The impact of different methods of gradient descent on the spectral bias of physics-informed neural networks

Alexander van den Arend Schmidt

Supervisors: Dr. Jing Sun[1], Dr. Alexander Heinlein[1], Dr. Tiexing Wang[2]

Examiner: Dr. Hayley Hung[1]

[1]EEMCS, Delft University of Technology, The Netherlands

[2]Shearwater GeoServices, UK

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 26, 2025

Name of the student: Alexander van den Arend Schmidt
Final project course: CSE3000 Research Project
Thesis committee: Dr. Hayley Hung, Dr. Jing Sun, Dr. Alexander Heinlein, Dr. Tiexing Wang

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Physics-Informed Neural Networks (PINNs) are intended to solve complex problems that obey physical rules or laws but have noisy or little data. These problems are encountered in a wide range of fields including for instance bioengineering, fluid mechanics, meta-material design and high-dimensional partial differential equations (PDEs). Whilst PINNs show promising results, they often fail to converge in the presence of higher frequency components; a problem known as the spectral bias. Multiple studies have explored ways to overcome or minimize spectral bias specifically for PINNs. This paper builds on previous studies by investigating the impact of different gradient descent methods on the spectral bias.

1 Introduction

The study of partial differential equations (PDEs) is key to a wide range of scientific fields [1] [2], yet these equations, often consisting of multiple partial derivatives, are challenging to solve numerically. More recently, PDEs have been solved using neural network-based approaches. One type of these approaches, physics-informed neural networks (PINNs), has shown success in solving inverse and forward problems involving PDEs [3] [4].

However, for PDEs with significant discrepancy or high-frequency components, PINNs fail to converge sufficiently [5] [6]. This problem is known as spectral bias. Several new methods and approaches have been researched to mitigate the effect of spectral bias of PINNs [6] [7].

The effect that different methods of gradient descent have on spectral bias for PINNs is not yet fully understood. This paper explores the ability of different gradient descent methods to mitigate the spectral bias seen in PINNs that simulate PDEs. By choosing multiple specific PDEs with varying frequency this paper aims to show how the different manners of gradient descent overcome or fail to overcome spectral bias for PINNs.

To be able to analyze the dynamics of the training of the PINN, the relatively new neural tangent kernel theory will be used. The neural tangent kernel explores the connections of deep neural networks and the corresponding kernel regression methods which provide insights into the training and convergence of deep learning models. More specifically, it has been proven that, at initialization, fully connected networks are equivalent to Gaussian processes in the infinite-width limit [8]. Previous work using the neural tangent kernel for PINNs has been performed [5] providing an insightful analysis of the training dynamics and also concluding the presence of spectral bias by using the eigenvalues of the neural tangent kernel matrix of a PINN.

This paper is organized as follows. In Section 2 background information is provided and the most significant research decisions are explained. In Section 3 the structure of the performed experiments is given and the corresponding results are shown. The results are then discussed and compared in Section 4. Finally, in Section 5 the conclusions and suggestions for future research are given.

2 Methodology

In this section a general explanation of PINNs is given (Section 2.1). Subsequently, this section highlights the research decisions to investigate the effect of gradient descent methods on the spectral bias of PINNs for PDEs:

1. Firstly, the problems of spectral bias of PINNs are highlighted in Section 2.2 to discover how to measure this.
2. In Section 2.3 the gradient descent methods are chosen based on their expected strengths and weaknesses regarding a PINN loss landscape.
3. In Section 2.4 PDEs are chosen specifically so that their frequency can be varied while keeping other parameters constant and still having an analytical solution for each frequency.
4. Finally, combining Sections 2.2, 2.3 and 2.4, for all methods of gradient descent, and for all PDEs, the effect of increasing the frequency of the PDE on the convergence will be explored. The way this is done in practice is explained in Chapter 3.

2.1 Physics-informed neural networks (PINNs)

This section describes what a PINN is and how it can be used to solve a PDE. A PINN is a type of deep neural network that uses known (physics) rules or conditions about a system compensate for a lack of data or improve the convergence. Therefore, the main difference between a deep neural network and a PINN is in the structure of the loss function. Where a classic deep neural network uses known labeled data to calculate the current loss of a neural network, a PINN uses rules and calculates how closely the neural network adheres to the rules. To describe how a PINN works, consider the example of the general 1D Wave PDE below in Equation 1 [9]:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{1}$$

