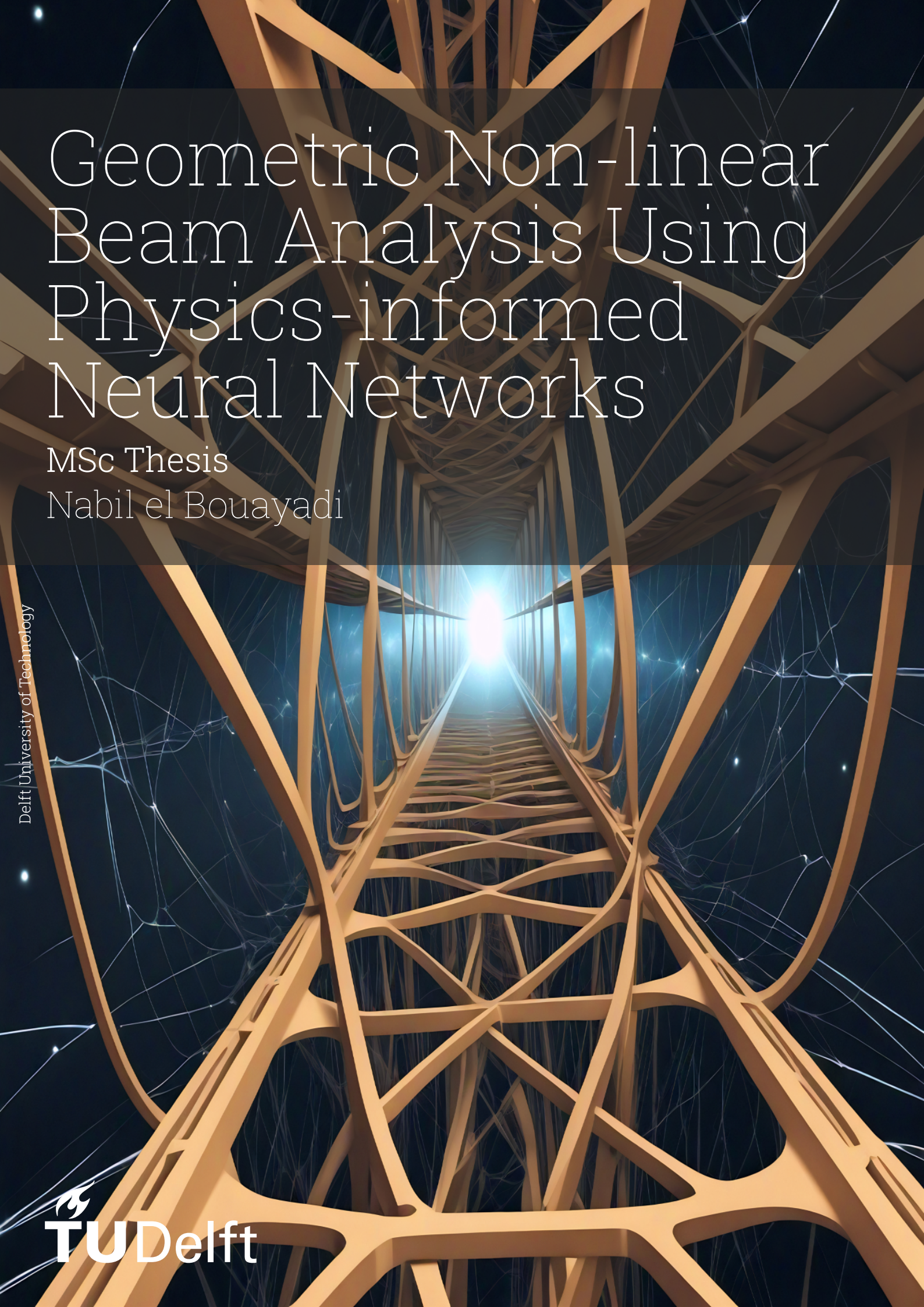
Geometric Non-linear Beam Analysis Using Physics-informed Neural Networks

MSc Thesis

Nabil el Bouayadi



Delft University of Technology

TU Delft

Geometric Non-linear Beam Analysis Using Physics-informed Neural Networks

by

Nabil el Bouayadi

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday December 19, 2024 at 11:30 AM.

Student number:	4598326	
Thesis committee:	Dr. H. Wang,	TU Delft, supervisor
	Dr. A.A. Núñez Vicencio,	TU Delft, supervisor
	Dr. E. Lourens,	TU Delft, chair

Cover:	Generated using openart.ai
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

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

TU Delft

Abstract

The finite element method has proven itself to be an effective method of performing both linear and non-linear structural analyses. This numerical method, however, has several limitations, of one which is that the discretization of complex geometry requires human effort. The rise of the application of machine learning models has opened new possibilities of approaching challenges within scientific fields. The neural network, and more specifically, the physics informed neural network, is a promising method and allows for performing analyses without any discretization that is necessary. This type of neural network, which is less of a black box than the standard neural network, utilizes the concept of partial differential equations to accurately predict solutions.

The objective of this thesis is to perform a structural analysis of a geometrically non-linear Timoshenko beam using a physics informed neural network. The network is built using a variational principle (the principle of virtual work) and a force residual. Furthermore, two optimization algorithms, the adaptive weight loss algorithm and the adaptive activation function are separately used in conjunction with the model to examine the potential improvements on the convergence rate.

It is found that the geometrically non-linear Timoshenko beam can be accurately modeled (relative error of below 2% with respect to the finite element output) with a physics informed neural network. This accuracy can be achieved with a model possessing a relatively shallow size of four hidden layers containing eight neurons each. The adaptive weight loss algorithm and the adaptive activation algorithm both improve the convergence rate of the model, though they are not necessary to maintain the practicality of the model, as the convergence rate is adequate without these. It is recommended that the hyperbolic tangent function is utilized in conjunction with the Adam optimizer. The adaptive activation function can be incorporated into the model to improve the convergence rate significantly without substantially increasing the computational cost of the model.

Contents

1 Introduction

The analysis of the non-linear behavior of beam-columns is a critical aspect within the field of structural engineering. This is particularly essential for predicting the performance of structures subjected to extreme events such as earthquakes, wind loads, and other dynamic forces, where assumptions of linearity can lead to significant inaccuracies and potentially unsafe designs. Over the years, engineers have relied on the finite element method to conduct non-linear structural analyses.

The finite element method (FEM) is a powerful tool used for performing both linear and non-linear structural analyses. However, one limitation of this method is the fact that the analyzed geometry needs to be discretized (meshed) before any results can be presented. Meshes of poor quality can lead to numerical inaccuracies such as singularities (inaccurate concentrations of stresses), artificial locking and convergence issues which is why proper discretization of the analyzed geometry becomes critical when performing any type of analysis using FEM. Proper discretization quickly becomes a time-consuming endeavor when complex geometries are to be investigated, as generating an accurate mesh is not always a fully automated process for such cases, despite recent advancements (Talebi et al., 2016).

A novel approach built on deep learning has surfaced as a promising substitute, which can perform structural analyses, is a meshless based method and as such does not suffer from the aforementioned pathologies (Raissi et al., 2019). This method, referred to as physics-informed neural networks (PINNs), employs a neural network to approximatively arrive at a solution while enforcing physical principles, often described by partial differential equations (PDEs), as a component of the loss function. This ensures that no training data is necessary to produce accurate results compared to a traditional neural network, while showing additional promise by being a meshless method.

The Euler-Bernoulli beam theory (Bauchau & Craig, 2009) is a popular theory utilized when performing structural analyses of beams and is sufficiently accurate when describing slender[1] beams. It, however, fails to accurately capture the behavior of beams where shear deformation is significant, which holds true for short and thick beams, whereas the Timoshenko beam theory (Timoshenko, 1921) does take shear deformation into account.

Hornik et al. (1982) already established that a feedforward neural network with at least one hidden layer can approximate any continuous function, given that a non-linear activation function is applied. With PINNs, knowledge from physics is leveraged to improve the learning process. Thus, any phenomenon within structural engineering that satisfies this criterium of continuity can be described by PINNs. In this thesis, PINNs will be utilized to perform both linear and geometric non-linear elastic analyses of one-dimensional beams assuming the Euler-Bernoulli and Timoshenko beam theory. The results of these analyses will be verified with the finite element program DIANA. The objective of this thesis is to accurately model the geometric non-linear elastic Timoshenko beam using PINNs.

[1] "slender" is here defined as beams whose length is several times greater than the cross-sectional dimensions, to the point that deformation caused by shear is negligible.

The structure of this thesis is as follows. In chapter 2, non-linearity and its sources within the context of structural analyses will be explained. In chapter 3, the traditional neural network and the physics-informed neural network will be described and compared, followed by the advantages and disadvantages of neural networks. In chapter 4, the methodology that is utilized in this thesis is described. In chapter 5, the linear analyses of both the Euler-Bernoulli and Timoshenko beam will be described accompanied by numerical examples. In chapter 6, the theoretical assumptions for the non-linear beam models will be explained. The structure of the PINN model that is used for solving each beam case will be clarified and the geometric non-linear analyses will be performed. In chapter 7, two stabilizing techniques will be used and compared. In chapter 8, a parameter study on the PINN configuration for the non-linear Timoshenko beam will be conducted. In chapter 9, the role of PINNs within monitoring along with its limitations will be discussed. Lastly, the thesis will be concluded and some recommendations regarding future studies will be made in chapter 10.

2

Structural Analysis and Beam Theories

2.1. Linear and Non-linear Static Analyses

Within structural analysis the distinction is made between two types of models: linear and non-linear models. Linear analyses hold a linear relationship between applied loads and the resulting deformations as depicted in Figure 2.1. Within the context of FEM, this means that the stiffness matrix of the model would remain constant. For non-linear analyses this will not be the case, as there is a non-linear relationship between the loads and deformations resulting in a stiffness matrix which alters during the load application.

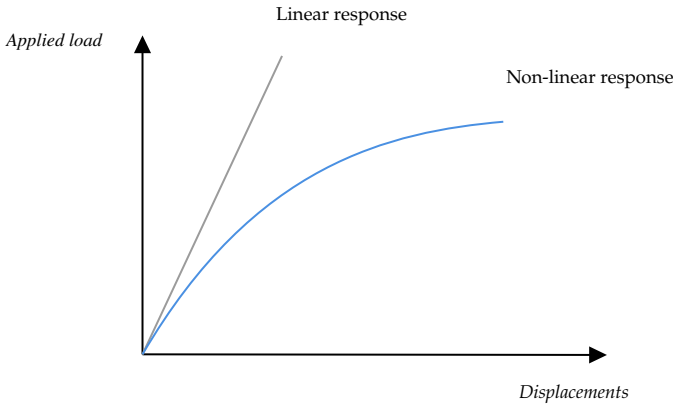


Figure 2.1: An example of response curves of both linear and non-linear analyses.

For linear analyses it is assumed that the structure remains in the elastic domain and that the corresponding displacements and rotations are sufficiently small. For non-linear analyses these do not hold as the structure exhibits either material, geometric or contact non-linearity or any combination of these.

2.2. Sources of Non-linearity

When performing structural analyses on any type of structure the consideration of contributing non-linear effects becomes vital, as these could lead to a significantly different structure response. These non-linear contributions can be time-dependent (such as creep of concrete, where deformation keeps increasing over a period of time under sustained loading) and time-independent (such as plastic

deformation of steel). As such, building codes, such as the Eurocode, incorporate non-linear effects to ensure safety in structural design. These include buckling of steel members and second-order effects which are both (geometrically) non-linear phenomena (Gardner & Nethercot, 2005).

2.2.1. Geometric Non-linearity

Geometric non-linearity becomes applicable when the strain definitions are non-linear (finite strains), such as when the true strain is opted for instead of the engineering strain definition, or when large displacements are exhibited by the structure such that the kinematic relations are no longer linear, despite assuming linear strains (finite displacements). An instance of the latter is when the deformed shape becomes large enough such that the equilibrium equations of this structure can no longer be accurately based on its undeformed shape. The equilibrium equations need to be written with respect to the deformed structural geometry.

2.2.2. Material Non-linearity

Material non-linearity becomes relevant when the mapping between the stresses and strains are non-linear. The material behavior becomes dependent on the strain history. Examples of non-linear stress-strain relationships include hyperelasticity and elastoplasticity as visualized in Figure 2.2.

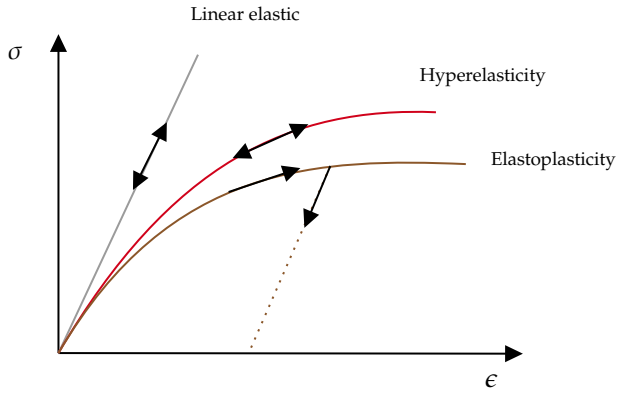


Figure 2.2: Different stress-strain relationships where the stresses σ of a material are plotted against the strain ε. Linear elastic stresses are linearly dependent on the strains and the deformations are reversible. Hyperelastic stresses are non-linearly dependent on the strains and the deformations are reversible. Elastoplastic stresses are non-linearly dependent on the strains, but the deformations are irreversible after unloading.

2.2.3. Contact Non-linearity

Contact non-linearity results from the conditions at the interfaces of interacting bodies, such as friction, separation, and slippage. Since this kind of non-linearity depends on the dynamic nature of the interactions, it presents significant challenges in accurately predicting structural behavior. For instance, friction prevents relative motion across surfaces, leading to intricate stress distributions that change depending on the surface properties and applied force. This complexity is increased by separation and re-contact events, which are frequently seen in cyclic loading scenarios. On the other hand, slippage can result in localized plastic distortion and wear, which compromises the structures' long-term integrity and performance.

Contact non-linearity can be simplified in certain cases by modelling it as material non-linearity, streamlining the analysis while still retaining sufficient accuracy. An example of this is utilizing a smeared crack model when modelling cracked concrete, where a homogenized non-linear material is used to represent the cracks (Weihe et al., 1998).

