In the process of FDM, we iteratively update the temperature at every grid point using approximations obtained from finite difference.This update is based on the temperature values of the previous iteration.We can keep repeating this process until a stable solution is reached, where the temperature at each point in the region does not change much. This were chosen for the sake of comparison with PDL. The FDM algorithm can be implemented step by step by following the below algorithm.

Algorithm of finite difference method.

Algorithm 1 Finite Difference Method for 2D Diffusion Equation

```
1: Define boundary conditions and initial conditions.
2: Choose grid spacing $\Delta x$ and $\Delta y$, and time step $\Delta t$.
3: Initialize grid values for $t = 0$ using initial conditions.
4: Set number of grid points $N$ and $M$
5: for $k = 0$ to $N - 1$ do
6:     for $i = 1$ to $N - 1$ do
7:         for $j = 1$ to $M - 1$ do
8:             Update grid values using discretize diffusion equation:
               $$u_{i,j}^{k+1} = u_{i,j}^{k} + \gamma(u_{i+1,j}^{k} - 2u_{i,j}^{k} + u_{i-1,j}^{k}) + \gamma(u_{i,j+1}^{k} - 2u_{i,j}^{k} + u_{i,j-1}^{k})$$
9:         end for
10:    end for
11: end for
```

3.2.3 Finite Element Method.

Finite element method (FEM), is among mesh-based approximation approach for numerical solutions of nonlinear PDEs. FEM discretize the domain into small triangular or quadrilateral elements.The PDE is then approximated by a set of algebraic equations within each element, using interpolation functions that describe the variation of function within each element.The interpolation functions are usually polynomial functions of degree 1 or 2, which are simple enough to calculate and solve but still provide an accurate approximation of the solution (Peiró and Sherwin, 2005). They are necessary conditions to apply finite element method to a given PDE. Mesh representation of the given domain of PDE and boundary conditions that link PDE with the region. In this study we will use dirichlet boundary conditions for simplification. In finite element method we multiply given PDE by the so-called test function and integrate both sides over the whole domain. Specifically let us apply this formula to equation (3.2.1) to obtain.

$$\int_{\Omega} V(x,y)\frac{\partial u}{\partial t}dv = D\int_{\Omega}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)V(x,y)dA \tag{3.2.10}$$

V is the test function, the next step is to do the integration by part to get a weak form.The weak form or weak equation is an alternative formulation of a partial differential equation that is suitable for numerical solution using FEM.

$$\int_{\Omega}\frac{\partial u}{\partial t}dA = \left[V(x,y)D\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right)\right]_{\partial\Omega} - \int_{\partial\Omega} D\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right)\frac{\partial V}{\partial x}dA, \tag{3.2.11}$$

$$\left[V(x,y)D\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right)\right]_{\partial\Omega} = 0, \text{this term is zero on the boundary.} \tag{3.2.12}$$

Using a set of basis functions, we approximate the solution and the test function and plug them into the weak equation. This process results in a set of algebraic equations, which can be expressed in matrix form

$$M\frac{du}{dt} + Au = f \tag{3.2.13}$$

where M represents the mass matrix, A represents the stiffness matrix, and f represents the forcing term.To calculate the mass, the stiffness matrices and the forcing term, we employ the basis functions and their derivatives. Once we obtain these matrices and the forcing term, we can solve the system of algebraic equations using numerical methods such as LU factorization, Gaussian elimination, or iterative methods. After solving the system of equations, we obtain the coefficients of the basis functions, which allow us to obtain an approximation of the solution to the 2D heat equation. By evaluating the solution at each node, we can visualize the temperature distribution across the domain at various time steps. Discretization of the domain into small finite elements is shown in below figure 3.2.This is one dimensional approach but during implementation we have considered two dimensional with respect to the given PDE.

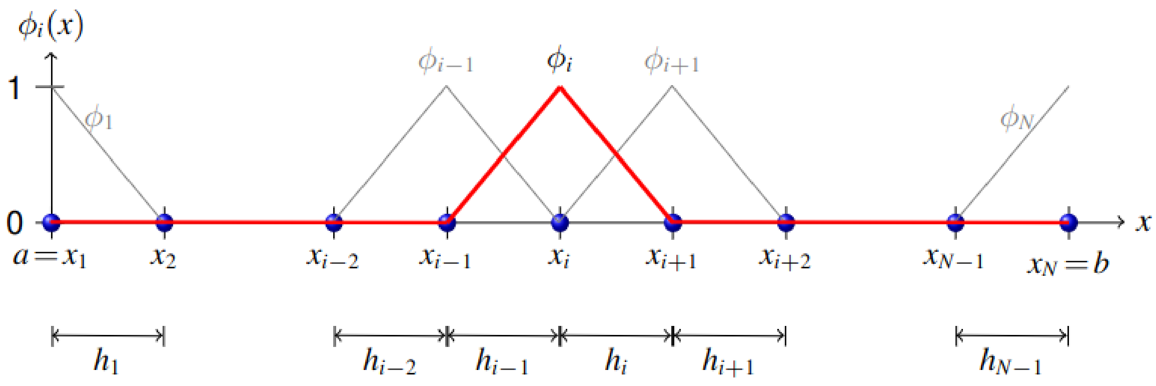


Figure 3.2: Discretization of the domain (Zieli, 1992)

The basis functions, denoted by $\phi_i(x)$ to $\phi_N(x)$, are utilized to provide an approximation of the unknown function across the domain of interest.Typically, basis functions are selected as piecewise polynomial functions that are defined on each element of the mesh. Basis function $\phi_i(x)$ is equal to 1 at node x_j and 0 at all other nodes Equation (3.2.14) and (3.2.15) shows mathematical expressions of basis functions

$$\phi_1 = \begin{cases} \dfrac{x_2 - x_1}{h_1} & for x \in d \\ 0 & otherwise \end{cases} \tag{3.2.14}$$

$$\phi_N = \begin{cases} \dfrac{x - x_{N-1}}{h_{N-1}} & for x \in d_{N-1} \\ 0 & otherwise \end{cases} \tag{3.2.15}$$

From the figure 3.2 the notation $X_N = b$ is to mean the diffusion constant of the given PDE

Algorithm for finite element method

Algorithm 2 Finite Element Method general algorithm

```
1: descretize the domain $\Omega$ and define the corresponding space of piecewise linear functions $\phi(x, y)$
2: Assemble the $N \times M$ stiffness matrix A and then $N \times 1$ load vector b, with entries
3: $$A_{i,j} = \int_{\Omega} \nabla\phi_j \nabla\phi_i dA$$
4: $$b_i = \int_{\Omega} f\phi_i dA$$
5: Apply boundary conditions to $A$ and $b$ to obtain modified $A_{ij}$ and $b$
6:    solve the linear system $$Au = b$$
7: Set $$u = \sum_{j=1}^{n_i} u_j\phi_j$$
8: Solve $Au = b$ for $u$
```

3.2.4 Spectral collocation Method.

Spectral collocation methods are a class of numerical methods used to solve PDEs by approximating the solution as a sum of basis functions defined on a set of collocation points.The basic idea behind spectral collocation methods is to approximate the solution of a PDE using a linear combination of basis functions that are defined on a set of collocation points.The collocation points are chosen in such a way that they are evenly spaced and usually chosen based on a Gauss-Lobatto or Gauss-Chebyshev quadrature rule.This rule involves approximating the integrals that appear in the collocation equations using a weighted sum of function values at a set of collocation points. By choosing a sufficiently large number of collocation points, we can obtain an accurate approximation of the solution to the 2D heat equation.