This PDE describes a wide set of general waves. This equation describes that the second order derivative of a function (u) with respect to the time (t) variable should be equal to the second order derivative of a function (u) with respect to the position (x) variable multiplied by c^2. In this equation c is the velocity of the wave. For the example below, $c = 1$ is used. One possible analytical solution is shown below in Equation 2 [10]:

$$u(t, x) = \sin(5\pi x)\cos(5\pi t) + 2\sin(7\pi x)\cos(7\pi t) \tag{2}$$

Next to the main equation, to make sure that this unique solution is described, a set of boundary or initial conditions are used. For this example, the following rules are used.
Initial Conditions:

$$u(0, x) = \sin(5\pi x) + 2\sin(7\pi x) \qquad x \in [0, 1] \tag{3}$$

$$\frac{\partial u}{\partial t}(0, x) = 0 \qquad x \in [0, 1] \tag{4}$$

Boundary Conditions:

$$u(t,1) = u(t,0) = 0 \qquad t \in [0,1] \tag{5}$$

There are now four rules describing the function, namely the 1D Wave PDE (Equation 1), the two initial conditions (Equations 3 and 4), and the two boundary conditions (Equation 5). All of these five rules must be met to correctly predict the actual wave function. Therefore, the loss function of a PINN must include all these rules.

$$\mathcal{L}(\theta) = \mathcal{L}_b(\theta) + \mathcal{L}_i(\theta) + \mathcal{L}_r(\theta) \tag{6}$$

In Equation 6 above, it is shown that the loss function consists of the boundary loss ($\mathcal{L}_b(\theta)$), the initial loss ($\mathcal{L}_i(\theta)$), and the residual loss ($\mathcal{L}_r(\theta)$) respectively. The variable θ in Equation 6 describes the set of parameters and weights used in the PINN. Using only these rules (shown in Equations 1, 3, 4, 5) describing the wave function, a PINN can model the whole wave equation without ever needing actual labeled data points from the equation (next to the initial and boundary conditions).

2.2 Spectral Bias for Physics-informed neural networks (PINNs)

For this section, the problem of spectral bias for PINNs is shown and explained. Spectral bias, also known as the frequency principle describes the way that (deep) neural networks fit a target function by approaching it from lower frequency predictions to higher frequency predictions [11]. The neural network can get stuck on a predicted solution that has a lower frequency than the actual solution. This leads to the neural network not converging sufficiently.

For PINNs, specifically, spectral bias has been shown to be strongly present [5] [6]. The problem of spectral bias is so apparent for PINNs for two main reasons. Firstly, PINNs are often used to model highly nonlinear formulas such as PDEs that may not have an analytical solution [12]. Because the actual solutions to PDEs usually have high frequency and a lot of discrepancy, they are harder to simulate.

The other reason PINNs are more affected by spectral bias than normal neural networks is due to the difference in loss function. As shown in Equation 6 the loss function of a PINN consists of multiple components. Because each component describes a different rule that can be satisfied by multiple functions, the different components of the loss function do not always cooperate and sometimes even counteract each other, resulting in a sub-optimal solution. To show an example of this, the 1D Wave PDE also described in Section 2.1 will be used.

The wave equation is described by 4 equations. Main wave equation shown in Equation 1. The initial conditions shown in Equations 3 and 4. The boundary conditions shown in Equation 5. Simply training a PINN on these four rules/conditions will result in the outcome seen in the Figures 1 and 2:

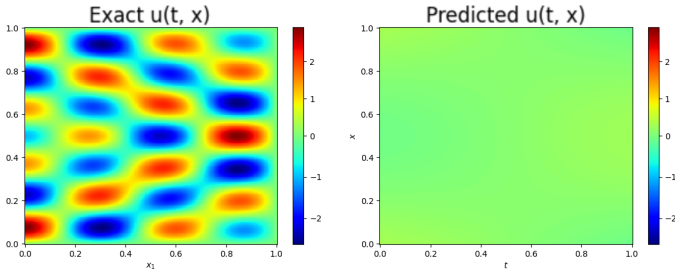


Figure 1: *1D Wave PDE:* PINN trained on all rules and conditions equally.

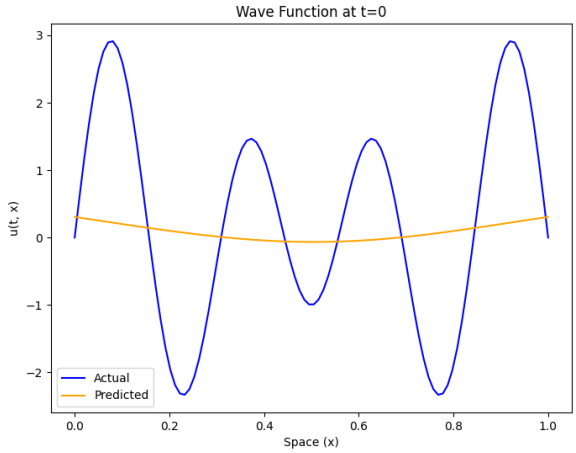


Figure 2: *1D Wave PDE:* Function at t = 0 for PINN trained on all rules and conditions equally.