2.3. Beam Theories

In the following paragraphs, the two beam theories, the Euler-Bernoulli beam theory and the Timoshenko beam theory will be explained. The main characteristics of each theory and the differences between them will be described. Each beam theory relies on its own kinematic definitions which influence how the internal forces of a beam element are described and are independent of the type of structural analysis conducted.

2.3.1. Euler-Bernoulli Beam Theory

The Euler-Bernoulli beam theory is a widely used theory to describe the behavior of one-dimensional beams. It relies on the following kinematic assumptions:

1. The cross-section of the beam remains plane after deformation.
2. The cross-section does not deform in its own plane.
3. The cross-section remains normal to the neutral axis, even after deformation.

These assumptions suffice for isotropic, slender beams with solid cross-sections (Bauchau & Craig, 2009).

The first two assumptions imply that the cross-section can only undergo rigid translations and rotations. The third assumption implies that the beam has an infinite shear stiffness, which indicates that the resulting deformations are underestimated (if there are shear stresses present within the beam). The Euler-Bernoulli beam is visualized in Figure 2.3.

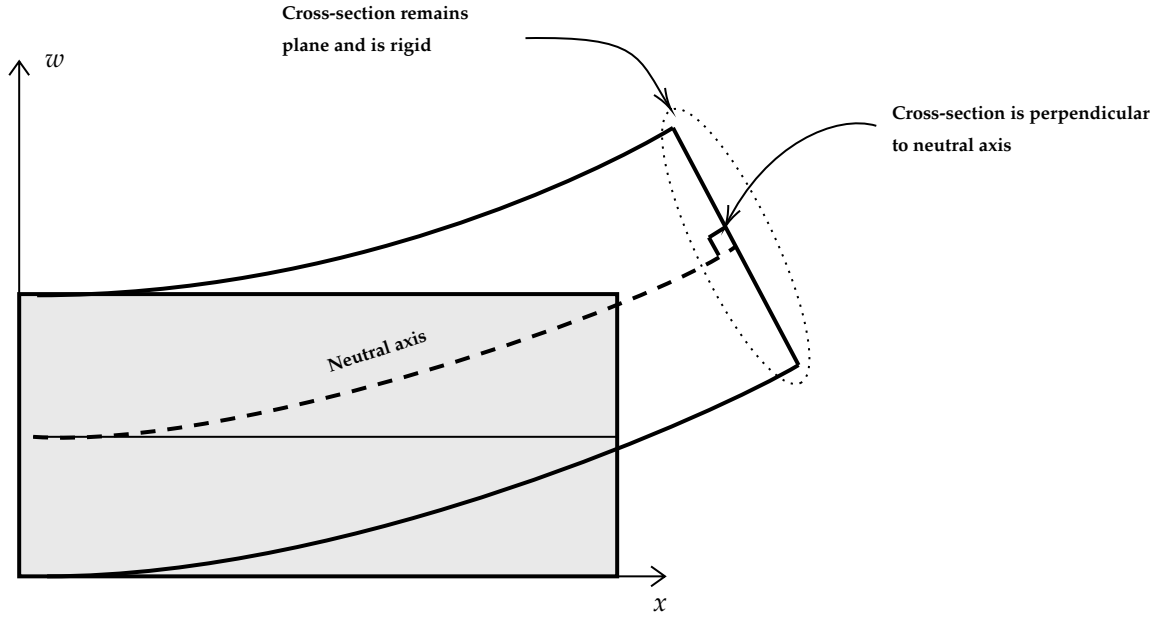


Figure 2.3: The deformed state of an Euler-Bernoulli beam and its kinematic assumptions.

2.3.2. Timoshenko Beam Theory

The Timoshenko beam theory takes shear deformation into account and can be considered as a more comprehensive theory than the Euler-Bernoulli beam theory. Here it is still assumed that the cross-section remains plane, but that it is no longer perpendicular to the neutral axis. The shear strain γ dictates the rotation of the cross-section with respect to the axis that is perpendicular to the neutral axis. This means that now, the rotation of the cross-section is not equal to the first derivative of the deflection, as shown in Figure 2.4.

The aforementioned assumptions are also utilized when it comes to plate theory. The Kirchhoff-Love plate theory can be considered as a two-dimensional extension of the Euler-Bernoulli beam theory as neglection of shear deformation is also assumed here. The Reissner–Mindlin plate theory takes shear deformation into account and can be considered as an extension of the Timoshenko beam theory in two-dimensional space.

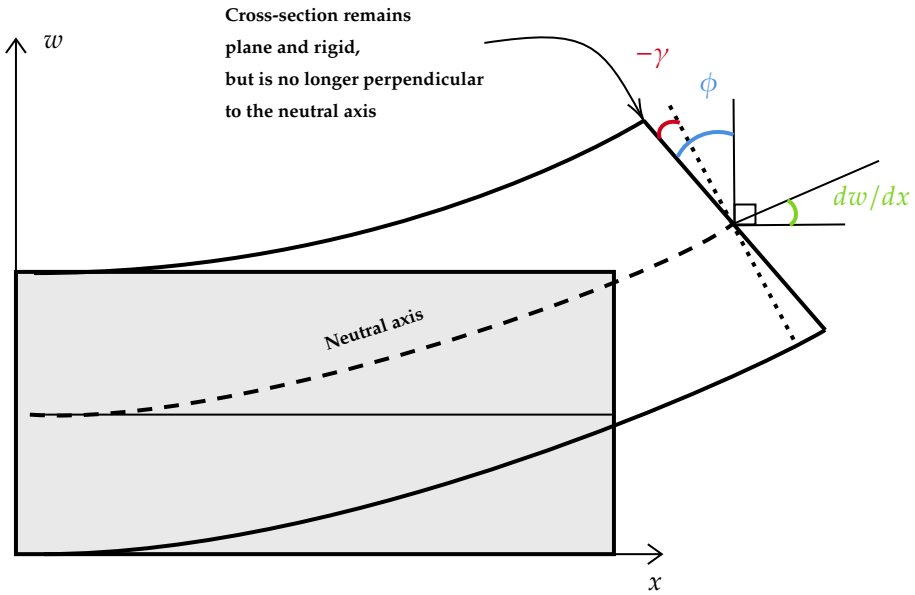


Figure 2.4: The deformed state of a Timoshenko beam. If the shear strain is equal to zero, then the beam becomes identical to the Euler-Bernoulli beam.

3

Physics-Informed Neural Networks

While FEM has proven itself effective in conducting both linear and non-linear structural analyses, it can require manual fine-tuning which can be time consuming when handling complex geometry, as discussed in the introduction. We turn to PINNs to address this challenge, as these are proven to be able to perform structural analyses (Kapoor et al., 2023) while also being a meshless method. First, a brief description of a traditional neural network will be given, followed by the description of the defining characteristics of PINNs which separate it from the traditional neural network. The preference of utilizing PINNs over the traditional network in the context of this study will also become apparent in this chapter.

3.1. Traditional Neural Networks

Neural networks are a type of machine learning model that can learn from data and perform various tasks, such as classification and regression. This paragraph gives a brief description of how a traditional neural network operates.

A neural network is made up of one or more layers, each of which is made up of several neurons. The first layer, known as the input layer, is where the data enters the system. The final layer, known as the output layer, creates the model's output. The layers in between are called hidden layers, which perform intermediate computations. Each neuron in a layer has a connection to every neuron in the following layer, as illustrated in Figure 3.1 The weights of these connections, which express how much effect each neuron has over the neuron in the layer that follows, are numerical values. Each neuron also has a value called a bias, which is added to the weighted sum of the inputs from the previous layer. Every neuron applies an activation function to its input, which establishes the neuron's output. There are various activation function types, such as the sigmoid, tanh and ReLU functions, each with its own characteristics and consequences on the model.

The effectiveness of a neural network must be evaluated to determine how to improve it. This is accomplished using a loss function, which determines the discrepancy between the model's output and the desired (true) output. The goal of the model is to reduce the loss function as much as is attainable. The model must make the necessary weight and bias adjustments in order to minimize the loss function. Back propagation, a technique that determines the gradient of the loss function with respect to each weight and bias in the network, is used to do this. The gradient indicates how each weight and bias should be altered in order to reduce the loss. The model then updates its weights and biases by subtracting a fraction of the gradient, called the learning rate.

In neural networks, optimizers are algorithms that modify the parameters, such as weights and biases, in order to minimize a loss function. This allows the model to learn from data. The loss function gauges how well the model fits the data and how closely the outputs are predicted to be produced. Different techniques, including gradient descent, are used by optimizers to update the parameters. Each

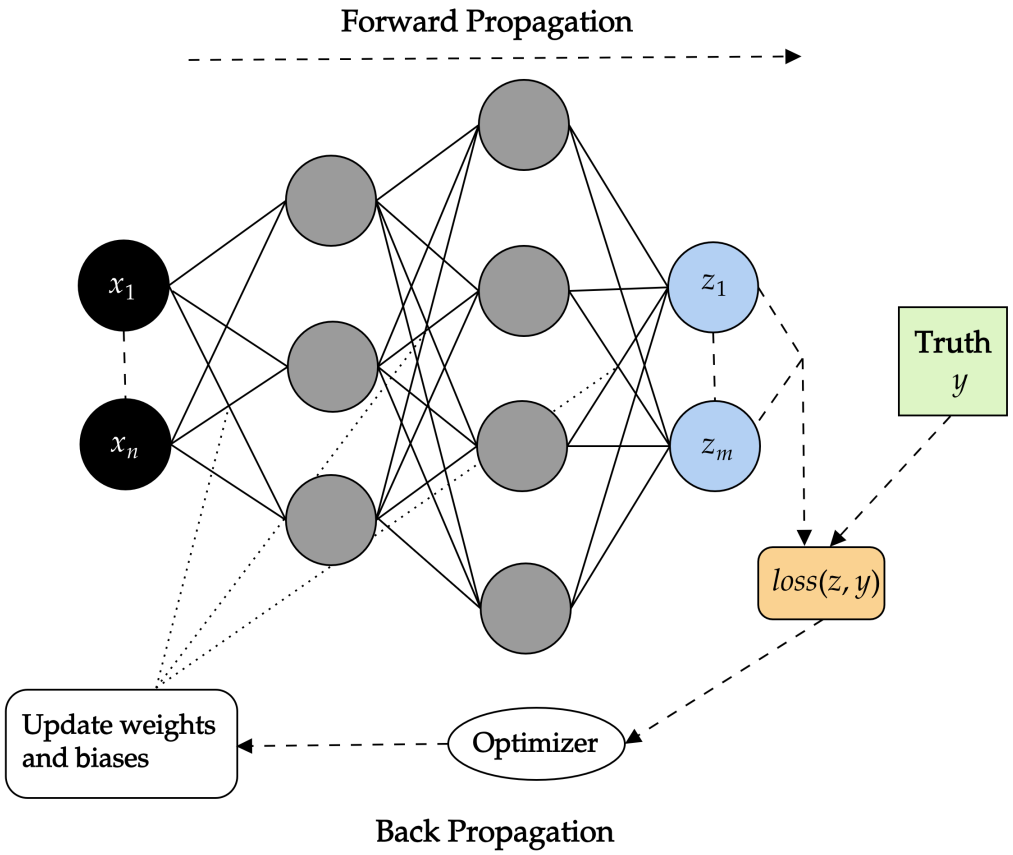


Figure 3.1: A visual representation of a neural network with input (black), hidden (grey), and output (blue) layers. Here the network contains a n amount of inputs and a m amount of neurons as output, where both input and output are self-defined. After conducting a forward propagation, where an output is gathered after the network is fed the input, the loss will be calculated. The loss is a function of both the output and the truth (the exact/desired value), and is commonly the mean squared error of both. Based on the loss, the weights and biases of the network are updated to ensure a more accurate prediction. This update is done using an optimizer such as gradient descent. The process where the weights and biases are updated through an optimizer is called back propagation.

optimizer has its advantages and disadvantages. They may differ in ease of implementation, rate of convergence and computational cost (Desai, 2020). Adaptive Moment Estimation (Adam) and Stochastic Gradient Descent are examples of optimizers.

There are different types of neural networks, such as the convolutional neural network, which is used to handle spatial data, and the physics-informed neural network, which is able to incorporate physical laws.

$$\mathbf{a^{i+1}} = \sigma\left(\mathbf{W^{i+1}} \cdot \mathbf{a^{i}} + \mathbf{b^{i+1}}\right),$$

where $\mathbf{a^i}$ denotes the vector containing the numerical value of each neuron in layer $\mathbf{i}$, $\mathbf{W^i}$ denotes the matrix containing the weights in layer $\mathbf{i}$, and $\mathbf{b^i}$ denotes the vector containing the biases in layer $\mathbf{i}$. σ is an activation function responsible for non-linearity.

Following this, the neural network in Figure 3.1 would be described as:

$$\mathbf{a^1} = \sigma\left(\mathbf{W^1} \cdot \mathbf{a^0} + \mathbf{b^1}\right),$$

$$\mathbf{a^2} = \sigma\left(\mathbf{W^2} \cdot \mathbf{a^1} + \mathbf{b^2}\right),$$

$$\mathbf{a^3 = W^3 \cdot a^2 + b^3}.$$

Here $\mathbf{a^0}$ indicates the input vector $\mathbf{x}$ and $\mathbf{a^3}$ the output vector $\mathbf{z}$. Note that there is no activation function applied for the output layer. Consideration of whether to utilize an activation function for the output layer depends on the context of the problem (Jagtap & Karniadakis, 2023).

3.2. Defining Characteristics of PINNs

PINNs are defined as a variant of a neural network that includes information about physical laws, often described by partial differential equations (Raissi et al., 2019). This provides the network with additional leverage in its learning process. Training now becomes significantly faster, as there is already established prior knowledge with which the network can be guided. This is in contrast to traditional neural networks, which can only rely on data for learning.

The application of PINNs can be especially useful when the underlying physics are understood, but the exact solution is difficult to obtain due to, for example, the complexity of the equations involved. This represents solving a forward problem. PINNs can also be utilized to solve inverse problems, that is, to solve for underlying parameters.