We begin by approximating the unknown solution of a given PDE ($\mathcal{L}u = f$)

$$u(x, y, t) = \phi_0(x, y, t) + \sum_{n=1}^{N} a_n(t)\phi_n(x, y, t) \tag{3.2.16}$$

$u(x, y, t)$ is the unknown solution that we need to approximate, $\phi_n(x, y, t)$ are the chosen spatial basis or trial function, $a_n(t)$ are time-independent coefficients and ϕ_0 is chosen to satisfy boundary conditions such that $\phi_n = 0$, $N = 1, \cdots N$ at the boundaries.

Because the numerical solution $u(x, y, t)$ is only approximated, it does not satisfy the original

PDE. Substituting its expansion into the given PDE we define a residue

$$r = f - \mathcal{L}u \tag{3.2.17}$$

The following algorithm shows step by step implementation of spectral collocation method.

Algorithm 3 Spectral Collocation Method for solving the 2D Diffusion Equation

```
1: Domain $[0, L_x] \times [0, L_y]$
2: Number of collocation points $N_x, N_y$
3: Diffusion coefficient $D$
4: Time step $\Delta t$
5: Final time $T$
6: Initial condition $u_0(x, y)$
7: Solution $u(x, y, t)$ to the 2D diffusion equation after time $T$
8: Construct the collocation points $x_i$, $y_j$ and differentiation matrices $D_x$, $D_y$ using Gauss-
   Chebyshev quadrature rule.
9: for $n = 1$ to $N_t$ do
10:     Compute the Laplacian $\nabla^2 u$ using the differentiation matrices
11:     Update $u$ using the equation
12:     $\frac{\partial u}{\partial t} = D\nabla^2 u$
13:     Apply boundary conditions to $u$
14: end for
```

3.2.5 Physical Inspired deep learning method.

PIDL method incorporates the physical properties of the underlying PDE into the training process, instead of relying solely on the available data.The use of the PDE's governing equation helps to guide the training process.The incorporation of known data points on top of the physics-based loss function, thereby increasing the speed of training. Similar to using $L2$ regularization during training a physics-inspired neural network also aims to reduce data loss while incorporating known physics as an extra regularization factor. In other words, the network is trained to match the available data but with the added constraint to ensure that the solution is compatible with the known physics equations.The goal of the network is to learn f(t) through unsupervised learning on the provided data. It follows the data loss function by calculating the Mean Squared Error (MSE) in terms of sum of a residue function, initial and boundary conditions of the given PDEs. The standard MSE formula is given by

$$\mathcal{L}_{data} = \frac{1}{n}\sum_{i=1}^{n}(s_i - \hat{s}_i)^2 \tag{3.2.18}$$

where s_i and $\hat{s_i}$ are true value and predicted value respectively. In order to use a neural network to solve a PDE, the network is trained on a specific region or domain where the PDE is valid and applicable.

Consider the following Cauchy problems of PDEs:

$$\mathcal{L}u = f(t,x,y) \in [0,T] \times \Omega \tag{3.2.19}$$
$$Iu = g(t,x,y) \in \{0\} \times \Omega \tag{3.2.20}$$
$$Bu = h(t,x,y) \in [0,T] \times \partial\Omega \tag{3.2.21}$$

The symbols used in the context refer to various operators and terms in a differential equation. Specifically, $\mathcal{L}$ represents the differential operator, I and B represent the initial and boundary operators respectively. The inhomogeneous terms, initial data, and boundary data are represented by f, g, and h respectively (Son, Jang, Han, and Hwang, 2021). A neural network is trained on uniformly sampled grid points $\{t_i, x_j\}_{i,j}^{N_b,N_x} \in [0,T] \times \Omega$.To ensure that a neural network satisfies the PDE, we have to minimize a physical loss function Φ_θ. The loss function contains both boundary and initial conditions

$$\phi_\theta := \Phi_\theta^r(X^r) + \Phi_\theta^0(X^0) + \Phi_\theta^b(X^b) \tag{3.2.22}$$

X is the collection of training data, $\Phi_\theta^r(X^r)$ which is the mean square error for residue and it is defined as

$$\Phi_\theta^r(X^r) := \frac{1}{N_r}\sum_{i=1}^{N_r} |\mathcal{L}u_\theta - f(t_i^r, x_i^r, y_i^r)|^2 \tag{3.2.23}$$

where X^r represents the number of collocation points, r_θ is the physics inspired neural network and $\mathcal{D}$ is the domain of definition of the given PDE. Hence, $X^r := \{(t_i^r, x_i^r)\}_{i=1}^{N_r} \in [0,T] \times \mathcal{D}$. We have

$$\Phi_\theta^0(X^0) := \frac{1}{N_0}\sum_{i=1}^{N_0} |Iu_\theta - g(t_i^0, x_i^0, y_i^0)\|^2 \tag{3.2.24}$$
$$\Phi_\theta^b(X^b) := \frac{1}{N_n}\sum_{i=1}^{N_b} |Bu_\theta - h(t_i^b, x_i^b, y_i^b)|^2 \tag{3.2.25}$$

where $\Phi_\theta^0(X^0)$ and $\Phi_\theta^b(X^b)$ are mean square errors for initial and boundary conditions respectively (Blechschmidt and Ernst, 2021). Now we have all the requirements for a neural network. Hence we can minimize the residue function which contain physical constraints of the given PDE with both initial and boundary functions.

3.2.6 Algorithm for PIDL.

Note: We do not have direct target values for the predictions, but we will train sample data set and add variable learning rate so as to increase the accuracy. Sample collocation points have to satisfy the PDEs initial and boundary conditions. In other word we perform unsupervised learning

Algorithm 4 Physics-Inspired Deep Learning for Image Processing

```
1: generate data
2: Initialize the domain at $t = 0$ with initial conditions
3: Define neural network architecture
4: Define the loss function
5: Train network by optimizing loss function with gradient based optimizers
```

3.2.7 The architecture of PIDL model for image denoising.

The neural network architecture is established using the Python library TensorFlow. The model comprises four 2D convolutional layers, with each layer consisting of 64 filters of size 3×3. The activation function used in each layer is Rectified Linear Unit (ReLU), which is known for its advantages in neural networks, including faster training times and addressing the issue of vanishing gradients that may arise with other activation functions.

Mathematically, convolutional layer is expressed as

Let
$$X \in \mathbb{R}^{K \times K \times C}$$
where x is $2D$ object with C channels.

Let s be a small integer and $s \times s$ represent kernel dimensions. For our model $s = 3$, we define weights for the model as $w \in \mathbb{R}^{s \times s \times C}$ and bias $b \in \mathbb{R}$. Hence the kernel applied to the position i, j where $1 \leq i, j \leq K$ can be expressed as

$$z_{i,j} = \sum_{\alpha=0}^{k-1} \sum_{\beta=0}^{k-1} \sum_{\gamma=0}^{C} w_{\alpha,\beta,\gamma} x_{i+\alpha}, j + \beta, \gamma + b \tag{3.2.26}$$

Then the activation is defined as usual $a_{i,j} = \sigma(z_{i,j})$ for an activation function $\sigma : \mathbb{R} \to \mathbb{R}$. In addition to that, the operation of applying a filter to a position is called convolution. One kernel is applied to all input coordinates with the same weights also known as weight sharing. In order to have output of the same dimension as input image we do padding which is the same for all layers.