The Figure 1 shows the actual solution on the left and the predicted solution on the right. The range of both the x and y axes is [0, 1]. In Figure 2 the function is shown at time 0 (t=0). The blue line represents the actual solution, and the orange line is the predicted solution. As can be seen in Figures 1 and 2, the predicted solution stays very close to 0 instead of converging to the actual solution. This can be explained because a function such as the one below, where all values are 0, satisfies both the main wave equation, one of the initial conditions, and the boundary condition.

$$u(t,x) = 0 \qquad x \in [0,1] \;\; t \in [0,1] \tag{7}$$

Only one of the initial conditions is not satisfied but because all the other rules are satisfied the different components of the loss function will counteract each other and the PINN will get stuck on a simple low frequency solution. By simply increasing the amount by which the initial condition affects the total loss function, the predicted solution is already much better. This can be done by multiplying the loss of the initial condition by a value (for Figures 3 and 4 this value is 9). The Figure 3 shows the actual solution on the left and the predicted solution on the right. The range of both the x and y axes is [0, 1].

By comparing the predicted solution in Figure 1 with the predicted solution in Figure 3 it can be seen that Figure 3

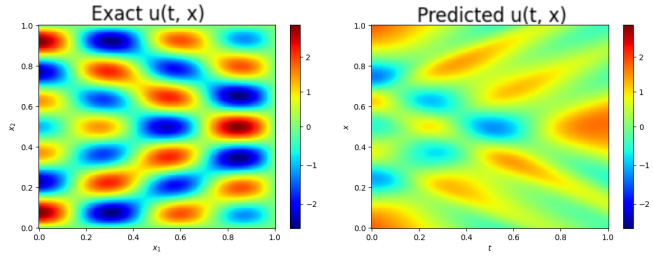


Figure 3: *1D Wave PDE:* PINN trained on all rules and conditions, the initial condition loss weighs 9 times as much as other loss components in the loss function.

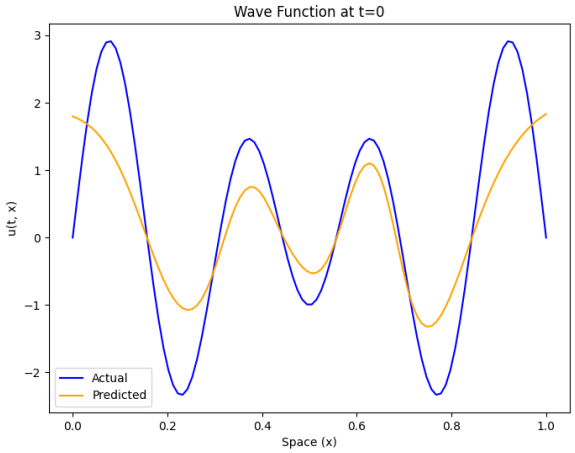


Figure 4: *1D Wave PDE:* Function at t = 0 for PINN trained on all rules and conditions, the initial condition loss weighs 9 times as much as other loss components in the loss function.

provides a prediction much closer to the actual solution. This can also be concluded by looking at Figure 2 and Figure 4, the predicted solution in Figure 4 follows the actual solution much better than in Figure 2.

2.3 Gradient descent methods for PINNs

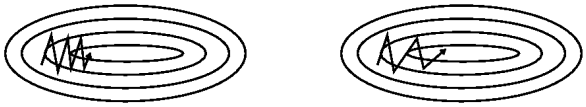
In this section, the gradient descent methods that have been experimented with are explained. For both understanding the initial choice of gradient descent methods and understanding comparisons between different methods, a general knowledge of gradient decent methods is necessary. In this first section, a general overview of gradient descent methods is given. In the next subsection, the choices of gradient descent methods used in this paper are explained.

Overview of gradient descent methods

There are many different gradient descent methods and optimizers. In the following subsections, the most important gradient descent methods are outlined, and a short explanation of their strengths and weaknesses is given. This overview made use of paper [13] and shows the most relevant sections specific for this paper. The variables that are used in Equations 8, 9, 10 and 11 are described below. Firstly, θ describes the parameters and weights of the neural network. v_t is the current update step and v_{t-1} is the previous update step. $\nabla_\theta J(\theta)$ is the gradient of the loss function with respect to θ. Lastly, γ describes the amount of momentum (usually 0.9) and η describes the current learning rate.