3.2.1. The Loss Function

The loss function, which assesses how effectively a neural network approximates the solution of a PDE and complies with the specified boundary and initial conditions, is one of the fundamental elements of PINNs. The loss function consists of a PDE term, a data term, a boundary term, and an initial term (Kapoor et al., 2023). Each term stands for a distinct element of the physical issue and makes a unique contribution to the total loss. The data term becomes necessary when solving inverse problems.

The data loss term is the mean squared error between the predicted outputs of the neural network and the observed data points. This term ensures that the neural network fits the data that is available. The data term can be written as:

$$\mathcal{L}_{Data} = \frac{1}{N_D} \sum_{i=1}^{N_D} (u_{PINN}(x_i, t_i) - u_{obs}(x_i, t_i))^2,$$

where N_D represents the number of points where the output is known, and $u_{obs}(x_i, t_i)$ represents the known (exact) output at (x_i, t_i). $u_{PINN}(x_i, t_i)$ denotes the predicted outcome of the neural network.

The PDE loss term is described as:

$$\mathcal{L}_{PDE} = \frac{1}{N_{PDE}} \sum_{i=1}^{N_{PDE}} f\,(x_i, t_i)^2,$$

where $f\,(x_i, t_i)$ denotes the equation for a governing physical law.

The boundary loss term is described as:

$$\mathcal{L}_{BC} = \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \left(u_{PINN}\left(x_b^i, t_b^i\right) - u_{BC}\left(x_b^i, t_b^i\right)\right)^2,$$

where $u_{BC}(x_b^i, t_b^i)$ denotes the known function at the boundaries (x_b^i, t_b^i).

Similarly, the initial loss term is described as:

$$\mathcal{L}_{IC} = \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} \left(u_{PINN}\left(x_I^i, t_I^i\right) - u_{IC}\left(x_I^i, t_I^i\right)\right)^2,$$

where $u_{IC}(x_I^i, t_I^i)$ denotes the initial condition function at (x_I^i, t_I^i).

The loss function becomes:

$$\mathcal{L} = \mathcal{L}_{Data} + \lambda_1 \mathcal{L}_{PDE} + \lambda_2 \mathcal{L}_{IC} + \lambda_3 \mathcal{L}_{BC}. \tag{3.1}$$

Here λ_1, λ_2 and λ_3 are weights utilized to balance each term in the loss function. These can be fixed values set at the start or also adapted over time. The loss function for a traditional neural network would only consist of $\mathcal{L}_{Data}$.

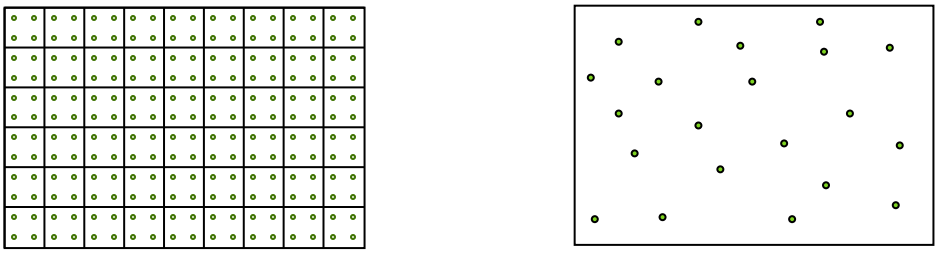


Figure 3.2: An example of how geometry is handled with FEM and PINNs. Within FEM (left), here, the domain of the geometry is discretized into quadrilateral elements with 4 integration points per element before performing the structural analysis and obtaining the output at the nodes and the integration points. For PINNs the calculations are performed on an arbitrarily defined set of points within the domain. The main difference is that with FEM the geometry has to be divided in sub-domains (elements) where with PINNs this is not necessary.

3.2.2. Collocation Points

Within the domain of the geometry specific locations, also called collocation points (Raissi et al., 2019), have to be assigned at which the specified physical laws have to be enforced. These discrete points can have an arbitrary distribution as depicted in Figure 3.2.

Though the collocation points can be arbitrary placed, there are methods devised in generating a collocation point distribution that would enhance the numerical efficiency of the network, such as Latin Hypercube sampling (LHS). The basic notion of LHS, as used in the work of Raissi et al. (2019), is that the domain is split into N intervals, with N being the amount of total collocation points. For each interval there is a collocation point allocated and the exact location within each interval is drawn from an uniform distribution. This sampling method ensures that there is a more comprehensive representation of the domain. There is supporting evidence, however, that utilizing a simple equidistance grid as a sampling method is superior for lower dimensional problems (Leiteritz & Pflüger, 2021).

3.2.3. Optimizers

The goal is to have the PINN model predict the true output as accurately as possible. Optimizers are used to update the model parameters, the weights and biases, to reach this goal. Optimizers update these model parameters based on the output of the loss function. After the PINN model performs a forward propagation, a corresponding value of the loss function will yield. The gradient of the loss function at this value will be calculated. The aim is to take a small step in the direction which will result in a lower value of the loss function and update the model parameters to reach this objective. This is gradient descent and the equation within the context of neural networks is as follows:

$$\mathbf{w_{n+1}} = \mathbf{w_n} - \alpha \nabla \mathcal{L}\left(\mathbf{w_n}\right),$$

where $\mathbf{w_n}$ denotes the weights and biases for iteration n and α denotes the step size or learning rate. Since the loss function takes as input all the weights and biases of the PINN model, the gradient becomes a vector of n entries, with n being the total number of weights and biases the PINN model contains. The loss function of a model with a single weight and no bias terms (which corresponds to a network containing only a single input and output neuron) can therefore be expressed as a graph function with the gradient being the tangent of the function at that point.

Stochastic gradient descent (SGD) is an extension of the traditional gradient descent. Here, the gradient for each iteration is calculated on only a small batch of the total available data. This 'mini-batch' is randomly selected from the total available data, and training is significantly less computationally expensive as a result.

The Adam (Adaptive moment estimation) optimizer is based on SGD. SGD maintains one fixed learning rate for all the weights and biases, in contrary to Adam which maintains an unique learning rate for each model parameter. The algorithm requires little memory and has been shown to be very effective (Kingma & Ba, 2014).

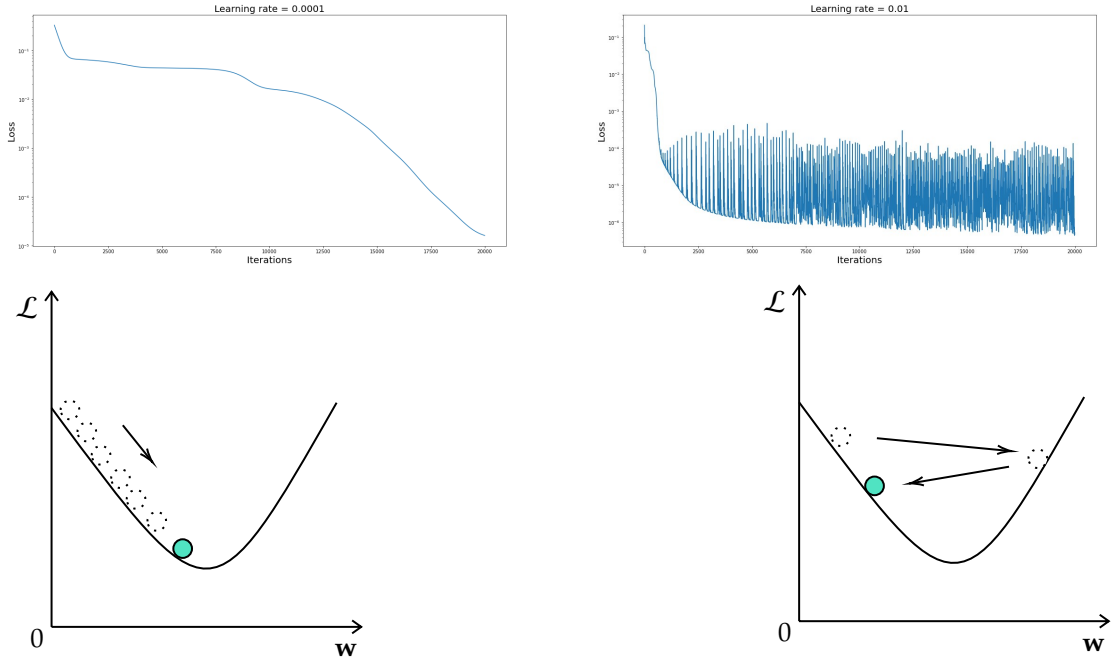


Figure 3.3: Two separate training samples compared of a linear Euler-Bernoulli beam problem that have been carried out, with the left having a learning rate of 0.0001 and the right having a learning rate of 0.01. The loss of the training sample on the right experiences a quicker initial drop, but strongly oscillates around the minimum after a few thousand iterations

The learning rate has a major influence on the overall duration of training. A low learning rate results in an overall steady approach to the minimum, while taking more iterations for the model to find the minimum. A large learning rate results in an overall quicker decrease of the loss function. However, it becomes more likely for the algorithm to 'overshoot' past the minimum. The loss will tend to diverge or oscillate around the global minimum as portrayed in Figure 3.3.

3.2.4. Limitations of PINNs

Despite the potential that PINNs possess, there are several limitations that persist. A common issue is the phenomenon of the network getting stuck in a local minimum, due to the non-convex nature of the loss function. As a result, the model reaches a suboptimal solution. The end state of the neural network is therefore heavily reliant on the initial state of the parameters. The concept is visualized in Figure 3.4.

The exploding gradients is another common issue neural networks face. This occurs when the model parameters are updated during back propagation. During back propagation the gradients are computed for each layer and each layer's gradient depends on the layers that come after it. As the gradients are propagated back through each layer, they are multiplied by the weights associated with that layer and the gradients can thus grow exponentially after each layer if these weights happen to be large. This causes the model parameters to update in excessively large steps, resulting in instability. A neural network with many hidden layers exacerbates this issue. Gradient clipping is a technique that is used to inhibit this issue. It entails setting a specific limit on the gradients to prevent them from becoming too large.

A similar issue to the exploding gradients is the issue of vanishing gradients, where the gradients become so small due to the weights being small, such that the model parameters barely change. If the gradients in the last layer are very small, then the gradients in the former layers end up being negligible, due to these small weights causing the gradients to shrink exponentially after each layer.

Activation functions are directly related to the vanishing gradients issue, with the hyperbolic tangent and sigmoid activation functions shown in Figure 3.5, for instance, being overall more prone to it. The derivatives of these functions attain a maximum value of 1.0 and 0.25 respectively. If the gradient is small in the last layer, then the updates in the former layers become exponentially less. The sigmoid is

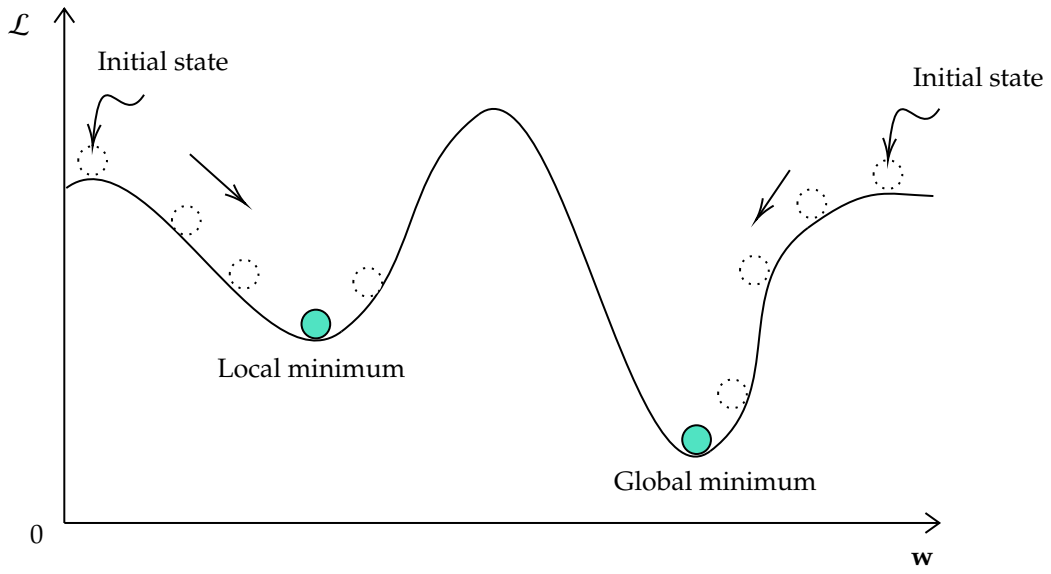
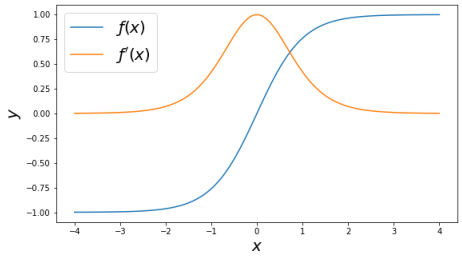
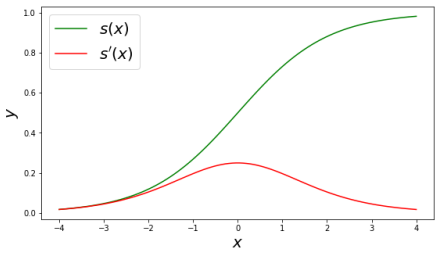


Figure 3.4: An illustrative example of a loss function with the model parameters as input portraying the importance of initialization of the model parameters. Due to the non-convex nature of loss functions, an unfortunate initialization may result in the network getting stuck in a local minimum resulting in suboptimal performance.

even more sensitive than the hyperbolic tangent function, as the maximum the function attains here is less and as such the compounding effect across the layers is more severe.



(a) The hyperbolic tangent function and its derivative.



(b) The sigmoid function and its derivative.

Figure 3.5: Examples of Activation Functions.