One convolutional layer with padding, the input has shape $K \times K \times C$ and we apply D filters with dimensions $k \times k$. We have, K^2C neurons (activations) in the input to the layer. To find number of weights and biases for each layer we use the following formulas. K^2D neurons in the output of the layer, k^2CD weights and number of filters D are biases. Bellow is the algorithm which helps to understand the full architecture.

Algorithm 5 Convolutional Neural Network with 64 3x3 filters and 9 layers for Image denoising

```
1: Input: tensor of size (size x size,1)
2: Layer 1: Convolution layer with 64 filters of size 3x3 and stride 1
3: Activation: ReLU activation function
4: Layer 2: Convolution layer with 64 filters of size 3x3 and stride 1
5: Activation: ReLU activation function
6: Layer 3: Convolution layer with 64 filters of size 3x3 and stride 1
7: Activation: ReLU activation function
8: Layer 4: Convolution layer with 64 filters of size 3x3 and stride 1
9: Activation: ReLU activation function
10: Layer 5: Convolution layer with 64 filters of size 3x3 and stride 1
11: Activation: ReLU activation function
12: Layer 6: Convolution layer with 64 filters of size 3x3 and stride 1
13: Activation: ReLU activation function
14: Layer 7: Convolution layer with 64 filters of size 3x3 and stride 1
15: Activation: ReLU activation function
16: Layer 8: Convolution layer with 64 filters of size 3x3 and stride 1
17: Activation: ReLU activation function
18: Layer 9: Output layer with one filter of size 1x1
```

4. Implementation and Discussions

In this section, we solve heat equation and we use laplacian filter for image denoising task using both traditional approximation method of solving PDE and PIDL techniques. We will compare the accuracy of both approaches. We will add Gaussian noise to a clean image, and then use PDE algorithms to clean the noisy image. Both isotropic and anisotropic diffusion methods will be used to obtain a clean image.

4.1 Implementations

Solution of two dimensional heat Equation using Traditional Methods

Traditional methods are known as mesh-based methods as discussed in section 3.2.These methods involve discretizing the domain into smaller geometric shapes, and then computing the solution by summing up the solution over each shape to obtain the final solution of the given PDE. Figure 4.1 below shows results obtained when approximating 2D heat equation using FDM and FEM.

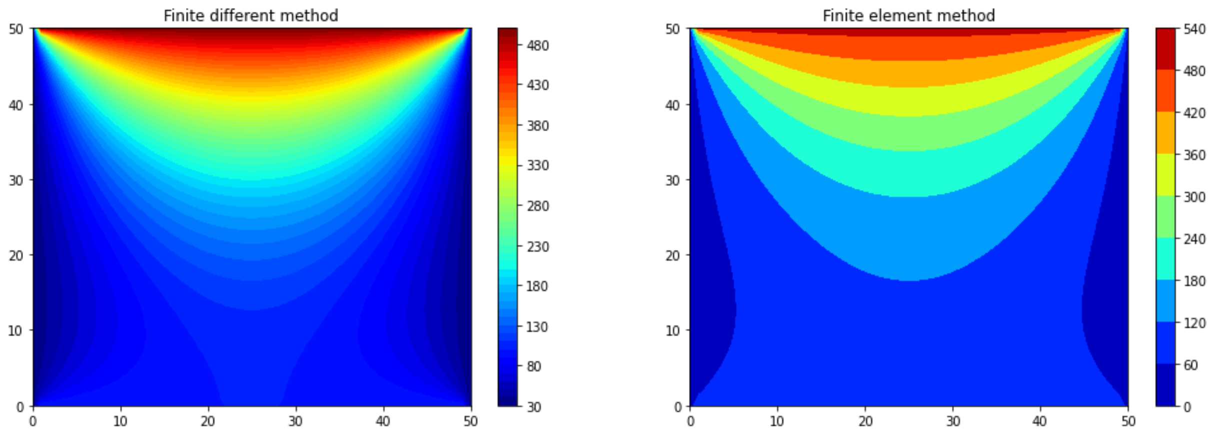


Figure 4.1: FDM and FEM.

From figure 4.1 we have used 1000 iterations, top_temperature of 500 and remaining sides have the temperature of 30 at time $t = 0$. All of these temperatures act as initial and boundary conditions. In this study, for all methods we applied same initial and boundary conditions.The goal is to determine the temperature distribution in a square domain whose sides are Lx and Ly and is divided into smaller squares with Nx and Ny grid points in the x and y directions, respectively. We represented the distance between the grid points in each direction by dx and dy. Algorithms $1\&2$ of section 3.2 describe the entire process of FDM and FEM approaches. From figure 4.1 above, as time changes the temperature spreads throughout all the side of the material starting from top to bottom. From the fact that heat flows from a high-temperature medium to a low-temperature medium, to spread faster or slow depends on physical constraint known as diffusion constant. It is a property of the material that determines how quickly the thermal energy is transported through the material, where a higher diffusion constant leads to faster heat transfer, while a lower diffusion constant leads to slower heat transfer, which is the case of FEM. In addition, we approximated the same PDE with spectral collocation and FVM, the figure **??** below illustrates the spread of heat over the domain.

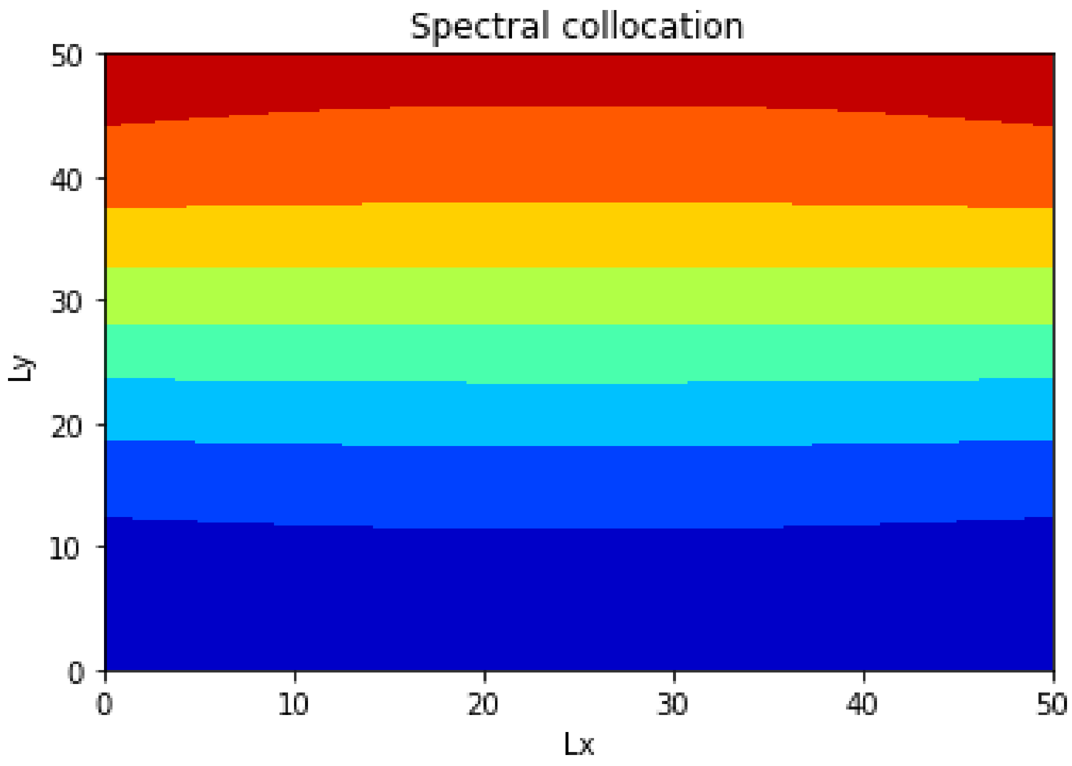


Figure 4.2: Spectral collocation method.