Momentum

Normal SGD converges very slowly when navigating ravines. Ravines are places in the loss landscape where the surface is much more steep in one dimension than it is in another, leading SGD to bounce side to side within the ravine instead of going in the direction of the minimum. These ravines are common around local optima. This is visualized in 5a.



(a) SGD without momentum (b) SGD with momentum

Figure 5: Source: Genevieve B. Orr

Momentum attempts to diminish this problem by adding a fraction of the previous gradient descent step to the current step as can be seen in the gradient descent with momentum formula below in Equation 8.

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \\ \theta &= \theta - v_t \end{aligned} \tag{8}$$

Nesterov Accelerated Gradient

Nesterov is strongly related to momentum. When using SGDM, it is possible to overshoot a minimum due to the component of the previous step having a large impact on the current step. Nesterov aims to solve this problem by decreasing the steps when nearing the (local) minimum.

The momentum term $\gamma\ v_{t-1}$ (a fraction of the previous step) is used in the current update. Therefore, updating using only $\gamma\ v_{t-1}$ gives us an approximation of the next position of the parameters. Using this rough approximation of the future position, it is possible to look ahead by calculating the gradient at the approximated future position of our parameters. The formula used is shown below in Equation 9.

$$\begin{aligned} v_t &= \gamma\ v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1}) \\ \theta &= \theta - v_t \end{aligned} \tag{9}$$

Adagrad

Both Momentum and Nesterov adapt their updates for each parameter equally. Adagrad on the other hand, adapts the learning rate to the parameters individually, performing larger updates for infrequent parameters and smaller updates for frequent parameters. A downfall of this method is that the learning rates can become infinitesimally small, halting convergence.

Adam

Adaptive Moment Estimation (ADAM) uses momentum as well as individually updating parameters. m_t and v_t in Equation 10 are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients. In Equation 10 β_1 and β_2 are decay rates and the authors of Adam propose the use of the values 0.9 for β_1, 0.999 for β_2.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{10}$$

These values m_t and v_t are then used to update the parameters using Equation 11. In Equation 11 the value of 10^{-8} is recommended for ϵ by the authors.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{11}$$

Choosing gradient descent methods for PINNs
After having seen an overview of gradient descent methods, the choices of which gradient descent methods to experiment with will be explained. In previous works such as the papers [7] [14] covering gradient descent for PINNs, the ADAM optimizer is shown to be strongly outperforming normal SGD. The paper [14] explains and shows that the PINN loss landscape has a higher density of local optima compared to deep neural networks. Using these findings, the choices of which gradient descent methods to experiment with were made.

Types of gradient descent:

1. Normal Stochastic Gradient Descent (SGD)
2. Stochastic Gradient Descent with Momentum (SGDM)
3. Nesterov Accelerated Gradient Descent (Nesterov)
4. Adagrad gradient descent (Adagrad)
5. ADAM gradient descent (ADAM)

SGD is chosen to be able to compare the other gradient descent methods fairly. Because the loss landscape of PINNs is known to have a large density of local minima, SGDM is selected because momentum performs well close to local optima [13]. Nesterov is often seen as an addition to SGDM. To see if Nesterov outperforms SGDM, Nesterov is chosen. Further, Adagrad is chosen to analyze whether individually tuning the parameters of a PINN results in better convergence. Lastly, ADAM is selected because it has shown promising results for PINNs in previous works [15], and because it combines previously described methods (momentum and individual parameter tuning).

2.4 PDEs for measuring Spectral Bias

This section explains what PDEs are used in this paper and why they were chosen. To successfully measure the degree of spectral bias of a PINN model for a type of gradient descent method, it must be possible to change only the frequency of the PDE while all other variables remain constant. This strongly narrows the set of possible PDEs down. Further, to be able to compute the L2 error after training, between the prediction of a PINN and the actual solution, a set of analytical solutions with varying frequency is necessary. As described in Section 2.2 the loss function of a PINN strongly affects its ability to converge. Therefore, two PDEs are investigated, one with a more complex loss function (1D Wave PDE) and one with a simpler loss function (1D Poisson PDE), with respectively 4 and 2 components in the loss function.

These limitations resulted in the choice of the following two PDEs:

Types of PDEs:

1. 1D Wave Equation (Equation 1)
2. 1D Poisson Equation (Equation 14)

3 Experimental Setup and Results

In this section, the experiments that have been performed are explained and the results that have been found are shown. To show the effects that different manners of gradient descent have on the spectral bias for physics-informed neural network, this paper looks at five methods of gradient descent, each on two different PDEs. To measure the degree of spectral bias, certain values within the PDE will be altered to change the frequency of the PDE.