3.3. Applications of PINNs within Structural Engineering

PINNs have been utilized to solve problems in numerous fields, including structural engineering and there have been multiple methods proposed to enhance its performance, such as Hybrid PINNs, that utilize convolutional neural networks to solve PDEs, and Bayesian PINNs (Lawal et al., 2022). It is important to note that there are limitations for these new approaches just as there are limitations of conventional PINNs. The training of Hybrid PINNs, for instance, is very sensitive to the initial hyperparameters. A significant amount of the studies involved the application of conventional PINNs, however, showing sufficient promise on its own based on the accuracy and computational efficiency it delivers (Kapoor et al, 2023; Guo & Fang, 2023; Bazmara et al, 2023). The focus of this thesis will be on the conventional PINN.

Within the context of structural engineering, Bazmara et al. (2023) investigated the non-linear buckling behavior of a three-dimensional functionally graded porous, slender beam resting on an elastic foundation (Winkler-Pasternak). PINNs have been utilized to output the critical nonlinear buckling load and the framework has been validated for various boundary conditions. It was concluded that the PINN model accurately predicted the buckling load.

Guo & Fang (2023) constructed a framework involving PINNs for parameter identification (solving inverse problems). The constructed model was used to identify stiffness parameters of frame structures with satisfactory accuracy. It was concluded that the PINN could estimate the stiffness parameters accurately despite being based on a limited dataset.

Chen et al. (2023) proposed a PINN framework able to solve for the geometrically non-linear one-dimensional Euler-Bernoulli beam, The transfer learning technique was applied to enhance computational efficiency. The PINN model produced accurate results, but was overall slower than FEM.

Grossman et al. (2024) investigated the computational expense of PINNs by having it solve different differential equations and compared the computational times to FEM.

It was discovered that the FEM generally was computationally less expensive for solving lower dimensional PDEs. But it was also noted that the PINN model saw no rise in computational cost when transitioning from the 2D to the 3D Poisson equation, possibly indicating that PINNs might be more efficient in high-dimensional settings than FEM and that the 'curse of dimensionality' might be less applicable to it, as supported by Hu et al. (2024).

The main benefit of PINNs, however, is that it is a mesh-free method. The discretization of complex geometries using FEM cannot currently be accurately performed without human interaction, increasing the time demand for performing analyses (Talebi et al., 2016), which is something not considered in the aforementioned studies.

4

Research Questions and Methodology

The aforementioned pathologies FEM suffers from, and the rise of the physics informed neural network with its meshless approach makes it a promising substitute worth investigating. More specifically in this study, the possibility of conducting a geometrically non-linear analysis will be investigated as this is, as mentioned previously, a critical aspect when considering safe structural design and is thus very prevalent in existing building codes (Gardner & Nethercot, 2005). Here, the Timoshenko beam is investigated as it is a more comprehensive beam theory compared to the Euler-Bernoulli beam theory. The main question therefore is:

"*How can physics-informed neural networks (PINNs) solve for the deflections of a geometric non-linear Timoshenko beam?*"

With the corresponding sub-questions being:

1. How can PINNs solve for the beam deflections for a linear beam problem?
2. How is geometric non-linearity introduced in the PINN model?
3. What configuration of the PINN model is necessary for effectively addressing the Timoshenko problem for a one-dimensional beam?
4. How to increase the convergence rate of the proposed model?

The linear beam problems will be investigated first to provide more context and to highlight the difference in approach between the linear and non-linear problems. An effective configuration of the network structure, that is the input neurons, output neurons and amount of hidden layers, will be provided to address the Timoshenko beam. Furthermore, a significant amount of the literature is dedicated to improve the convergence rate of training models. Two algorithms, the adaptive activation function and the adaptive weight loss algorithm, will be investigated.

The PINN model will be built using Python utilizing the library PyTorch. Gathering data is not necessary, as it is a forward problem that is being solved. It is, however, necessary to verify the output of the PINN model. Analytical expressions will be used to verify the linear cases, whereas the non-linear cases will be verified with the Finite Element program DIANA FEA version 10.8. Two-noded beam elements, L7BEN, with two degrees of freedom, the vertical and horizontal translations, have been utilized for discretizing the beam model. This element type will be used for the non-linear Euler-Bernoulli FEM analysis as there is no shear deformation assumed for this element type. For the Timoshenko beam FEM analysis L6BEA elements will be used, which are two-noded as well, but includes shear deformation. In DIANA, all three convergence criteria are applied: the energy, displacement and force convergence norms are satisfied simultaneously with a convergence tolerance of $1.0*10^{-5}$.

Non-dimensionalization of the equations in the PINN model will be utilized where it is deemed necessary. This technique will be applied to the model if the model has not converged to the solution after 50,000 iterations for the non-linear cases and after 100,000 iterations for the linear cases. This limit

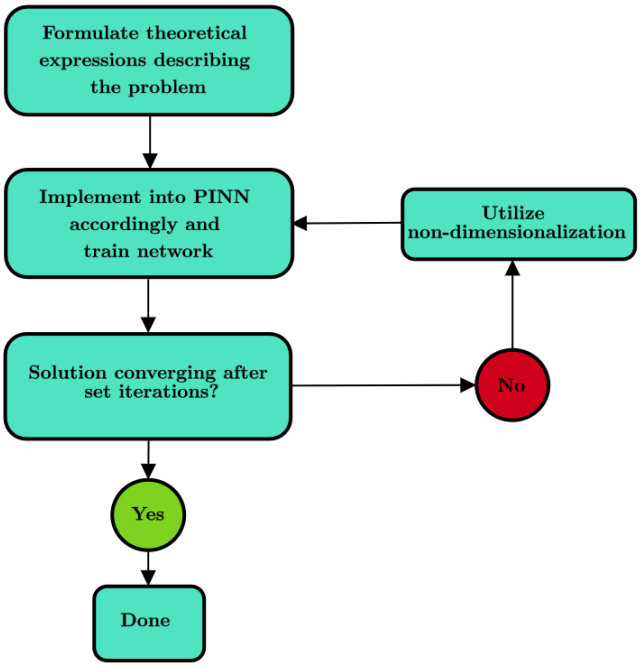


Figure 4.1: The process that is employed for building the PINN model.

is based on practicality as there is no guarantee that an accurate solution will not be achieved after this specified number of iterations. The process is summarized in Figure 4.1.

The performance metric that will be used to compare the accuracy of the output of the PINN model with respect to the FEM model is the relative error:

$$\frac{\|w_{PINN} - w_{ref}\|}{\|w_{ref}\|}$$

where w_{PINN} is the vector of the predicted displacements of the PINN model, and w_{ref} is the vector containing the exact analytical displacements when examining the linear cases. When examining the non-linear cases, the FEM output is used for w_{ref}.

For the non-linear cases, the loss functions will be formulated by utilizing the principle of virtual work based on the Green-Lagrange strain theory assuming Von Kármán strains. For both the Euler-Bernoulli and the Timoshenko beam A brief description of the principle of the virtual work is given in chapter 6.

Each optimization algorithm (the adaptive activation function and the adaptive weight loss algorithm) will be incorporated in the non-dimensionalized geometrically non-linear Timoshenko beam case and the result will be compared to the same model without using this algorithm.

5

Linear Analysis of One-Dimensional Beams

The linear cases of both beam theories will be examined to contextualize the research question and to highlight the difference in approach that is used for the non-linear cases. First, the linear Euler-Bernoulli beam is examined followed by the linear Timoshenko beam.

5.1. Euler-Bernoulli Beam

An Euler-Bernoulli beam with a constant bending stiffness can be described with the following differential equation:

$$EI\frac{d^4w}{dx^4} = q, \tag{5.1}$$

where w is the transverse deflection of the beam, E is the Young's modulus of the beam and I the second moment of area, with q being the distributed line load acting on the beam. For a simply supported beam with length L, the boundary conditions are:

$$w(0) = w(L) = 0,$$
$$\frac{d^2w}{dx^2}(0) = \frac{d^2w}{dx^2}(L) = 0,$$

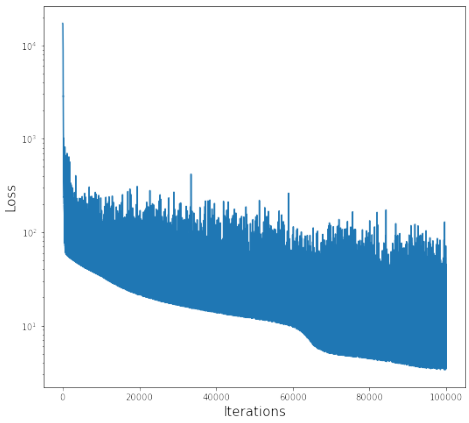
which results in the following analytical solution:

$$w_{exact}(x) = \frac{qL^4}{24EI}\left(\frac{x}{L} - \frac{2x^3}{L^3} + \frac{x^4}{L^4}\right).$$

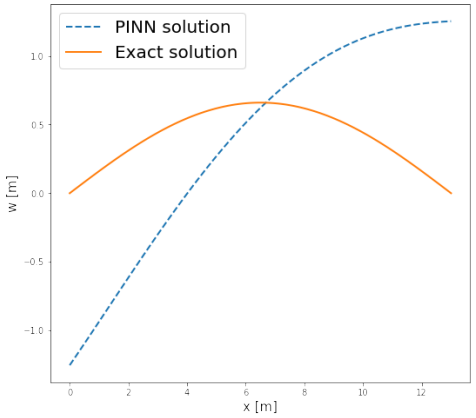
For the PINN model the input layer is chosen to consist of a single neuron s, which is a vector consisting of the locations of all collocation points on the beam and a vector suffices since it is one-dimensional problem. The output layer reflects the number of degrees of freedom (DOFs) in the system. In this case, the output layer consists of a single neuron w_{PINN} which is a vector containing the predicted deflection at each collocation point. Following from Eq. (3.1), the loss function for this case becomes:

$$\begin{aligned}\mathcal{L} = {} & (w_{PINN}(0) - 0)^2 + (w_{PINN}(L) - 0)^2 + \left(\frac{d^2w_{PINN}}{dx^2}(0) - 0\right)^2 + \left(\frac{d^2w_{PINN}}{dx^2}(L) - 0\right)^2 \\ & + \frac{1}{N_{PDE}}\sum_{i=1}^{N_{PDE}}\left(EI\frac{d^4w}{dx^4} - q\right)^2,\end{aligned} \tag{5.2}$$

where N_{PDE} is the amount of collocation points of the interior of the beam. The initial conditions are irrelevant, since a static problem is described. As an example, a simply supported beam with the



(a) The computed loss throughout the iterations.



(b) The displacement output after training.

Figure 5.1: The loss history of the PINN model after training and the predicted beam deflection after 100,000 training iterations.

following parameters is investigated: $q = 131\frac{kN}{m}$, $L = 13\ m$, $I = 3.6 \times 10^{-4}\ m^4$, $E = 2.05 \times 10^8\ \frac{kN}{m}^2$. The PINN model consists of one input layer and output layer, as stated before, and 3 hidden layers containing 10 neurons each. The interior of the beam consists of 40 equidistant collocation points, bringing the total to 42 if the boundaries are included. The weights and biases are initialized with an uniform distribution X ~ U (-0.001 , 0.001). The model has been trained for 100.000 iterations with a learning rate of 0.01 using the Adam optimizer. The hyperbolic tangent is used as the activation function for all layers except the output layer. The results are displayed in Figure 5.1.

The predicted solution in Figure 5.1 is very inaccurate despite the number of iterations that have passed. The loss decreases by order of magnitudes in the first thousand iterations, after which the loss begins to oscillate. This oscillation can be inconvenient if there is a set training iteration limit, instead of a loss tolerance limit, where the training is more likely to end on an iteration which coincides with high loss and thus a less accurate result.

A cause for this issue is that there is a large discrepancy between the variables present in equation (5.1). This leads to a slow and unstable training because the optimization algorithm is less consistent when accounting for this discrepancy (Wang et al., 2023). Non-dimensionalization is therefore a technique that could alleviate this issue.

We introduce the following non-dimensional variables:

$$\tilde{x} = \frac{x}{L},\ \tilde{w} = \frac{w}{L},\ \tilde{q} = \frac{qL^3}{EI},$$

resulting in the following non-dimensionalized differential equation:

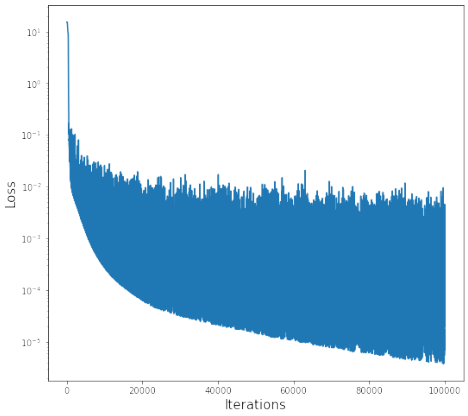
$$\frac{d^4\tilde{w}}{d\tilde{x}^4} = \tilde{q}.$$

The loss function for the non-dimensionalized case becomes:

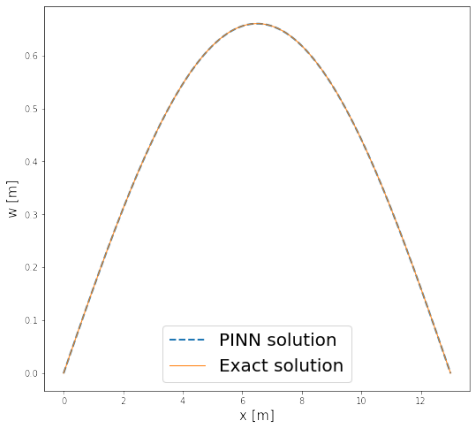
$$\mathcal{L} = (\tilde{w}_{PINN}(0) - 0)^2 + (\tilde{w}_{PINN}(1) - 0)^2 + \left(\frac{d^2\tilde{w}_{PINN}}{d\tilde{x}^2}(0) - 0\right)^2 + \left(\frac{d^2\tilde{w}_{PINN}}{d\tilde{x}^2}(1) - 0\right)^2 + \frac{1}{N_{PDE}}\sum_{i=1}^{N_{PDE}}\left(\frac{d^4\tilde{w}_{PINN}}{d\tilde{x}^4} - \tilde{q}\right)^2. \tag{5.3}$$

In Figure 5.2, the results of the model utilizing this new loss function are shown.