From figure 4.2 heat diffuses quickly in all parts of the domain as the time changes. This is because in spectral collocation technique we use non-uniform grid known as Gauss-Chebyshev points, which is designed for precise approximation. We have also used differentiation matrix which allows the computation of derivatives of a function at the collocation points and enables the approximation of heat equation. Although the spectral collocation show good results, the diffusion process is not clear.

Solving heat equation using PIDL

We have used unsupervised learning to predict the solution of 2D heat equation. One of the ways to get data for training model is to sample data from initial and boundary conditions. We created an instance of the Latin Hypercube engine from the quasi-monte carlo method (qmc) module and set the dimension of the sampling space to 2. Latin Hypercube Sampling (LHS) is a type of quasi-Monte Carlo sampling that tries to better cover the sampling space by partitioning it into equally-sized intervals and randomly sampling from each interval once.Then we define the dimensionality d of the sampling space, i.e., the number of variables that need to be sampled.

Figure 4.3 below shows obtained data points.

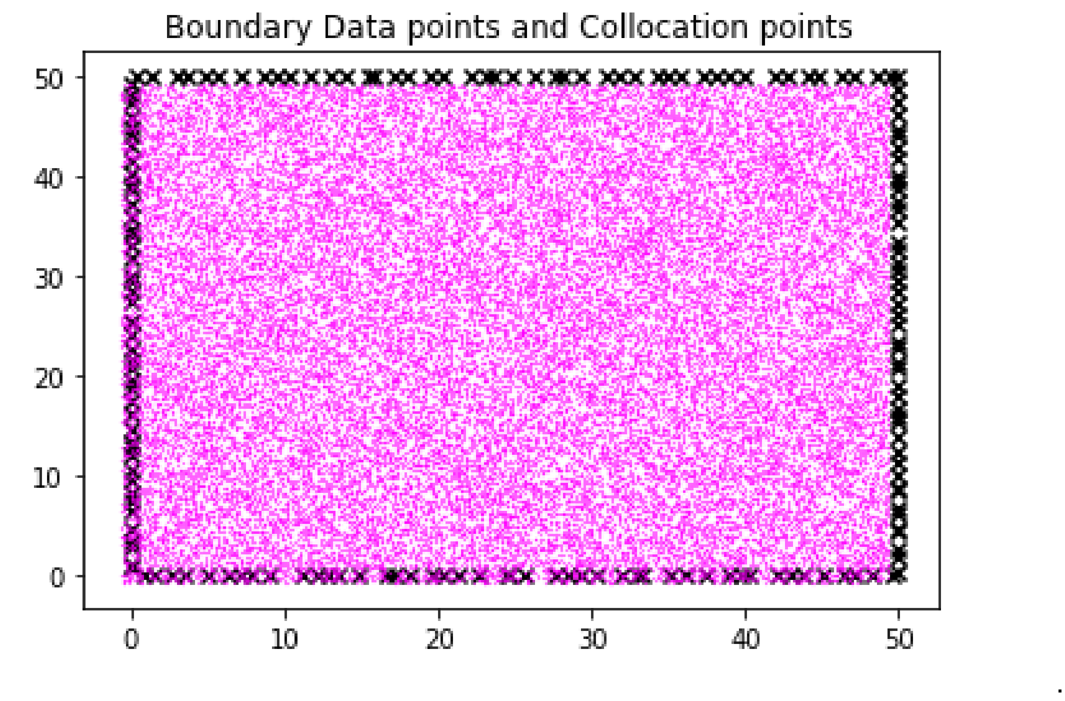


Figure 4.3: Sample data points

We built a deep learning model to find a solution based on the dataset we collected.The model has 10 hidden layers, with a total of 3,861 trainable parameters, and 20 neurons in each hidden layer. Figure 4.4 shows the results of our experiment of how heat is diffusing throughout the defined area, as determined from the data we collected.Total collocation points generated are 25000 and 25 point for each boundary of the domain.

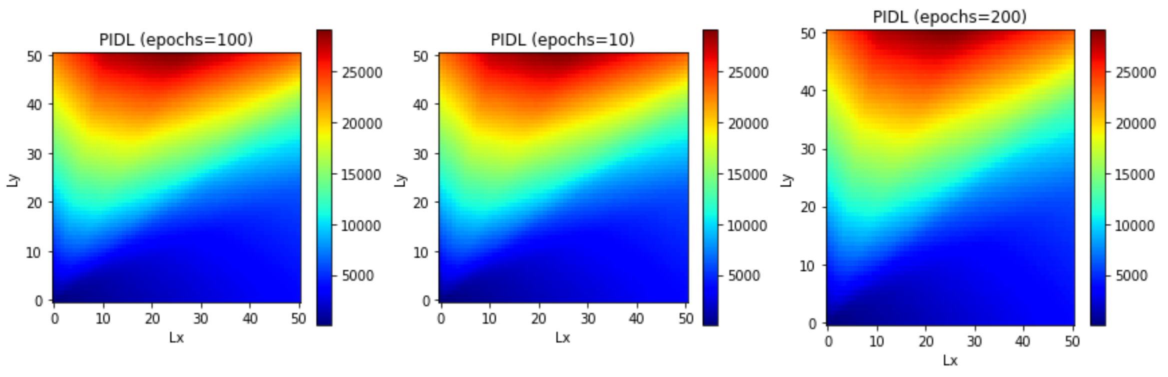


Figure 4.4: Solution of diffusion equation with PIDL.

The accuracy of a prediction in a deep learning model depends on several factors, including the number of epochs and the number of hidden layers in the model and the trainable parameters.

In the current experiment, we have utilized 10 layers, 100, 10 and 200 epochs to make our predictions.The results of our experiment show the diffusion of heat to all sides of the plate. By compare the results of this experiment with the solution obtained using the traditional methods, we observe that variable number of epochs and collocation points in the PIDL, resulted in a solution that is more realistic compared to that obtained using the traditional methods.This indicates that PIDL gives results that is highly accurate. We have used the same initial and boundary conditions as the traditional methods such as 1000 iterations, top_temperature of 500 and remaining sides have the temperature of 30 at time $t = 0$. As the number of epochs changes, heat diffuses quickly across the entire plate (domain), as shown in figure 4.4. The complexity of the model as well as the number of epochs affect accuracy. In this case, we can say that the accuracy increases as the number of epochs increases with the large amount of trainable parameters and loss decrease. In this study we have semi-logarithmic scale improve the ability to visualize both large and tiny loss numbers. Changes in the loss are more visible on the logarithmic scale, especially when the range of loss values is large. Below figure 4.5 represent who loss varies.