All experiments were performed in Python using TensorFlow. Each experiment was performed using a sufficiently large architecture and optimized hyperparameters specific for the PDE. The architecture for the 1D Wave PDE consisted of 3 hidden layers of 500 nodes. For the 1D Poisson PDE a single hidden layer of 2048 nodes was used. Hyper parameters such as learning rate and activation functions were optimized using previous works [15] [16] and by testing. In addition, for my experiments, the lambda values in the loss function of the PINNs were optimized only at the start of training. To do this, the neural tangent kernel of the different loss components was used [5]. Throughout the training of the PINN however, the lambdas were kept constant.

Firstly, for all combinations of gradient descent and PDE the PINN will be trained and the results will be analyzed and compared to show which type of gradient descent provides the best solution.

This section is split into two subsections, firstly experiments are done for the 1D Wave PDE. After that, experiments will be performed for the 1D Poisson PDE.

3.1 1D Wave PDE

In this subsection, all types of gradient descent are tested on 2 different types of the 1D wave PDE with varying frequency and the results are shown as heat maps and values. No other variables are altered. First the experiment is explained then the result are shown.

Experiment setup
To vary the frequency of the 1D Wave PDE the value c, in the main wave equation shown in Equation 1, is altered.
The variable c in the 1D Wave PDE corresponds to the speed of the wave. Using the formula for the speed of a wave given below in Equation 12, because the wavelength (λ) is constant, increasing the speed (c) by a factor of two will

increase the frequency (f) by a factor of two.

$$v = f * \lambda \tag{12}$$

Using the analytical solution below in Equation 13 which also has the variable c in it, the frequency of the PDE can now be varied easily. For c the values 0.5 and 1 are chosen to show a clear difference between the gradient descent methods while also showing the decrease in accuracy as the frequency increases.

$$u(t,x) = \sin(5\pi x)\cos(5c\pi t) + 2\sin(7\pi x)\cos(7c\pi t) \tag{13}$$

Results

Table 1: Heat-maps created using different methods of gradient descent when training a PINN on two versions of the 1D Wave PDE, versus the actual solution (top).

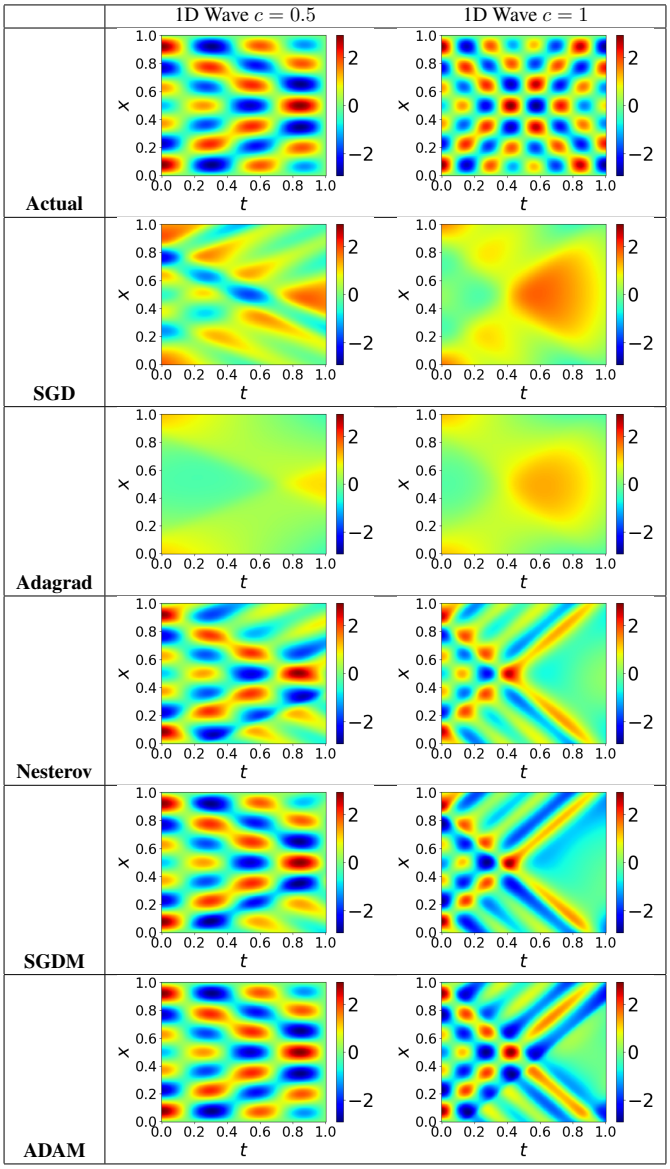


Table 1 shows heat-maps of the predicted functions of the PINN model after training for the corresponding problem and gradient descent method. In Table 1, the value of c that has been used in the 1D Wave PDE is shown in the column heading. The type of gradient descent is shown on the left of the table, with the top one (Actual) being the actual analytical solution. For each heat-map, the domain is the same, that is, the x and y axes both have the range [0, 1]. In Table 2 the L2 errors corresponding to the difference between the predicted solution and the analytical solution are also given as values.