For the dimensionalized case, the relative error after training is equal to 162%, where the model fails to converge, whereas it is 0.0267% for the non-dimensionalized case. It is worth noting that the

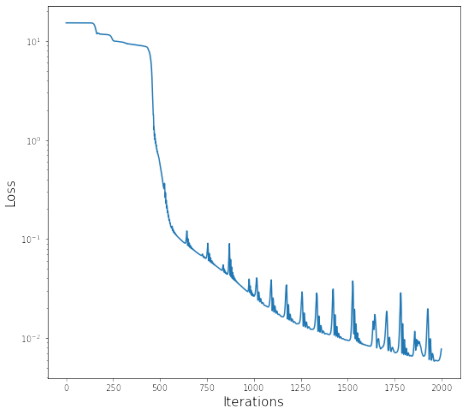


(a) The computed loss throughout the iterations.

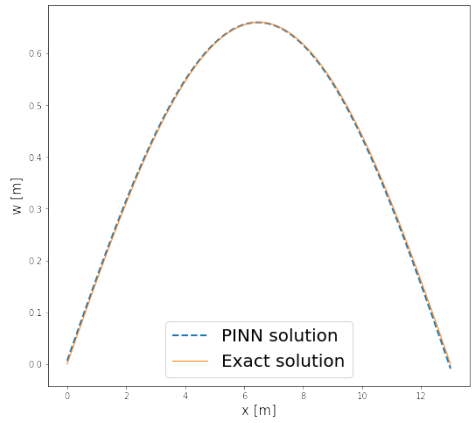


(b) The displacement output after training.

Figure 5.2: The PINN model with the non-dimensionalized loss function. The loss after 100.000 iterations is now orders of magnitude lower.



(a) The computed loss throughout the iterations.



(b) The displacement output after training.

Figure 5.3: The PINN model after training it for 2000 iterations. The model achieves satisfactory accuracy after 2000 iterations.

non-dimensionalized PINN model achieves a relative error of 0.98% after only 2000 iterations as shown in Figure 5.3.

5.2. Timoshenko Beam

The linear Timoshenko beam can be described with the following coupled ODEs:

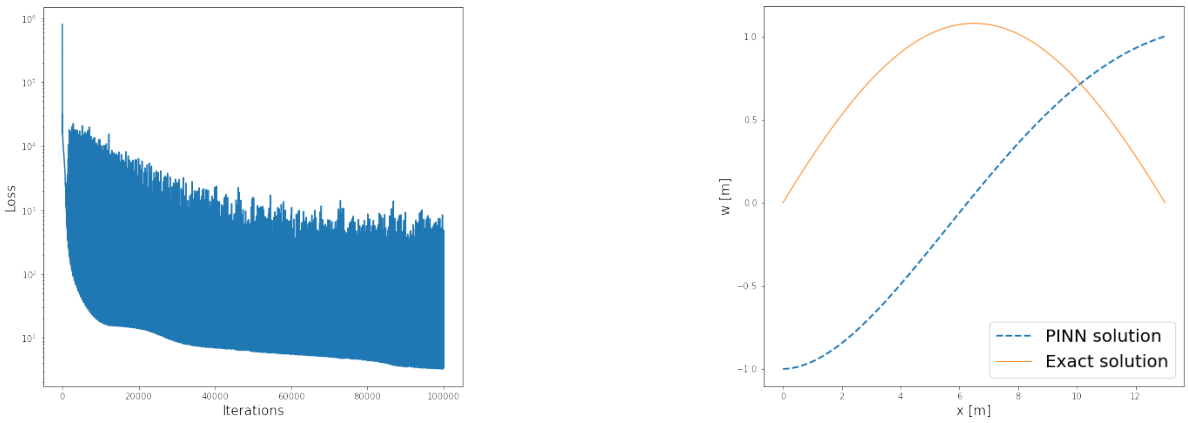
$$EI\phi_{xx} + GA_s\left(w_x - \phi\right) = 0,$$
$$GA_s\left(w_{xx} - \phi_x\right) + q = 0,$$

where G is the shear modulus, A_s is the effective shear area and φ the rotation of the cross-section with respect to the w-axis. The non-dimensional variables are as follows:

$$\tilde{x} = \frac{x}{L}\ ,\ \tilde{w} = \frac{w}{L}\ ,\ \tilde{q} = \frac{qL}{GA_s},\ \chi = \frac{EI}{L^2GA_s},$$

which results in the following non-dimensional coupled ODEs:

$$\chi\phi_{xx} + \tilde{w}_x - \phi = 0,$$
$$\tilde{w}_{xx} - \phi_x + \tilde{q} = 0.$$



(a) The computed loss throughout the iterations.

(b) The displacement output after training.

Figure 5.4: The loss history and output after 100.000 iterations of training the PINN model simulating a simply supported Timoshenko beam model.

The χ holds theoretical significance. It describes the ratio between bending and shear stiffness. If χ tends to zero (the beam has infinite shear stiffness), then $\tilde{w}_x = \phi$ and so the system reverts to an Euler-Bernoulli beam scenario.

The output layer of the PINN model now consists of two output neurons, w_{PINN} and ϕ_{PINN}. For the following numerical example, the same situation and parameters that have been used for the Euler-Bernoulli example are kept the same. A_s is set to 8.333e-5 m^2. The exact solution of a simply supported beam subject to a distributed line load is:

$$w_{exact}(x) = \frac{qL^4}{24EI}\left(\frac{x}{L} - \frac{2x^3}{L^3} + \frac{x^4}{L^4}\right) + \frac{qL^2}{2GA_s}\left(\frac{x}{L} - \frac{x^2}{L^2}\right).$$

The loss function of the non-dimensionalized Timoshenko beam in this case becomes:

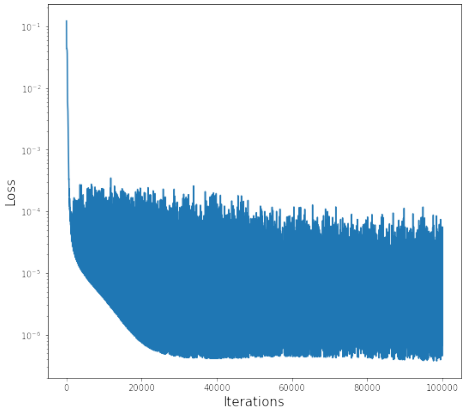
$$\begin{aligned}\mathcal{L} = &\left(\tilde{w}_{PINN}(0) - 0\right)^2 + \left(\tilde{w}_{PINN}(1) - 0\right)^2 + \left(\phi_{PINN}(0) - 0\right)^2 + \left(\phi_{PINN}(1) - 0\right)^2 \\ &+ \frac{1}{N_{PDE}} \sum_{i=1}^{N_{PDE}} \left(\chi\phi_{PINN,xx} + \tilde{w}_{PINN,x} - \phi_{PINN}\right)^2 \\ &+ \frac{1}{N_{PDE}} \sum_{i=1}^{N_{PDE}} \left(\tilde{w}_{PINN,xx} - \phi_{PINN,x} + q\right)^2.\end{aligned} \tag{5.4}$$

The results of the regular dimensionalized model after training are shown in Figure 5.4.

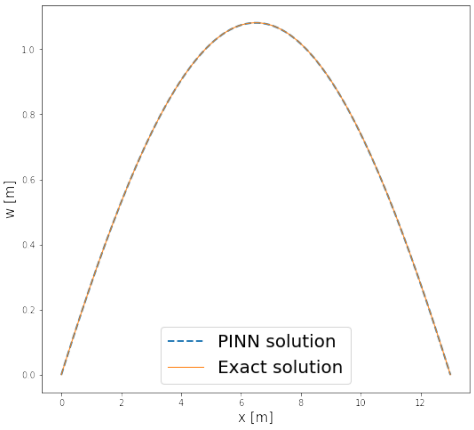
The accuracy of the trained model remains very insufficient, even after 100.000 iterations, reaching a similar loss magnitude to the dimensionalized Euler-Bernoulli case. The results of the non-dimensionalized model are shown in Figure 5.5.

The chaotic oscillations of the losses in the aforementioned examples are the result of selecting a relatively high learning rate of 0.01. Selecting a lower learning rate results in a more stable training progress, but at the cost of sacrificing overall efficiency. The results are shown in Figure 5.6.

The model achieves a relative error of 0.0008% after training it for 100.000 iterations which is order of magnitudes lower than with a learning rate of 0.01, as shown in Figure 5.7. It takes overall longer, however, for the model with this learning rate to achieve a satisfactory accuracy. There exists a trade-off between a low and high learning rate, where the former results overall in better accuracy in the long term whereas the latter achieves a more accurate result faster, but has relatively lower accuracy in the long term, as displayed in Figure 5.7. For the linear beam cases, it is recommended that a non-dimensionalized model is used with a high learning rate in conjunction with a set loss tolerance limit. This ensures that an accurate output is reached relatively quickly, while the unpredictable output caused by the oscillatory nature of a high learning rate is nullified.

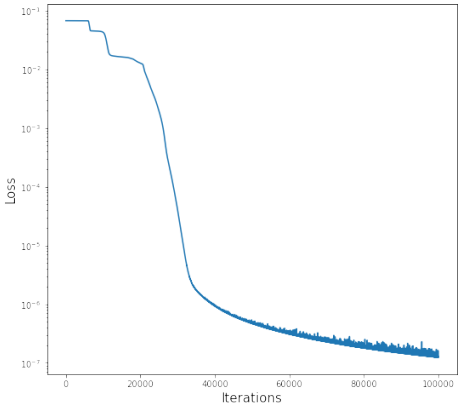


(a) The computed loss throughout the iterations.

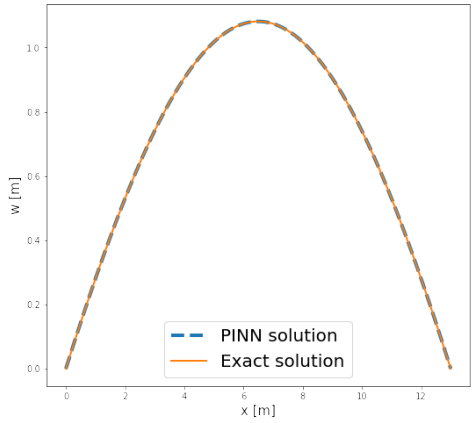


(b) The displacement output after training.

Figure 5.5: The loss history and output after 100.000 iterations of training the non-dimensionalized Timoshenko beam PINN model. The relative error after training is 0.0349%.

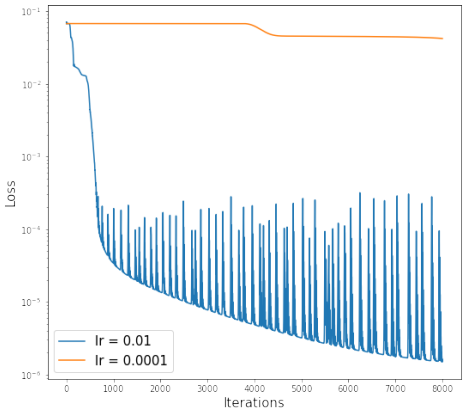


(a) The computed loss throughout the iterations.

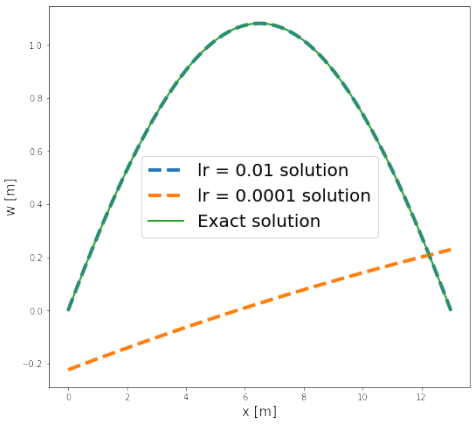


(b) The displacement output after training.

Figure 5.6: The loss history and output after 100.000 iterations of training the non-dimensionalized Timoshenko beam PINN model with a learning rate of 0.0001.



(a) The computed loss throughout the iterations.



(b) The displacement output after training.

Figure 5.7: A comparison between learning rates when solving the linear Timoshenko beam case after training for 8000 iterations.

6

Non-linear Analysis of One-dimensional Beams

In this chapter the geometrically non-linear Euler-Bernoulli beam will be investigated, followed by the geometrically non-linear Timoshenko beam. A brief description of the utilized theory will be given and the applied equations in the PINN model will be derived.

6.1. Green-Lagrange Strain

For implementing geometric non-linearity we utilize the Green-Lagrange strain:

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} + \frac{\partial u_m}{\partial x_i}\frac{\partial u_m}{\partial x_j}\right).$$

Consider a beam within a two-dimensional space. Let:

$$\begin{aligned} u_1 &= u_0(x_1) - x_3\phi(x_1), \\ u_2 &= 0, \\ u_3 &= w(x_1), \end{aligned} \tag{6.1}$$

where u_0 is the axial deformation at the neutral axis, u_1 is the axial deformation along any fiber of the beam, u_2 is the out of plane deformation and u_3 the deformation of the fiber at the neutral axis in the x_3 direction, as depicted in Figure 6.1. ϕ is the angle between the deformed and undeformed plane cross-section of the beam.