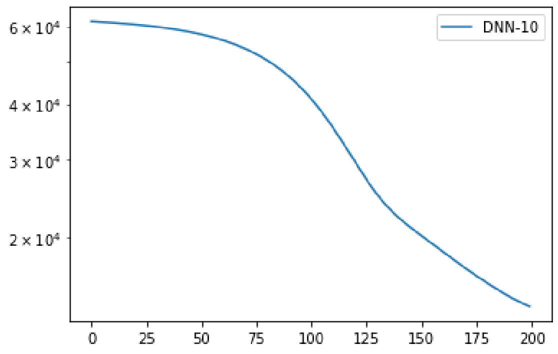


Figure 4.5: Variation of loss values

4.2 Image denoising

Image noise is the irregular variation of brightness or color information within captured images, which is caused by external sources and lowers the quality of the image. Image denoising is a fundamental procedure in image processing that aims to eliminate unnecessary noise from an image while maintaining its most important features. To de-noise gray-scale images, we began with traditional methods such as FDM and FEM. Deep learning algorithm particularly PIDL is used to perform the same task where the obtained results has to be compared with one for classical methods. When denoising mage we take an original or clean_image and add some noise to it. There are different type of noises such as guassian noise, salt and pepper noise, poison noise, impulse noise, and speckle noise. In this study, we have applied guassian noise, which is a type of statistical noise that follows a normal distribution and it is uniformly distributed across the entire image (Lendave, 2021). To generate a noisy image, each gray-level of the original image is added with a random value following the Gaussian probability distribution function. We used mean $\mu = 0$ and standard deviation $\sigma = 25$. This added value is known as Gaussian noise, and it contributes to the final gray-level value of the noisy image. The figure 4.6 below displays the probability distribution function of Gaussian noise as well as the way Gaussian noise is represented in gray-levels.

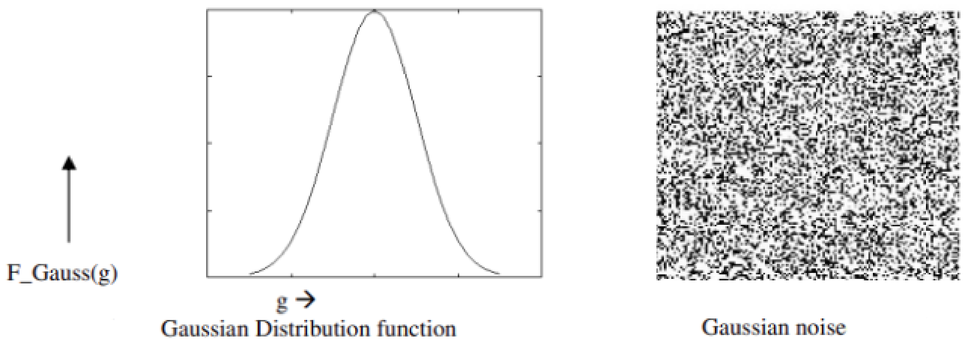


Figure 4.6: Gaussian distribution and Guassian noise (Lendave, 2021).

Mathematically Guassian noise is written as probability density function of normal distribution

$$G_\sigma(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|X\|^2}{2\sigma^2}\right) \tag{4.2.1}$$

where σ is the standard deviation and X is random noise.The images on the left and right hand sides of the figure 4.7 bellow are clean and noisy gray_scale images that are being used in our experimentation. Implementation were done using python and TensorFlow library where the size of the original gray_scale has size of 256×256 and the noisy_image has the same size.

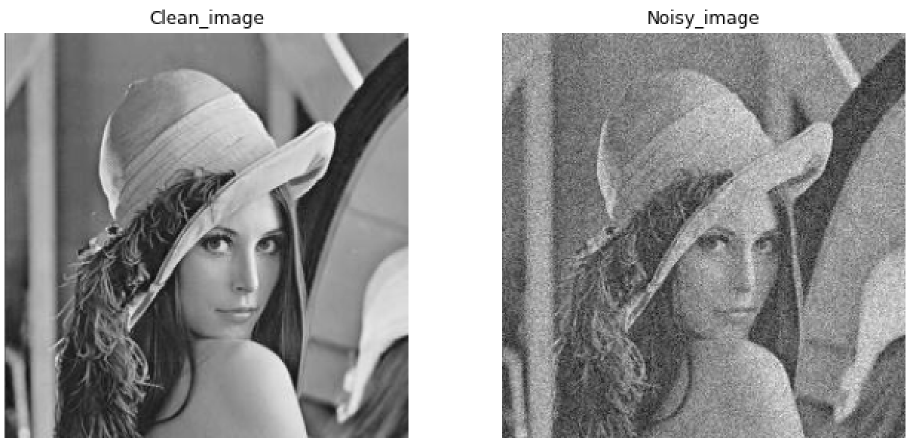


Figure 4.7: Adding noise to a clean image

The image with added noise is transformed into a heat equation with constant diffusion and then approximated using both FDM and FEM.The output image is free of noise but not clear as the original image.This situation is caused by two reasons, firstly, when a large amount of noise is removed from an image, important details and edges can also be lost, resulting in a blurry image.The second reason is that constant diffusion smooths the image in the same way in all directions.This can make the image lose important details and become blurry when some of its important features are being removed with noise. Figure 4.8 below shows isotropic diffusion using traditional methods

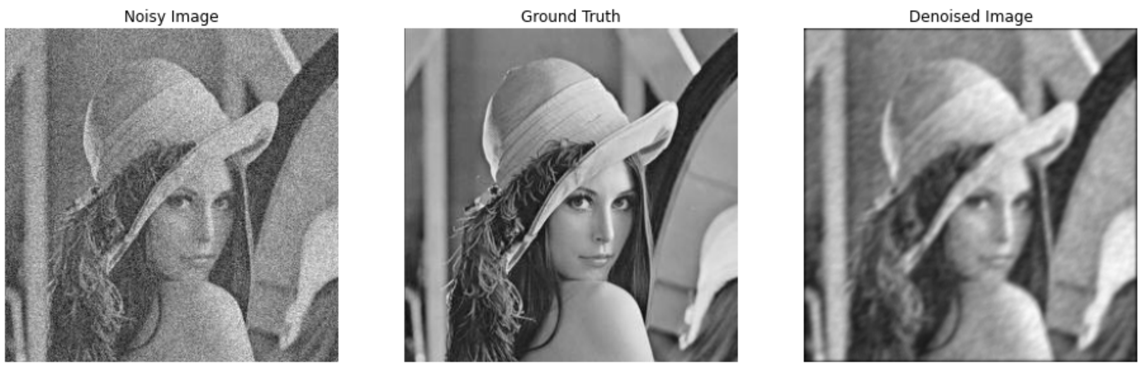


Figure 4.8: Denoising image by traditional methods

To enhance the smoothness of the output image, we employed alternative techniques such as nonlinear diffusion, specifically anisotropic diffusion using Perona-Malik formula refer to section 3.2.This method aims to reduce image noise and improve overall image quality. Furthermore, we conducted a gray-level-by-gray-level comparison between the denoised image and the original image. If the gray-level difference exceeds 0, it indicates the presence of residual noise that has not been adequately removed. In our case, the difference observed after applying anisotropic diffusion was measured at 25 gray-level while for isotropic diffusion was 112 gray-levels.The results of our experiments, presented in figure 4.9. It illustrate the impact of these methods on the output image quality.