Table 2: L2 errors between the predicted graph and the actual solution for the 1D Wave PDE.

	1D Wave $c = 0.5$	1D Wave $c = 1$
SGD	0.8249	1.2652
Adagrad	1.0301	1.1740
Nesterov	0.3592	0.7204
SGDM	0.1406	0.7024
ADAM	0.0034	0.5147

3.2 1D Poisson PDE

For the 1D Poisson PDE two experiments are performed. Firstly, all 5 types of gradient descent are tested on 2 different types of the 1D Poisson PDE. For the other experiment, for each type of gradient descent, the frequency of the 1D Poisson PDE is increased in steps, and the results are stored. For each method of gradient descent, the learning rate is optimized, but all other hyperparameters are kept constant. In the first subsection, the 1D Poisson PDE is described and the experiments are explained. The final subsection will show the results of the experiments as a table of the L2 errors and a graph showing how the loss changes when the frequency is increased.

Experiment setup

In this section the 1D Poisson PDE is described and the experiments are explained. The general 1D Poisson PDE is shown in Equation 14. The Poisson equation is known as a general form of the well-known Laplace equation. Again for this PDE it is necessary to find a set of analytical solutions that only vary in frequency. Therefore, the equation shown in Equation 15 has been chosen.

$$\frac{\partial^2 u}{\partial x^2} = f(x) \tag{14}$$

$$\frac{\partial^2 u}{\partial x^2} = -(\pi a)^2 \sin(\pi a x) \tag{15}$$

This gives the set of analytical solutions shown in Equation 16. The variable a in Equation 15 and Equation 16 now describes the frequency of the analytical solution. As a result, the value a can now be varied in the experiments to change the frequency of the PDE.

$$\sin(\pi a x) \tag{16}$$

For the first experiment, the values 10 and 20 have been chosen for the variable a. For the 1D Poisson equations for both values of a, all gradient descent methods are tested and the L2 errors are stored. For the second experiment, the frequency (a) of the 1D Poisson PDE is increased from 10 to 40 in steps of two. For this experiment, the L2 residual loss is determined three times for each combination of frequency and gradient descent method, and the average is stored.

Results

In Table 3 the L2 errors are given between the predicted solution and the analytical solution. The predicted solution is the solution obtained by training a PINN on the 1D Poisson PDE shown in the header of the columns with the different gradient descent methods shown on the rows.

Table 3: L2 errors between the predicted graph and the actual solution for the 1D Poisson PDE.

	1D Poisson $a = 10$	1D Poisson $a = 20$
SGD	0.0199	1.1696
Adagrad	0.0050	2.1572
Nesterov	0.0030	0.0335
SGDM	0.0007	0.0050
ADAM	0.0002	0.0024

In Figure 6 a graph is given showing the L2 residual loss for different methods of gradient descent for different frequencies of the 1D Poisson PDE. This graph uses the L2 residual loss instead of the L2 error between the predicted solution and the actual solution. This was chosen to more accurately assess how well the different methods of gradient descent performed. When the PINN does not converge to the correct solution the L2 residual loss is 1, the actual L2 error however becomes extremely large and inconsistent, making it a bad measure of accuracy when a model does not converge.

The frequency in this experiment is shown from (a = 10) up to (a = 40) using steps of two. In this graph, a L2 residual loss of 1 means that the PINN did not converge to the actual solution. A L2 residual loss close to 0 means that a good solution was acquired.

4 Discussion

In line with expectations [5] [6], Tables 1, 2 and 3 and Figure 6 show that spectral bias is present for PINNs even for simple partial differential equations such as the 1D Wave PDE and the 1D Poisson PDE.