Let $x_1 = x$, $x_2 = y$, $x_3 = z$. Utilizing the defined displacements in (6.1), the resulting strains will be:

$$\begin{aligned} \epsilon_{xx} &= \frac{\partial u_0}{\partial x} - z\frac{\partial \phi}{\partial x} + \frac{1}{2}\left(\frac{\partial u_0}{\partial x} - z\frac{\partial \phi}{\partial x}\right)^2 + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2, \\ \epsilon_{yy} &= 0, \\ \epsilon_{zz} &= \frac{1}{2}\phi^2, \\ \epsilon_{xy} &= \epsilon_{yx} = 0, \\ \epsilon_{xz} &= \epsilon_{zx} = \frac{1}{2}\left(\frac{\partial w}{\partial x} - \phi\left(1 + \frac{\partial w}{\partial x}\right)\right). \end{aligned} \tag{6.2}$$

We assume Von Kármán strains, which indicate that significant rotations and small strains are considered (Khodabakhshi & Reddy, 2017). Furthermore we assume the engineering strain definition ($\gamma_{xz} = 2\epsilon_{xz}$). The strains in (6.2) then simplify to:

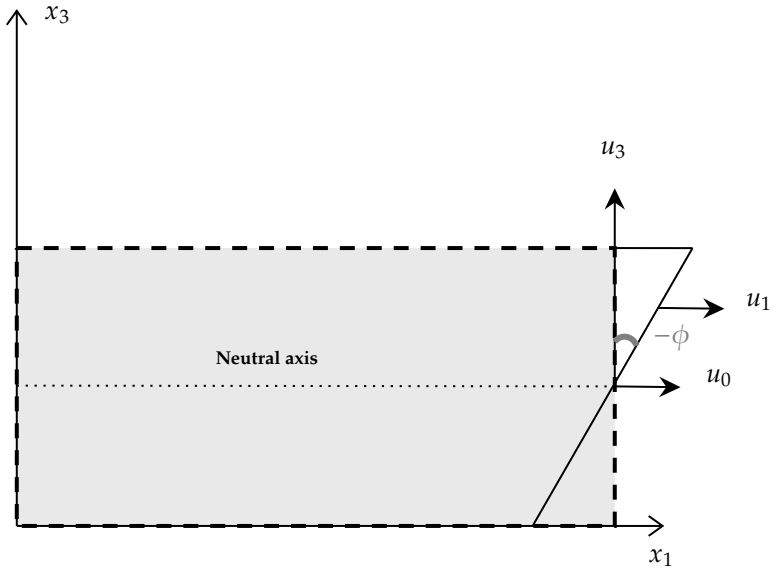


Figure 6.1: The defined displacements. ϕ is positive if the angle goes counter-clockwise around the x_2 axis, which is orthogonal to the x_1-x_3 plane, pointing into the page.

$$\begin{aligned} \epsilon_{xx} &= \frac{\partial u_0}{\partial x} - z\frac{\partial \phi}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2, \\ \epsilon_{yy} &= \epsilon_{zz} = \epsilon_{xy} = 0, \\ \gamma_{xz} &= \frac{\partial w}{\partial x} - \phi. \end{aligned} \tag{6.3}$$

The strains in (6.3) apply to the Timoshenko beam. Note that (6.1) describe displacements which result in a constant shear strain along the cross-section (γ_{xz} in (6.3)). This is not in accordance with the fact that the shear strain is not necessarily constant across the cross-section. The total strain energy in the cross-section can be made accurate by applying a shear correction factor, as shown in Figure 6.2. The shear correction factor depends on the cross-section; this factor is equal to $5/6$ for a homogeneous rectangular cross-section.

For the Euler-Bernoulli case there is no shear deformation, $\gamma_{xz} = 0$, and so (6.3) simplifies to:

$$\begin{aligned} \epsilon_{xx} &= \frac{\partial u_0}{\partial x} - z\frac{\partial^2 w}{\partial x^2} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2, \\ \epsilon_{yy} &= \epsilon_{zz} = \epsilon_{xy} = 0, \\ \phi &= \frac{\partial w}{\partial x}. \end{aligned} \tag{6.4}$$

Note that in (6.4), the non-linear terms of the shear strain have become negligible.

6.2. Principle of Virtual Work

For performing the analysis using PINNs, the principle of virtual work is considered in formulating the loss functions. This principle states that a system is in equilibrium if the internal virtual work is equal to the external virtual work:

$$\delta W_{Internal} = \delta W_{External},$$

where $\delta W_{Internal}$ is the total virtual strain energy stored in the system and $\delta W_{External}$ is the work applied due to external forces. In the context of a beam system subjected to transverse and axial forces,

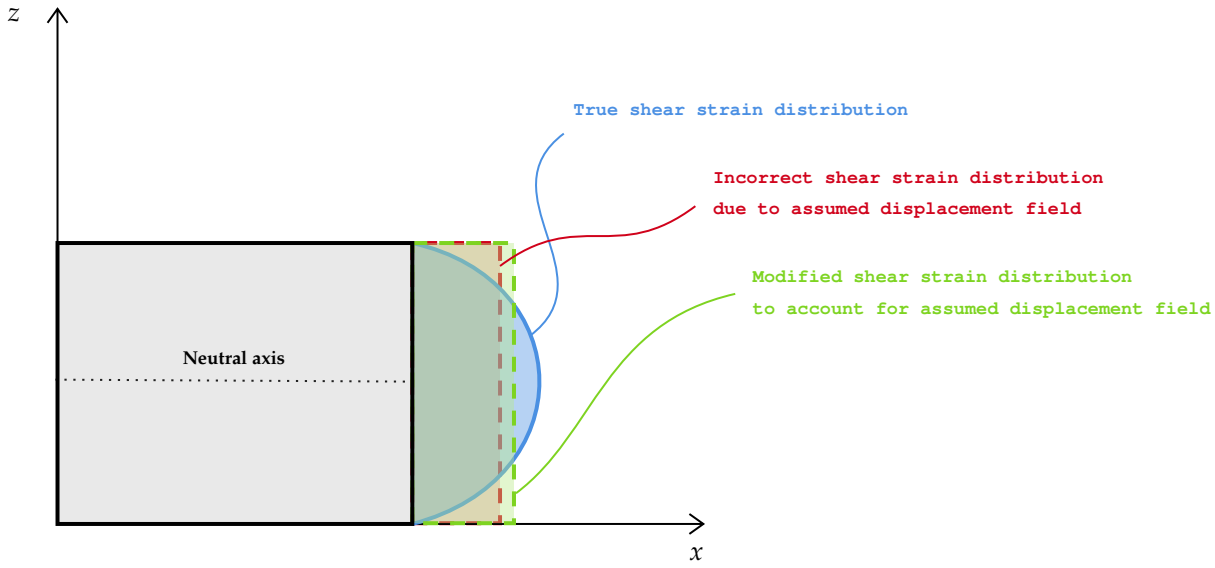


Figure 6.2: Utilizing the Timoshenko beam theory results in an uniform transverse shear distribution (red) for the cross-section, which is not in accordance with the true shear distribution. A shear correction factor has to be applied so that the total shear strain energy is still correct.

$\delta W_{Internal}$ and $\delta W_{External}$ can be specified as:

$$\begin{aligned}
\delta W_{Internal} &= \int_V \delta\epsilon_{ij}\sigma_{ij}dV, \\
\delta W_{External} &= \int_l q\,(x)\,\delta w\,dx + \int_l f\,(x)\,\delta u\,dx + \sum_{i=1}^{n} F_i \delta u_i\,(x_i), \\
\sigma_{xx} &= E\epsilon_{xx},
\end{aligned} \tag{6.5}$$

where ϵ is the internal strain, σ is the internal stress, $q\,(x)$ is the distributed transverse load along the beam, δw is the virtual transverse displacement, δu is the virtual axial displacement, F_i is the point load at i and $\delta u_i(x_i)$ is the displacement at location x_i where the point load is applied. E is the elastic modulus.

6.3. Euler-Bernoulli Beam

For the analysis of a geometrically non-linear Euler-Bernoulli beam the kinematic relations in Eq. 6.5 are assumed. Combining (6.4) and (6.5) results in the following internal work equation for the Euler-Bernoulli beam:

$$\begin{aligned}
\delta W_{Internal} = {} & E\int_0^L \int_A \frac{\partial \delta u_0}{\partial x}\left(\frac{\partial u_0}{\partial x} - z\frac{\partial^2 w}{\partial x^2} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right) dA\,dx \\
& - E\int_0^L \int_A z\frac{\partial^2 \delta w}{\partial x^2}\left(\frac{\partial u_0}{\partial x} - z\frac{\partial^2 w}{\partial x^2} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right) dA\,dx \\
& + E\int_0^L \int_A \frac{\partial \delta w}{\partial x}\frac{\partial w}{\partial x}\left(\frac{\partial u_0}{\partial x} - z\frac{\partial^2 w}{\partial x^2} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right) dA\,dx.
\end{aligned} \tag{6.6}$$

Note that $\delta\epsilon_{xx} = \dfrac{\partial \delta u_0}{\partial x} - z\dfrac{\partial^2 \delta w}{\partial x^2} + \dfrac{\partial \delta w}{\partial x}\dfrac{\partial w}{\partial x}$. Furthermore, we can state:

$$I = \int_A z^2 \, dA,$$
$$S = \int_A z \; dA,$$

where S is the first moment of area. This is equal to zero for a cross-section that is symmetric around the neutral axis. If this is the case, then (6.6) simplifies to:

$$\delta W_{Internal} = EA \int_0^L \frac{\partial \delta u_0}{\partial x}\left(\frac{\partial u_0}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right) dx + EI \int_0^L \frac{\partial^2 \delta w}{\partial x^2}\frac{\partial^2 w}{\partial x^2} dx$$
$$+ EA \int_0^L \frac{\partial \delta w}{dx}\frac{\partial w}{dx}\left(\frac{\partial u_0}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right) dx.$$

Combining the internal and external work equations and separating them with respect to each type of virtual displacement, we get:

$$\delta W_{\delta u_0} = EA \int_0^L \frac{\partial \delta u_0}{\partial x}\left(\frac{\partial u_0}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right) dx - \int_0^L f(x)\,\delta u_0 \, dx - \sum_{i=1}^{n} F_{h,i}\delta u_{0,i} = 0,$$
$$\delta W_{\delta w} = EI \int_0^L \frac{\partial^2 \delta w}{\partial x^2}\frac{\partial^2 w}{\partial x^2} dx + EA \int_0^L \frac{\partial \delta w}{dx}\frac{\partial w}{dx}\left(\frac{\partial u_0}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right) dx - \int_0^L q(x)\,\delta w \, dx$$
$$- \sum_{i=1}^{n} F_{v,i}\delta w_i - \sum_{i=1}^{n} M_i \frac{\delta w_i}{dx_i} = 0,$$

where $F_{h,i}$ are the external axial point loads, $F_{v,i}$ are the external transverse force loads and M_i are the external applied moments. $\delta W_{\delta u_0}$ relates to the axial deformations of the system whereas $\delta W_{\delta w}$ relates to the transverse and bending deformations of the system.

The loss function of a geometrically non-linear Euler-Bernoulli beam, considering virtual work, would be the following:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{W_{\delta u_o}} + \lambda_2 \mathcal{L}_{W_{\delta w}} + \textstyle\sum_i^n \lambda_i \mathcal{L}_{BC,i},$$

with:

$$\mathcal{L}_{W_{\delta u_o}} = \delta W_{\delta u_0}^2,$$
$$\mathcal{L}_{W_{\delta w}} = \delta W_{\delta w}^2.$$

An issue that arises is that the trivial solution, where there are zero displacements, is something which now has to be avoided unlike the linear cases with the strong form formulations. We introduce the following term to also be a part of the loss function:

$$\mathcal{L}_g = \frac{\lambda_g^2}{\left(\int_0^L |u_{PINN}(x)|\, dx + \int_0^L |w_{PINN}(x)|\, dx\right)^2},$$

where λ_g is a scalar utilized tot balance this loss term. This term is meant to penalize the model if it attempts to converge towards the trivial solution. Because of this term, the trivial solution no longer is a (global) minimum, instead it becomes a "peak". The final loss function becomes:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{W_{\delta u_o}} + \lambda_2 \mathcal{L}_{W_{\delta w}} + \textstyle\sum_i^n \lambda_{BC,i} \mathcal{L}_{BC,i} + \mathcal{L}_g.$$

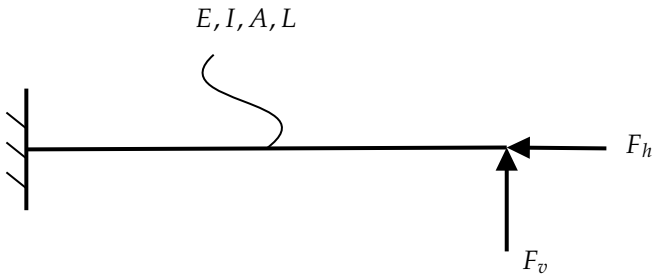


Figure 6.3: The case investigated for the non-linear Euler-Bernoulli beam analysis.

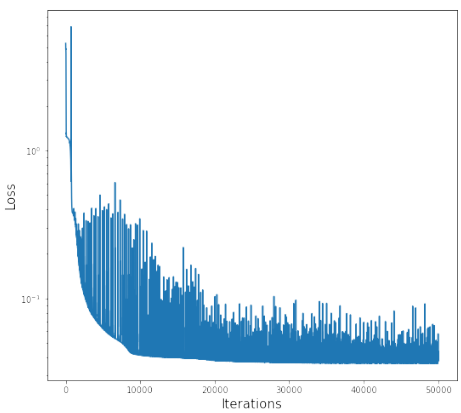
Property	Value
Elastic modulus, E [kPa]	2.05×10^8
Second moment of area, I [m^4]	1.6×10^{-4}
Cross-sectional area, A [m^2]	1.15×10^{-2}
Length, L [m]	10
Vertical point load, F_v [kN]	10
Horizontal point load, F_h [kN]	200

Table 6.1: The properties of the case used for the non-linear Euler-Bernoulli analysis using a symmetric cross-section.

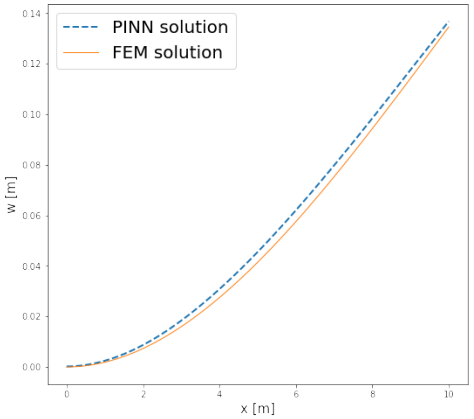
A cantilever beam with a vertical point load and a horizontal compressive load acting on the edge is investigated as portrayed in Figure 6.3. The beam properties are described in Table 6.1.

The analysis with the PINN model was done using 1 input neuron and 2 output neurons (horizontal and vertical displacement vector). The beam interior consists of 99 equidistant collocation points, bringing the total to 101 if the boundaries are included. The weights and biases are initialized with an uniform distribution X ~ U (0, 0.01). The model has been trained for 50.000 iterations with a learning rate of 0.001 using the Adam optimizer and the hyperbolic tangent is used as the activation function for all layers except the output layer. Furthermore, the hyperparameters $\lambda_1 = \lambda_2 = 1$, $\lambda_{BC} = 100000$ and $\lambda_g = 0.1$. The results are shown in Figures 6.4 and 6.5.