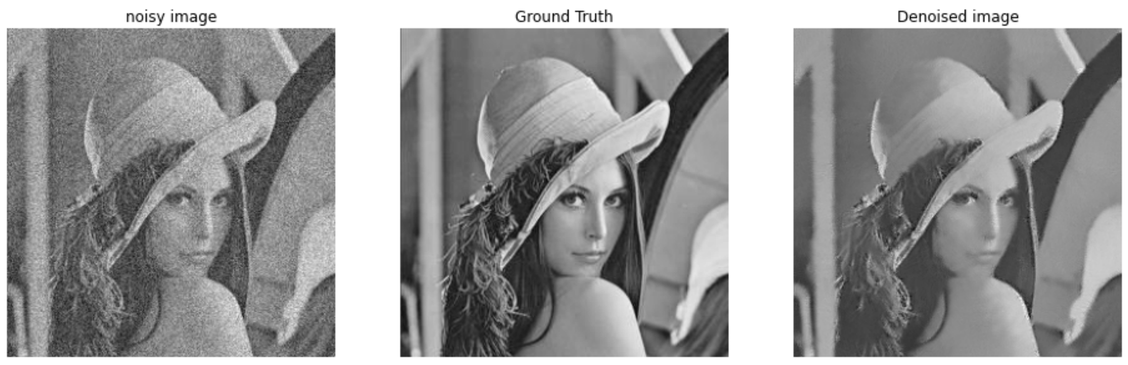


Figure 4.9: Denoising image by anisotropic diffusion

After applying conventional methods, we employed PIDL and contrasted the outcomes with the former methods. CNN model was used to denoise a gray_ scale image.The CNN model was trained with a regularization term, of the heat equation, added to the loss function as well as Laplacian filter.The heat equation was used as a way to impose a smoothness constraint on the images during the denoising process. The goal was to produce images with higher quality and less noise.The results of the study are presented in Figure 4.10 below.

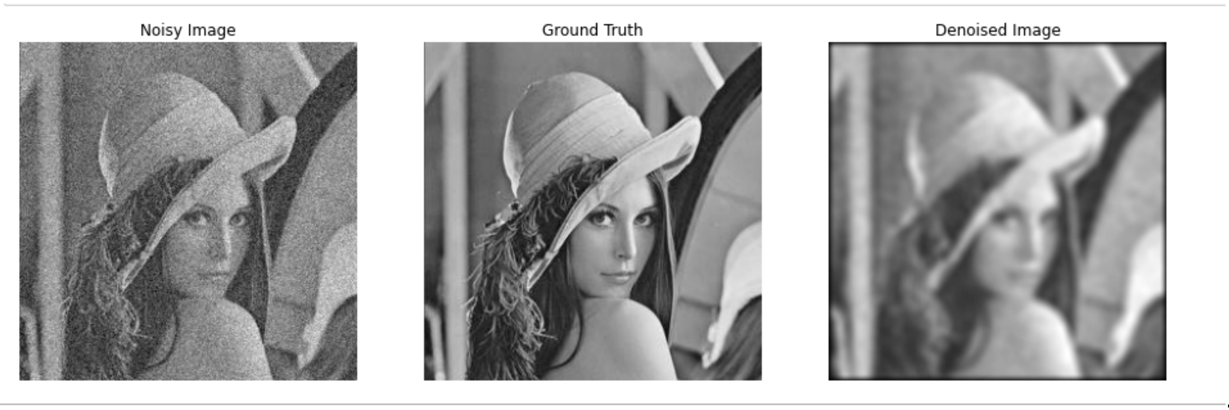


Figure 4.10: Denoising image by PIDL

Difference approaches of improving model performance have been done. We have chosen suitable learning rate and added more convolutional layers. Metrics that were used to evaluate the performance of the model during training is mean square error between denoised image and original image. Secondary is the peak signal-to-noise ratio (psnr), which measures the quality of the reconstructed image compared to the original clean image.The results obtained were similar to that of anisotropic diffusion using traditional methods. The figure 4.11 below shows the obtained results.

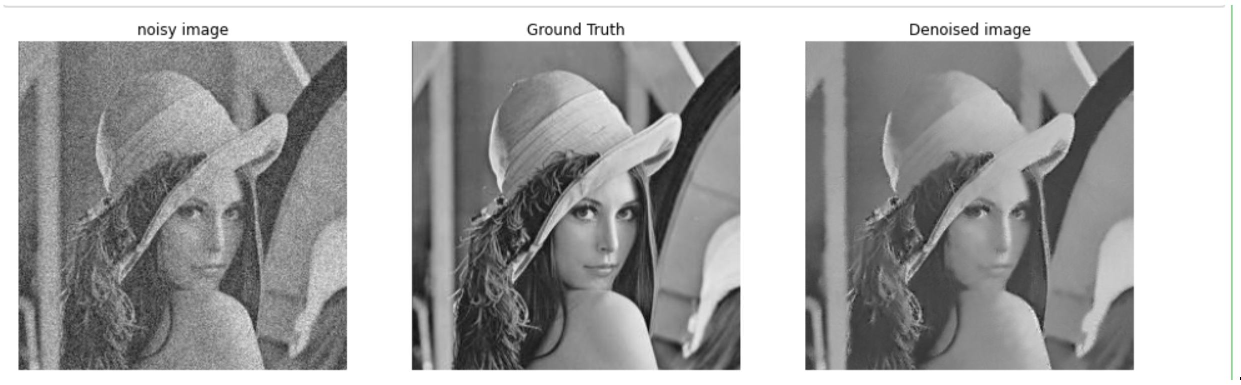


Figure 4.11: Denoising image by PIDL

4.3 Discussions

Data generation according to initial and boundary conditions

In most traditional methods, excluding spectral collocation, the data used consists of points on a grid obtained by dividing the domain into smaller parts. For PIDL, however, it is different since before producing data, numerous factors need to be taken into account. In order to train a deep learning model, it is necessary to have data for training. In this study we have done unsupervised learning, where we only had knowledge of the initial and boundary conditions within a defined domain. The domain is a rectangular region that measures of length $L_x$and width Ly, with a grid spacing of 0.02 in both the x and y directions, resulting in a grid composed of 50×50 points. The upper boundary is set to a temperature of 500 degrees Celsius, while the other three boundaries are set to 30 degrees Celsius.These boundaries are known as Dirichlet boundary conditions. Initially, the temperature is set to 30 at all points within the domain, and for the boundaries we applied the Dirichlet boundary conditions. By using this information, we were able to predict the solution, based on the generated collocation points.The solution obtained through the use of generated data is known as a data-driven solution. Although the same initial and boundary conditions are used for both FDM,FEM and PIDL, the solution of PIDL was more realistic compared to that of traditional methods.

Choosing time steps and diffusion constant

After a specific interval of time, the heat spread from the area of high temperature to the area of low temperature.This time period was chosen in this study as one method of assessing and judging the approximation of the solution to the heat equation over a defined domain. As a parabolic partial differential equation with diffusion-like behavior, to approximate the 2D heat equation we have used a robust time integration approach. All the computations were done using python, therefore the choice of time step we refer to is Crank-Nicolson approach, $dt = \frac{(dx)^2}{4D}$, D is the diffusion constant.Choosing time step using this approach is known to be more accurate and stable than other known approaches because it is an integration path of the heat equation. After choosing time step, there is another important aspect which governs the diffusion known as diffusion constant. According to Perona- Malike we used the so-called stopping function in contrast to normal constant diffusion especially when denoising images.

Avoiding over-fitting of deep learning model

After producing training data, it is recommended to add some noise so that the predictability of the model may be assessed.This method allows the model to distinguish between data points and noise. When we applied it to the image which needs to be denoising, the results revealed that some essential components of the image were also removed, and they were comparable to the outcomes of anisotropic diffusion using conventional techniques. In this study, we tested our PIDL model using a single sample gray-scale image, thus regardless of whether we had a massive dataset, the model would continue to operate effectively.