Firstly, analyzing the results of the experiments on the 1D Wave PDE in Tables 1 and 2, it is apparent that the 1D Wave PDE with higher frequency (c = 1) is predicted less accurately than the 1D Wave PDE with lower frequency (c = 0.5) for all gradient descent methods. Further, in Table 1 it is shown that even for the wave equation with lower frequency (c = 0.5) Adagrad does not converge properly. This is also confirmed in Table 2. For the higher frequency (c = 1) none of the gradient descent methods achieves a visually correct solution (Table 1). Still, gradient descent methods that use momentum

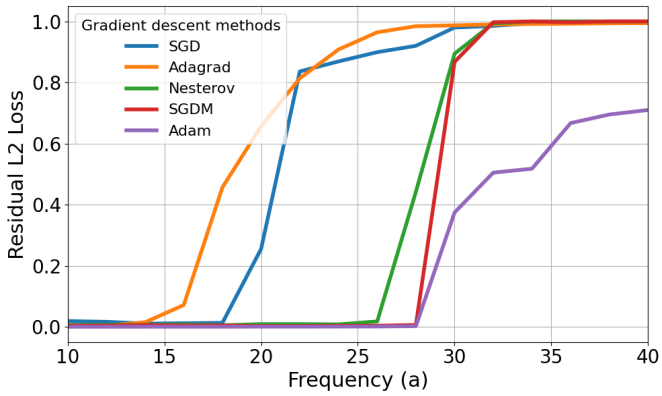


Figure 6: *Residual L2 losses for different gradient descent methods and varying frequency of the 1D Poisson PDE.*

achieve a (visually) better solution than those that do not. Table 2 also shows that the gradient descent methods using momentum outperform those that do not. Of the methods utilizing momentum, Nesterov performs worst and ADAM results in the best solution for both frequencies of the wave equation (Table 2). For the wave equation with lower frequency (c = 0.5), ADAM is clearly the best, with a L2 error which is 41 times smaller then SGDM ($\frac{0.1406}{0.0034}$) as shown in Table 2. At the higher frequency (c = 1) wave equation ADAM still outperforms SGDM but now only has an L2 error that is 1.36 times smaller than the L2 error for SGDM ($\frac{0.7024}{0.5147}$) (Table 2).

Analyzing the results of Table 3 for the experiments with the 1D Poisson PDE; the Poisson equation with the lower frequency (a= 10) is resolved by all gradient descent methods, while at higher frequency (a=20) only the methods with momentum converge. Within the methods with momentum, Nesterov performs worst, and ADAM performs best. Figure 6 shows: an L2 residual loss of 0 means that the PINN converged sufficiently, an L2 residual loss of 1 means that the PINN did not converge to the correct solution, and the frequency (a) at which the convergence fails is the point at which the line increases from 0 to 1. It can be seen that this point (frequency, a) differs per gradient descent method. The first methods to fail are the methods without momentum (SGD and Adagrad), and the gradient descent methods that use momentum (Nesterov, SGDM and ADAM) perform up to higher frequencies. ADAM performs up to the highest frequency for the 1D Poisson PDE. Furthermore, Figure 6 shows that methods that use only momentum (SGDM and Nesterov) fail fairly abruptly. In contrast, those methods that use individual parameter tuning (Adagrad and ADAM) fail more gradually and continue to provide an approximation of diminishing quality.

Comparing the different methods of gradient descent looking at the results of the experiments for both the 1D Wave PDE and the 1D Poisson PDE (Tables 1, 2 and 3 and Figure 6) some common results can be noted. First, all experiments show that individual parameter tuning (Adagrad) does not improve over normal SGD. The methods deploying momentum (Nesterov, SGDM and ADAM) outperform those that do not (SGD, Adagrad). The improvement of using a momentum

component in the gradient descent method was expected as the loss landscape of PINNs has a relatively large amount of local minima, and gradient descent methods with momentum are known to deal better with local minima [14] [13]. Nesterov, which is often seen as an improvement over SGDM, performed worse than SGDM for both PDEs. Apparently, its ability to not overshoot minima is a disadvantage given the many local minima in the loss landscape of PINNs. Of all the gradient descent methods, ADAM produced the best results and was the last to fail whilst increasing the frequency of the PDEs. Still, none of the gradient descent methods was able to mitigate the spectral bias all together. As described in Section 2.2 the complexity of a PINN loss function could affect the results. The 1D Wave PDE has a loss function of 4 components, the 1D Poison PDE just two components. Despite the difference in the complexity of the loss function, the relative performance of the gradient descent methods and the observations are the same across the experiments for both PDEs.