The mentioned initialization of the weights and biases, X ~ U (0, 0.01), is purposefully chosen such that the beam remains close to the undeformed state before any training. This results in the beam converging faster to the FEM solution. In Figure 6.6, the results are shown of the same model using a Xavier normal distribution as a initialization method instead. Gloriot & Bengio (2010) introduced the Xavier initialization as a method of avoiding the phenomena of vanishing and exploding gradients. The initial weights can cause the initial deflection of the beam (and so also the loss) to be very large,

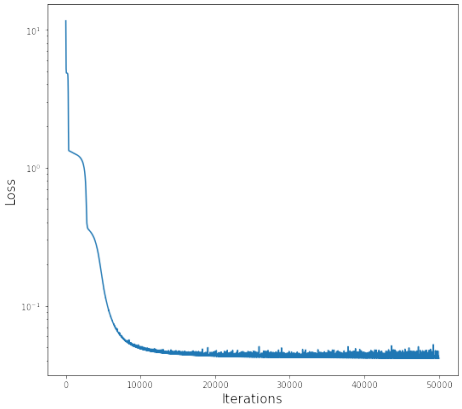


(a) The computed loss throughout the iterations.

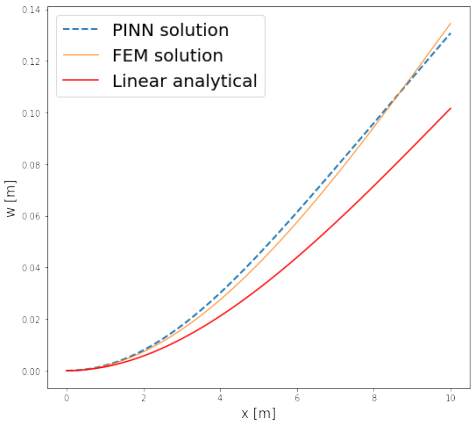


(b) The displacement output after training.

Figure 6.4: The result of the cantilever example with a learning rate of 0.001. The relative error of the PINN solution with respect to the FEM solution is 4.99%.

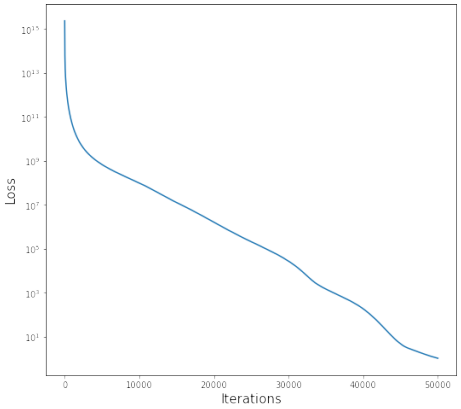


(a) The computed loss throughout the iterations.

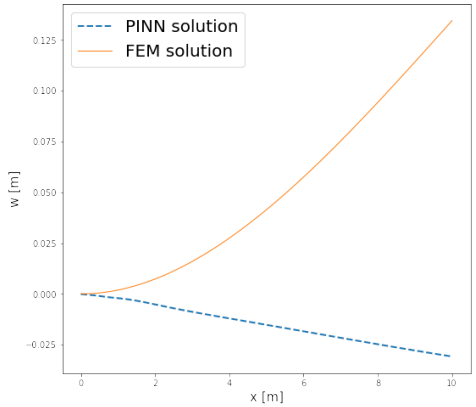


(b) The displacement output after training.

Figure 6.5: The result of the cantilever example with a learning rate of 0.0001. The relative error of the PINN solution with respect to the FEM solution is 3.62%.



(a) The computed loss throughout the iterations.



(b) The displacement output after training.

Figure 6.6: The output of the same model but with a Xavier normal distribution utilized as initialization method. The initial loss is so large that no accurate solution is achieved after the same amount of training iterations

however, which can necessitate more training iterations before reaching an accurate solution. In this case, the model failed to converge within the 50,000 iterations.

6.4. Timoshenko Beam

Applying (6.3) to get the internal virtual work components of the beam results in the following for the axial component:

$$\begin{aligned}\int_V \delta\epsilon_{xx}\sigma_{xx}dV = E\int_0^L\int_A \frac{\partial\delta u_0}{\partial x}\left(\frac{\partial u_0}{\partial x} - z\frac{\partial\phi}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right)dA\,dx \\ - E\int_0^L\int_A z\frac{\partial\delta\phi}{\partial x}\left(\frac{\partial u_0}{\partial x} - z\frac{\partial\phi}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right)dA\,dx \\ + E\int_0^L\int_A \frac{\partial\delta w}{\partial x}\frac{\partial w}{\partial x}\left(\frac{\partial u_0}{\partial x} - z\frac{\partial\phi}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right)dA\,dx.\end{aligned} \tag{6.7}$$

If the cross-section is symmetric with respect to the neutral axis, (6.7) simplifies to:

$$\begin{aligned}\int_V \delta\epsilon_{xx}\sigma_{xx}dV = \; & EA\int_0^L \frac{\partial\delta u_0}{\partial x}\left(\frac{\partial u_0}{\partial x}+\frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right)dx \\ & + EI\int_0^L \frac{\partial\delta\phi}{\partial x}\frac{\partial\phi}{\partial x}dx \\ & + EA\int_0^L \frac{\partial\delta w}{dx}\frac{\partial w}{dx}\left(\frac{\partial u_0}{\partial x}+\frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right)dx.\end{aligned} \tag{6.8}$$

For the shear component, it is:

$$\begin{aligned}\int_V \delta\gamma_{xz}\sigma_{xz}dV = \; & G\int_0^L\int_A \left(\frac{\partial\delta w}{\partial x}-\delta\phi\right)\left(\frac{\partial w}{\partial x}-\phi\right)dA\,dx \\ & = GA_S\int_0^L \frac{\partial\delta w}{dx}\frac{\partial w}{dx}-\frac{\partial w}{dx}\delta\phi-\phi\frac{\partial\delta w}{dx}+\phi\delta\phi\;dx.\end{aligned} \tag{6.9}$$

The total internal work therefore is the summation of (6.8) and (6.9):

$$\begin{aligned}\delta W_{Internal} = \; & EA\int_0^L \frac{\partial\delta u_0}{\partial x}\left(\frac{\partial u_0}{\partial x}+\frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right)dx \\ & + EI\int_0^L \frac{\partial\delta\phi}{\partial x}\frac{\partial\phi}{\partial x}dx \\ & + EA\int_0^L \frac{\partial\delta w}{dx}\frac{\partial w}{dx}\left(\frac{\partial u_0}{\partial x}+\frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right)dx \\ & + GA_S\int_0^L \frac{\partial\delta w}{dx}\frac{\partial w}{dx}-\frac{\partial w}{dx}\delta\phi-\phi\frac{\partial\delta w}{dx}+\phi\delta\phi\;dx.\end{aligned}$$

Combining the internal and external work equations and separating them with respect to each type of virtual displacement as done earlier, we get:

$$\begin{aligned}\delta W_{\delta u_0} = \; & EA\int_0^L \frac{\partial\delta u_0}{\partial x}\left(\frac{\partial u_0}{\partial x}+\frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right)dx - \int_0^L f(x)\,\delta u_0\,dx - \sum_{i=1}^n F_{h,i}\delta u_{0,i} = 0, \\ \delta W_{\delta w} = \; & EA\int_0^L \frac{\partial\delta w}{dx}\frac{\partial w}{dx}\left(\frac{\partial u_0}{\partial x}+\frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right)dx + GA_S\int_0^L \frac{\partial\delta w}{dx}\frac{\partial w}{dx}-\phi\frac{\partial\delta w}{dx}\;dx \\ & - \int_0^L q(x)\,\delta w\,dx - \sum_{i=1}^n F_{v,i}\delta w_i = 0, \\ \delta W_{\delta\phi} = \; & EI\int_0^L \frac{\partial\delta\phi}{\partial x}\frac{\partial\phi}{\partial x}dx + GA_S\int_0^L \phi\delta\phi-\frac{\partial w}{dx}\delta\phi\;dx - \sum_{i=1}^n M_i\delta\phi_i = 0.\end{aligned}$$

The loss equations of the Timoshenko PINN model are:

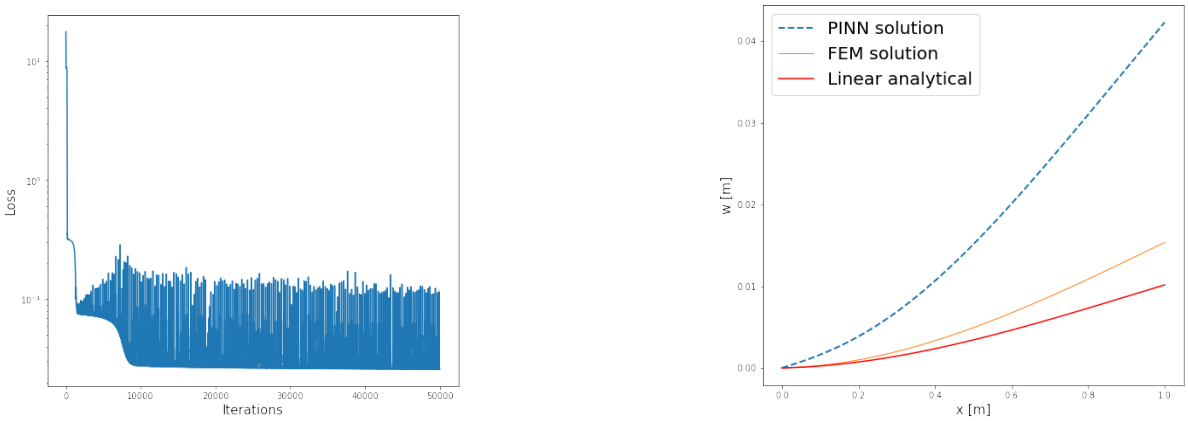
$$\mathcal{L} = \lambda_1\mathcal{L}_{W_{\delta u_o}} + \lambda_2\mathcal{L}_{W_{\delta w}} + \lambda_3\mathcal{L}_{W_{\delta\phi}} + \textstyle\sum_i^n \lambda_i\mathcal{L}_{BC,i} + \mathcal{L}_g.$$

Note that there is an additional loss term added related to ϕ, which now is a separate degree of freedom. The loss terms, similar to the Euler-Bernoulli case, are defined as:

$$\mathcal{L}_{W_{\delta u_o}} = \delta W^2_{\delta u_0},\; \mathcal{L}_{W_{\delta w}} = \delta W^2_{\delta w},\; \mathcal{L}_{W_{\delta\phi}} = \delta W^2_{\delta\phi}.$$

Property	Value
Elastic modulus, E [kPa]	2.05×10^8
Second moment of area, I [m^4]	1.6×10^{-4}
Cross-sectional area, A [m^2]	1.15×10^{-2}
Length, L [m]	1
Vertical point load, F_v [kN]	1000
Horizontal point load, F_h [kN]	20000

Table 6.2: The properties of the case used for the non-linear Timoshenko analysis using a symmetric cross-section.



(a) The computed loss throughout the iterations.

(b) The displacement output after training.

Figure 6.7: The output of the Timoshenko beam analysis after training the model for 50000 iterations. The relative error is 186% with respect to the FEM solution.

The PINN model now has three output neurons: u, w, and ϕ. The input remains a singular neuron which is the vector of collocation points along the beam.

For the numerical example, the same cantilever example will be utilized with the relevant parameters described in Table 6.2.

The analysis was conducted with 101 collocation points. The weights and biases are initialized with an uniform distribution X ~ U (0, 0.01). The model has been trained for 50.000 iterations with a learning rate of 0.001 using the Adam optimizer and the hyperbolic tangent is used as the activation function for all layers except the output layer. $\lambda_1 = \lambda_2 = \lambda_3 = 10^{-6}$, $\lambda_4 = 10^5$, and $\lambda_g = \sqrt{0.00001}$. The results are shown in Figure 6.7.

The issue that arises by using the introduced penalty term, $\mathcal{L}_g$, is the trial and error nature of finding the optimal value of λ_g. The accuracy of the PINN output is heavily reliant on the magnitude of $\mathcal{L}_g$ relative to the other loss terms. The PINN result shown in figure 6.8 included a loss $\mathcal{L}_g$ of 0.021 after training, whereas the losses of $\mathcal{L}_{W_{\delta u_o}}$, $\mathcal{L}_{W_{\delta w_o}}$ and $\mathcal{L}_{W_{\delta \phi}}$ were 0.026, 0.008 and 0.034 respectively. The $\mathcal{L}_g$ having a loss of the same magnitude as the other three mentioned is detrimental for the accuracy of the model, as $\mathcal{L}_g$ is simply implemented to avoid the trivial solution and does not contribute to the mechanical formulation of the problem. To circumvent this issue, a more robust penalty term is needed. Inspired by the FEM in which the force convergence criterion is used for non-linear analyses, consider the following force residual:

$$\mathcal{L}_{g_1} = \|M_{ext} - M_{int}\|^2,$$
$$\mathcal{L}_{g_2} = \|V_{ext} - V_{int}\|^2,$$
$$\mathcal{L}_{g_3} = \|N_{ext} - N_{int}\|^2.$$

For every force type, the l^2-norm squared is taken. In other words, these three terms ensure that there is force equilibrium at every interior collocation point. Here, M_{ext}, V_{ext} and N_{ext} denote the moment,

shear and axial force vectors due to the external loads, respectively. These describe the external load acting on the beam with respect to its deformed shape. For the cantilever example, these would be:

$$\begin{aligned} M_{ext}(x) &= F_v(L-x) - F_h w(x), \\ V_{ext}(x) &= -F_v \cos\theta(x) + F_h \sin\theta(x), \\ N_{ext}(x) &= F_h \cos\theta(x) + F_v \sin\theta(x). \end{aligned}$$