We have added the Gaussian noise to the clean image and it does not affect the image structure. It is a useful option for modeling small changes or variations in the data because it is a continuous random variable that can take on any value within a specific range with an appropriate probability.

PIDL model for image denoising

When denoising grayscale photos, we used supervised learning to train the CNN model.The model was created while taking into account the physical constraints that the diffusion equation's Laplacian operator encodes. A number of convolutional layers are present in the model design, which is followed by an output layer with linear activation.To ensure that the output image is the same size as the original image, we applied padding that is the same for all layers.The filter uses the Laplacian operator and a customized loss function that is based on the 2D heat equation.To calculate the regularization term, the heat equation is defined using finite differences and applied to the wanted image.The loss function is a combination of the data residue function term and the regularization term.

Both the clean or ground truth image and the noisy image have been used to train the model. During training, an Adam optimizer with a $1e-4$ learning rate and a unique loss function are used.The input image is denoised using the trained model after training and there is lack of clarity in the resulting image. However, we suggest that additional adjustments are required to increase accuracy.The Laplacian operator may not be accurate to denoise image. Additionally, a more complicated architecture could be used to enhance the model, like a U-Net, to better capture the underlying patterns in the data.

5. Summary and conclusions

5.1 Summary

Approximating the numerical solution of PDEs, in particular 2D heat equation was done by different approaches such as mesh-based and mesh-free methods. Mesh-based methods are expensive and time-consuming when approximating complex PDE systems like high-dimensional problems.They require many mesh points to accurately represent the domain of interest, which the main reason to become computationally expensive. As the number of mesh points increases, lead to high computational cost and memory requirements. Furthermore, because changing the discretization of the domain requires a large computational cost, mesh-based approaches are not highly flexible and adaptable. In contrast, PIDL method is more flexible and adaptable, which makes it to become a better choice for complex problems.

For the task of image denoising, FDM can capture the second-order derivative of an image accurately, and it is computationally efficient. It can handle non-linear and non-stationary noise but can be challenging to generate high-quality mesh for image structures. In contrast FEM is more flexible and can handle complex geometries more easily. From the fact that it has a test function when multiplied to the given PDE we obtain a weak function.This becomes easy to be approximated. It can capture higher-order derivatives of the image, but it is more computationally intensive than FDM. However FEM may be less accurate when dealing with non-linear or non-stationary noise similar to FVM and spectral collocation. PIDL is adaptable and can learn complex relationships between input and output, but it requires high-quality and quantity of training data. Ultimately, the choice of method depends on the specific problem and available resources.

PIDL methods have the benefit of being more versatile and adjustable than mesh-based methods, since they don't need a predetermined mesh to depict the domain of interest. Rather, they depend on a neural network to estimate the PDE's solution directly, making them useful for dealing with intricate geometries and unorthodox boundary conditions. Furthermore, PIDL can be quicker than conventional mesh-based methods.

However,PIDL methods may require large amounts of data to accurately represent the domain of interest, and they may be limited by the available computational resources. Additionally, their effectiveness can be sensitive to the quality and quantity of training data, and they may require careful when choosing metrics to achieve optimal results. In addition to that when one manage to get data, it is not that easy to split data into training and test set because generated data are not enough to the extent they can be shared.The model may experience over-fitting due to the random sampling of generated data.

5.2 Conclusion

The use of PIDL as a method for approximating complex systems for PDEs is a promising approach, particularly for image denoising and approximating PDE solutions.The solution obtained using PIDL is more realistic compared to that of traditional methods. Although the algorithm for this method is short and simple, the incorporation of physical constraints into a defined loss function remains a challenging task. Further research and time is required to explore this area and address this challenge.

In image denoising, using PDEs as a filter function, the resulting output image may not be as clear as the original image due to inaccuracies in Laplacian function or gradient calculations. Thus, it is essential to ensure that the denoised image and the original image have similar or equal gray-levels.This can be accomplished by calculating the difference between the gray-levels in both images. If the difference is significant, the noise has not been removed, and the process must be repeated. Accurately defining the Laplacian filter is critical in achieving a high-quality output image also the type of noise matters when performing image denoising task.This study focused on verifying the effectiveness of the PIDL model for gray images. However, for future research, we suggest using larger datasets that contain a variety of images, including colored images, to evaluate the PIDL model accuracy more comprehensively.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Klaus-Jürgen Bathe. Finite element method. *Wiley encyclopedia of computer science and engineering*, pages 1–12, 2007.

Christian Beck, Martin Hutzenthaler, Arnulf Jentzen, and Benjamin Kuckuck. An overview on deep learning-based approximation methods for partial differential equations. *arXiv preprint arXiv:2012.12348*, 2020.

Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.

Jan Blechschmidt and Oliver G Ernst. Three ways to solve partial differential equations with neural networks—a review. *GAMM-Mitteilungen*, 44(2):e202100006, 2021.

Kristian Bredies, Dirk Lorenz, et al. *Mathematical image processing*. Springer, 2018.

Martin Burger and Carola-Bibiane Schönlieb. Image denoising: Can plain neural networks compete with bm3d? *Advances in neural information processing systems*, 25:1031–1039, 2012.

Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

DM Causon and CG Mingham. *Introductory finite difference methods for PDEs*. Bookboon, 2010.

Antonin Chambolle. Partial differential equations and image processing. *In Proceedings of 1st International Conference on Image Processing*, 1(1):16–20, 1994.

Akshay L Chandra and McCulloch-Pitts Neuron—Mankind's First. Mathematical model of a biological neuron. *Published in Towards Data Science*, 2018.

Bruno Costa. Spectral methods for partial differential equations. *CUBO, A Mathematical Journal*, 6(4):1–32, 2004.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

Guy Gilboa, Nir Sochen, and Yehoshua Y Zeevi. Forward-and-backward diffusion processes for adaptive image enhancement and denoising. *IEEE transactions on image processing*, 11(7): 689–703, 2002.

Sam Greydanus, Misko Huang, and Brandon Moore. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, pages 15382–15393, 2019.

Yanan Guo, Xiaoqun Cao, Bainian Liu, and Mei Gao. Solving partial differential equations using deep learning and physical constraints. *Applied Sciences*, 10(17):5917, 2020.

M Yousuff Hussaini, David A Kopriva, and Anthony T Patera. Spectral collocation methods. *Applied Numerical Mathematics*, 5(3):177–208, 1989.

Ameya D Jagtap and George E Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. In *AAAI Spring Symposium: MLPS*, pages 2002–2041, 2021.

Kyong Hwan Jin, Eunji Lee, Junghoon Lee, and Yongduek Song. Deep convolutional neural network for image deconvolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 26–33, 2017.

Sijie Jin, Zhen Zhang, Yufei Su, and George Em Karniadakis. Learning to solve high-dimensional partial differential equations via deep learning. *arXiv preprint arXiv:1906.02355*, 2019.

Nathan Jordre. Estimating the heat equation using neural networks. 2021.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Karthik Kashinath, M Mustafa, Adrian Albert, JL Wu, C Jiang, Soheil Esmaeilzadeh, Kamyar Azizzadenesheli, R Wang, A Chattopadhyay, A Singh, et al. Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, 379(2194):20200093, 2021.

Amirhossein Kazerouni, Ehsan Khodapanah Aghdam, Moein Heidari, Reza Azad, Mohsen Fayyaz, Ilker Hacihaliloglu, and Dorit Merhof. Diffusion models for medical image analysis: A comprehensive survey. *arXiv preprint arXiv:2211.07804*, 2022.