5 Conclusions and Future Work

It is concluded that the method of gradient descent has a significant impact on the spectral bias of PINNs. Momentum seems to be the most important component of a gradient descent optimizer when training high-frequency PDEs with PINNs. Methods of gradient descent using a component of momentum outperform other methods of gradient descent. Nesterov Accelerated Gradient results in slightly worse outcomes compared to normal SGDM, and ADAM outperforms both. In addition, ADAM's performance deteriorates more gradually than the other tested gradient descent methods using momentum (Nesterov and SGDM), still providing an rough approximation of the target function at the higher frequency levels rather than complete failure. Overall, it is concluded that ADAM performs best of all tested gradient descent methods and is least affected by spectral bias, although spectral bias cannot be completely mitigated with the evaluated gradient descent methods.

This paper shows results based on two PDEs (1D Wave PDE and the 1D Poisson PDE) and does not cover higher-order PDEs or non-linear PDEs. Investigating more complex PDEs could be a valuable addition to this work. Although it is clear that momentum is important to capture the higher frequency components in PINNs, there are differences in the performance of different implementations of momentum. It would be interesting to understand the underlying interactions that may allow for further optimization of these momentum gradient decent methods for PINNs. Also, this paper focused on common gradient descent methods. Researching more complex gradient descent methods such as quasi-Newtonian gradient descent methods might provide new insights.

6 Responsible Research

To ensure that research is done responsibly, all results should be reproducible and data should be publicly available. In addition, all conclusions drawn should be fully supported by the results provided and the referenced papers.

To make this research reproducible, all optimizations and research decisions used in the research process have been stated in this paper. Also all hyperparameters used for training the PINNs in this paper are also explained. Furthermore, the code used for the research and for the experiments is available online at (https://cse3000-research-project.github.io/) and is documented to make it more understandable and easy to use. Due to the weight initialization and the data sampling used for PINNs, the outcomes are not deterministic. Although the PINNs used in this paper are not deterministic, no significantly large differences were discovered between runs. To further ensure reliable results for this paper, all experiments have been performed at least three times and the average results are displayed.

No data was used in the training process because PINNs do not use data but rules and conditions to train. It is possible that for a PDE no analytical solution exists; in that case data would be needed to test the accuracy of the PINN. However, for this paper, both PDEs have analytical solutions, and for both PDEs all rules and analytical solutions are provided in the paper and in the code.

Lastly, the conclusions in this paper are exclusively supported by the results represented in this paper and the referenced papers.

References

[1] Burger Martin, Caffarelli Luis, and Markowich Peter A. Partial differential equation models in the socio-economic sciences. *Phil. Trans. R. Soc.*, 2014.

[2] Antontsev S., Diaz J., and Shmarev S. Energy methods for free boundary problems: Applications to nonlinear pdes and fluid mechanics. *ASME.*, 48, 2002.

[3] Yuyao Chen, Lu Lu, George Em Karniadakis, and Luca Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics express*, 28(8):11618–11633, 2020.

[4] Jie Zhang, Yihui Zhao, Fergus Shone, Zhenhong Li, Alejandro F Frangi, Sheng Quan Xie, and Zhi-Qiang Zhang. Physics-informed deep learning for musculoskeletal modeling: Predicting muscle forces and joint kinematics from surface emg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 31:484–493, 2022.

[5] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449, 2022.

[6] Xintao Chai, Wenjun Cao, and Jianhui Li. Overcoming the spectral bias problem of physics-informed neural networks in solving the frequency-domain acoustic wave equation. *IEEE*, 2024.

[7] Ghazal Farhani, Alexander Kazachek, and Boyu Wang. Momentum diminishes the effect of spectral bias in physics-informed neural networks. *Cornell University*, 2022.

[8] Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradi-

ent independence, and neural tangent kernel derivation. *arXiv*, 2019.

[9] Harpreet Sethi, Doris Pan, and Pavel Dimitrov. Physics-informed neural networks for acoustic wave propagation. *Cornell University*, 2020.

[10] Joel Feldman. Solution of the wave equation by separation of variables. *UBC Mathematics Department*, 2007.

[11] Yuan Cao, Zhiying Fang, and Yue Wu. Towards understanding the spectral bias of deep learning. *Cornell University*, 2019.

[12] Edward R, Benton, and George W. Platzman. A table of solutions of the one-dimensional burgers equation. *Quarterly of Applied Mathematics*, 30:195–212, 1972.

[13] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv*, 2016.

[14] Pratik Rathore, Weimu Lei, and Zachary Frangella. Challenges in training pinns: A loss landscape perspective. *arXiv*, 2024.

[15] Nima Hosseini Dashtbayaz, G. Farhani, and Boyu Wang. Physics-informed neural networks: Minimizing residual loss with wide networks and effective activations. *International Joint Conference on Artificial Intelligence*, 2024.

[16] Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv*, 2020.