Stefan Kollmannsberger, Davide D'Angella, Moritz Jokeit, Leon Herrmann, Stefan Kollmannsberger, Davide D'Angella, Moritz Jokeit, and Leon Herrmann. Physics-informed neural networks. *Deep Learning in Computational Mechanics: An Introductory Course*, pages 55–84, 2021.

Seid Koric and Diab W. Abueidda. Data-driven and physics-informed deep learning operators for solution of heat conduction equation with parametric heat source. *International Journal of Heat and Mass Transfer*, 203:123809, 2023.

Vaishnavi Lendave. A guide to different types of noises and image denoising methods. *Analytics India Magazine*, 22, 2021.

Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.

Xin-Yang Liu, Hao Sun, and Jian-Xun Wang. Predicting parametric spatiotemporal dynamics by multi-resolution pde structure-preserved deep learning. *arXiv preprint arXiv:2205.03990*, 2022.

Yasaman Lotfi and Kourosh Parand. Efficient image denoising technique using the meshless method: Investigation of operator splitting rbf collocation method for two anisotropic diffusion-based pdes. *Computers & Mathematics with Applications*, 113:315–331, 2022.

Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

Prosper Mawuli. Solving 2d heat equation numerically using python. *Level Up Coding*, November 2020.

Piotr Maziarz, Jian-Xun Wang, Xinyu Bao, and Pingwen Zhang. Physics-informed deep learning for computational physics: A review. *International Journal of Computational Physics*, 2021: 1–35, 2021.

Karol Mikula. Image processing with partial differential equations. *Modern methods in scientific computing and Applications*, pages 283–321, 2002.

Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.

Joaquim Peiró and Spencer Sherwin. Finite difference, finite element and finite volume methods for partial differential equations. *Handbook of Materials Modeling: Methods*, pages 2415–2446, 2005.

Bastian Pentenrieder. Finite element solutions of heat conduction problems in complicated 3d geometries using the multigrid method. *Degree thesis, Technical University of Munich*, 2005.

Pietro Perona, Takahiro Shiota, and Jitendra Malik. Anisotropic diffusion. *Geometry-driven diffusion in computer vision*, pages 73–92, 1994.

William K Pratt. *Introduction to digital image processing*. CRC press, 2013.

Christopher Rackauckas, Yizhi Yao, Ricky Lu, Jun Zhu, Lucy Luo, Matthias Schmid, Ethan Warner, Rohit Supekar, Dylan Skinner, Qiyang Zhou, et al. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

Maziar Raissi. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357: 125–141, 2018.

Maziar Raissi, Paris Perdikaris Yi Wang, and George Em Karniadakis. Hidden fluid mechanics: A navier–stokes informed deep learning framework for assimilating flow visualization data. *Science Advances*, 6(25):eaba4947, 2020.

Mukul Rathi. Demystifying deep learning: Feed-forward neural network. https://mukulrathi.com/demystifying-deep-learning/feed-forward-neural-network/, 2018. Accessed on June 8, 2023.

Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 62:352–364, 2020.

Nadeem Salamat, Malik Muhammad Saad Missen, and VB Surya Prasath. Recent developments in computational color image denoising with pdes to deep learning: a review. *Artificial Intelligence Review*, pages 1–32, 2021.

FY Saptaningtyas and AD Setyarsi. Finite volume method with explicit scheme technique for solving heat equation. In *Journal of Physics: Conference Series*, volume 1097, page 012089. IOP Publishing, 2018.

CB SchÈonlieb. Image processing±variational and pde methods. *Mathematical Tripos Part*, 3: 2014, 2013.

Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

Hyojin Son, Jaewook Jang, Woosuk James Han, and Ha Joong Hwang. Sobolev training for physics informed neural networks. *arXiv preprint arXiv:2101.08932*, 2021.

John Taylor, Wenyi Wang, Biswajit Bala, and Tomasz Bednarz. Optimizing the optimizer for data driven deep neural networks and physics informed neural networks. *arXiv preprint arXiv:2205.07430*, 2022.

Nils Thuerey, Philipp Holl, Matthias Mueller, Philipp Schnell, Florian Trost, and Kiwon Um. Physics-based deep learning. *arXiv preprint arXiv:2109.05237*, 2021.

Juan Gabriel Triana and Luis Alejandro Ferro Alfonso. Finite difference methods in image processing. 2021.

Jihong Wang, Zhi-Qin John Xu, Jiwei Zhang, and Yaoyu Zhang. Implicit bias in understanding deep learning for solving pdes beyond ritz-galerkin method. 2022.

Nan Wang, Yi Shang, Yuzhen Chen, Min Yang, Qingsong Zhang, Yuxuan Liu, and Zhiguo Gui.

Zhu Wang, Hao Zhang, Jie Liu, and George Em Karniadakis. Physics-informed generative adversarial networks for stochastic differential equations. *arXiv preprint arXiv:2010.03204*, 2020.

Joachim Weickert et al. *Anisotropic diffusion in image processing*, volume 1. Teubner Stuttgart, 1998.

Ning Wen, Qichao Liu, and Liang Xiao. Perona-malik diffusion driven cnn for supervised classification of hyperspectral images. In *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*, pages 850–853. IEEE, 2020.

Logan G Wright, Tatsuhiro Onodera, Martin M Stein, Tianyu Wang, Darren T Schachter, Zoey Hu, and Peter L McMahon. Deep physical neural networks trained with backpropagation. *Nature*, 601(7894):549–555, 2022.

Jie Wu, Dongbin Gao, Xifeng Xie, Eric Qian, Yanqing Wu, Junjie Yan, Jianfeng Lu, and Lawrence Carin. Data-driven discovery of governing physical laws. *Science Advances*, 4(6):eaao6459, 2018.

Yu-chen Wu and Jun-wen Feng. Development and application of artificial neural network. *Wireless Personal Communications*, 102:1645–1656, 2018.

Junyuan Xie, Linli Xu, and Enhua Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pages 341–349, 2012.

Yuan Xue, Yong Li, Kai Zhang, and Fuqian Yang. A physics-inspired neural network to solve partial differential equations–application in diffusion-induced stress. *Physical Chemistry Chemical Physics*, 24(13):7937–7949, 2022.

Jianwei Yang, Jinyang Sun, Huafeng Li, and Zongben Xu. Physics-inspired deep learning for imaging inverse problems. *IEEE Transactions on Image Processing*, 28(3):1086–1101, 2019.

Runzhe Yu, Jiawei Liu, and George Em Karniadakis. Physics-informed deep learning for solving forward and inverse problems in quantum mechanics. *Computer Methods in Applied Mechanics and Engineering*, 372:113404, 2020.

Yuval Zelig and Shai Dekel. Numerical methods for pdes over manifolds using spectral physics informed neural networks. *arXiv preprint arXiv:2302.05322*, 2023.

Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

Lan Zhang, Yu Feng Nie, and Zhen Hai Wang. Image de-noising using deep learning. *Applied Mechanics and Materials*, 641, 2014.

Xiaolong Zhang, Wenxuan Huang, Xiao Liu, and George Em Karniadakis. Interpretable deep learning for accurate prediction of flow physics. *Journal of Computational Physics*, 415:109461, 2020.

Han Zhu and Nicholas Zabaras. Physics-informed neural networks for the solution of forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 394:56–81, 2019.

Tomasz G Zieli. Introduction to the finite element method. 1992.