The internal force components M_{int}, V_{int} and N_{ext} for an arbitrary cross-section, following the Timoshenko beam theory, are:

$$\begin{aligned} M_{int}(x) &= E\int_A \left(\frac{\partial u_0}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right) z\, dA + EI\frac{\partial \phi}{\partial x}, \\ V_{int}(x) &= GA_s\left(\phi + \frac{\partial w}{\partial x}\right), \\ N_{int}(x) &= EA\left(\frac{\partial u_0}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right) + E\int_A \frac{\partial \phi}{\partial x} z\, dA. \end{aligned}$$

For a symmetric cross-section, the coupling terms between normal force and bending vanish and these equations simplify to:

$$\begin{aligned} M_{int}(x) &= EI\frac{\partial \phi}{\partial x}, \\ V_{int}(x) &= GA_s\left(\phi + \frac{\partial w}{\partial x}\right), \\ N_{int}(x) &= EA\left(\frac{\partial u_0}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2\right). \end{aligned}$$

Furthermore, the following non-dimensionalized virtual work equations of the Timoshenko beam are used to enhance stability:

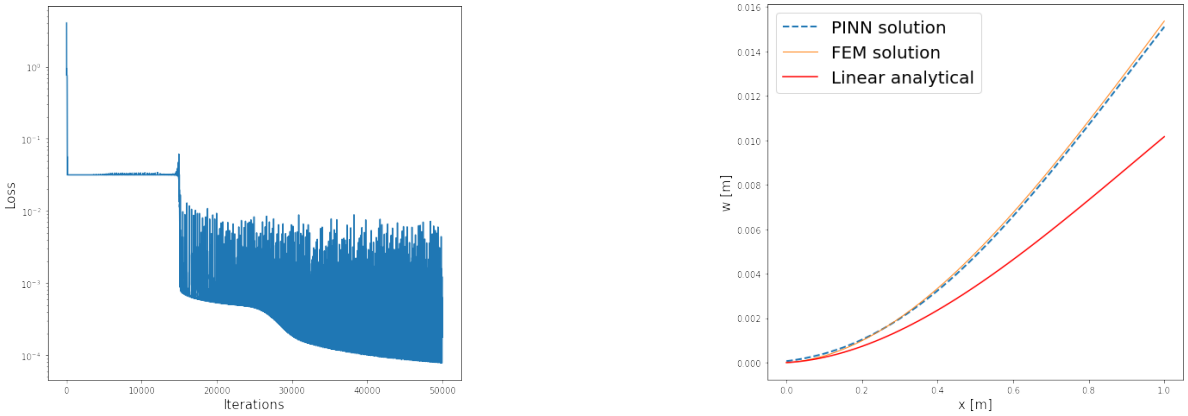
$$\begin{aligned} \delta\tilde{W}_{\delta u_0} &= \int_0^1 \frac{\partial \delta\tilde{u}_0}{\partial \tilde{x}}\left(\frac{\partial \tilde{u}_0}{\partial \tilde{x}} + \frac{1}{2}\left(\frac{\partial \tilde{w}}{\partial \tilde{x}}\right)^2\right) d\tilde{x} - \int_0^1 \tilde{f}(\tilde{x})\,\delta\tilde{u}_0\, d\tilde{x} - \sum_{i=1}^{n} \tilde{F}_{h,i}\delta\tilde{u}_{0,i} = 0, \\ \delta\tilde{W}_{\delta w} &= \frac{EA}{GA_s}\int_0^1 \frac{\partial \delta\tilde{w}}{d\tilde{x}}\frac{\partial \tilde{w}}{d\tilde{x}}\left(\frac{\partial \tilde{u}_0}{\partial \tilde{x}} + \frac{1}{2}\left(\frac{\partial \tilde{w}}{\partial \tilde{x}}\right)^2\right) d\tilde{x} + \int_0^1 \frac{\partial \delta\tilde{w}}{d\tilde{x}}\frac{\partial \tilde{w}}{d\tilde{x}} - \phi\frac{\partial \delta\tilde{w}}{d\tilde{x}}\, d\tilde{x} \\ &\quad - \int_0^1 \tilde{q}(\tilde{x})\,\delta\tilde{w}\, d\tilde{x} - \sum_{i=1}^{n} \tilde{F}_{v,i}\delta\tilde{w}_i = 0, \\ \delta\tilde{W}_{\delta\phi} &= \int_0^1 \frac{\partial \delta\phi}{\partial \tilde{x}}\frac{\partial \phi}{\partial \tilde{x}} d\tilde{x} + \frac{GA_S L^2}{EI}\int_0^1 \phi\delta\phi - \frac{\partial \tilde{w}}{d\tilde{x}}\delta\phi\, d\tilde{x} - \sum_{i=1}^{n} \tilde{M}_i\delta\phi_i = 0, \end{aligned}$$

with:

$$\tilde{x} = \frac{x}{L},\ \tilde{u}_0 = \frac{u_0}{L},\ \tilde{w} = \frac{w}{L},\ \tilde{f}(\tilde{x}) = \frac{f(x)L}{EA},\ \tilde{F}_h = \frac{F_h}{EA},\ \tilde{q}(\tilde{x}) = \frac{q(x)L}{GA_s},\ \tilde{F}_v = \frac{F}{GA_s},\ \tilde{M} = \frac{ML}{EI}.$$

The loss function becomes:

$$\mathcal{L} = \lambda_1\mathcal{L}_{W_{\delta u_o}} + \lambda_2\mathcal{L}_{W_{\delta w}} + \lambda_3\mathcal{L}_{W_{\delta\phi}} + \lambda_4\mathcal{L}_{g_1} + \lambda_5\mathcal{L}_{g_2} + \lambda_6\mathcal{L}_{g_3} + \textstyle\sum_i^n \lambda_i\mathcal{L}_{BC,i}. \tag{6.10}$$



(a) The computed loss throughout the iterations.

(b) The displacement output after training.

Figure 6.8: The output of the PINN model after training it for 50000 iterations using the loss function described in Eq. 6.24. The relative error is 1.85% with respect to the FEM solution.

[Layers] x [Neurons per layer]	Relative error after training [%]
2 x 16	2.1
4 x 8	1.6
8 x 4	100.0

Table 6.3: Three different configurations of hidden layers compared.

The results shown in Figure 6.8 is the output of utilizing the non-dimensionalized virtual work equations and the new loss function as described in (6.10), using the same PINN model structure as described for the previous example, with all $\lambda_j = 1$. The remaining hyperparameters are the same.

The influence of the PINN's width and depth has been investigated. In Table 6.3 and Figure 6.9, the results of three different configurations of hidden layers are shown. The 8x4 configuration is too shallow and as a result lacks the capability to capture the non-linear behavior of the beam.

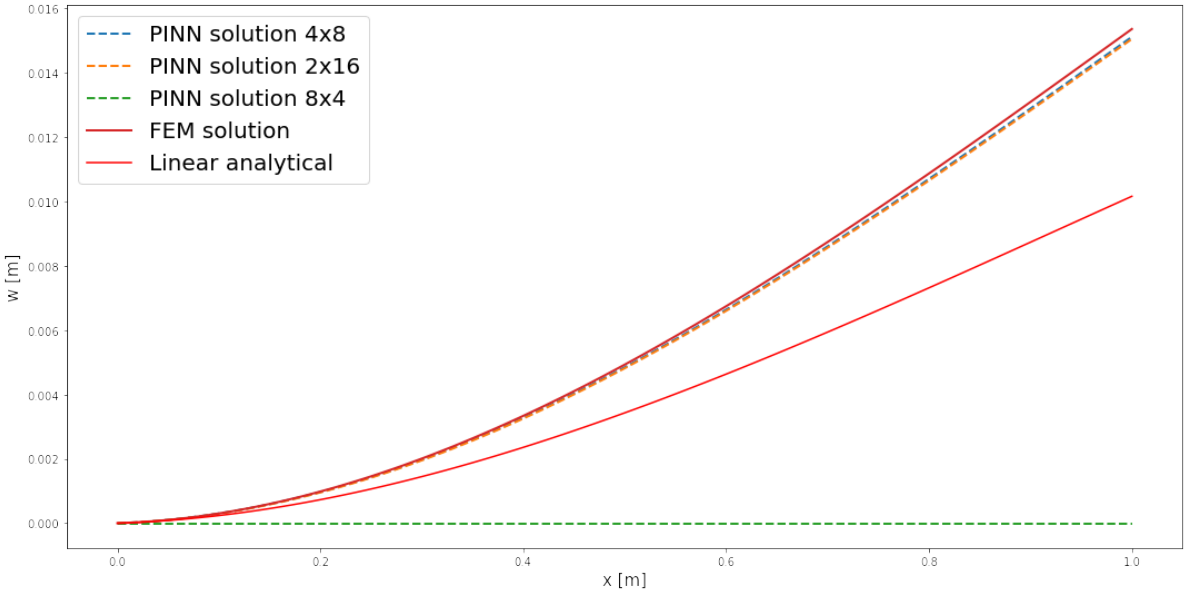


Figure 6.9: The comparison of different hidden layers' configuration with respect to the FEM solution.

7

Stabilizing Techniques

Significant research has been conducted on understanding training pathologies and developing algorithms that improve training efficiency with the goal of reducing computational cost. Among these are the studies performed by Wang et. al (2021) and Jagtap et. al (2020). Wang et. al (2021) developed an algorithm which scales the weights λ_i during training enhancing its effectiveness removing the need to manually find the optimal values of these weights by trial and error. Jagtap et. al (2020) improved the convergence rate by introducing a learnable hyperparameter into the activation function which, like the weights and biases, also gets optimized during training. The influence of these two algorithms on the training of the PINN model will be investigated. A manual seed will be used in PyTorch to keep the initialization of the weights and biases fixed.

7.1. Adaptive Weight Loss Algorithm

Wang et. al (2021) introduced this algorithm due to the impracticality of having to manually adjust the weights λ_i. The goal is to assign suitable weights such that the gradients of each loss term in the loss function during backpropagation are of similar magnitude (Wang et. al, 2021). Here, only the data-fit terms (in this case the boundary conditions) are assigned the weight factors. Loss terms that are PDE residuals are not considered to have weight factors since it is assumed that their gradients are more dominant than those of the data-fit terms. As such, the loss function in our context would then be:

$$\mathcal{L} = \mathcal{L}_{W_{\delta u_o}} + \mathcal{L}_{W_{\delta w}} + \mathcal{L}_{W_{\delta \phi}} + \mathcal{L}_{g_1} + \mathcal{L}_{g_2} + \mathcal{L}_{g_3} + \sum_i^n \lambda_i \mathcal{L}_{BC,i}.$$

Furthermore:

$$\widetilde{\lambda}_{i,t} = \frac{\max\left(\left|\nabla \mathcal{L}_{W_{\delta u_o,t}}\right|, \left|\nabla \mathcal{L}_{W_{\delta w,t}}\right|, \left|\nabla \mathcal{L}_{W_{\delta \phi,t}}\right|, \left|\nabla \mathcal{L}_{g_1,t}\right|, \left|\nabla \mathcal{L}_{g_2,t}\right|, \left|\nabla \mathcal{L}_{g_3,t}\right|\right)}{\overline{\left|\nabla \lambda_{i,t} \mathcal{L}_{BC,i,t}\right|}}.$$

The weights are then updated with:

$$\lambda_{i,t+1} = (1-\alpha)\,\lambda_{i,t} + \alpha \widetilde{\lambda}_{i,t}.$$

Here $\lambda_{i,t}$ denotes the weight factor at training iteration t. α is a hyperparameter which is recommended to be set at 0.1 (Wang et. al, 2021). The PINN model has been trained with the Adam optimizer using the adaptive weight loss algorithm for 50000 iterations, starting with unity λ for all boundary terms. The weights are updated every 500^{th} iteration. The history of the weights during training are shown in Figure 7.1.

Other optimizers are investigated as well, and the question arises how each performs as each possesses its own dynamics regarding convergence. The RMSprop and Adagrad optimizers are investigated as well. The results are summarized in Table 7.1.

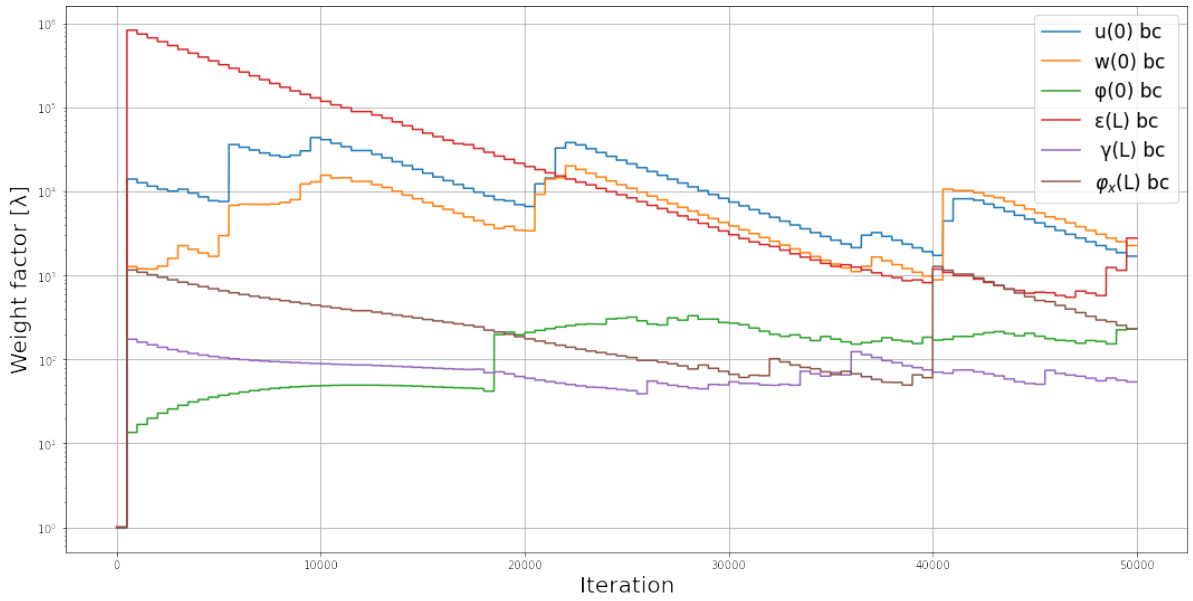


Figure 7.1: The history of the weight factors for each boundary term during training which were updated every 500^{th} iteration using the Adam optimizer.

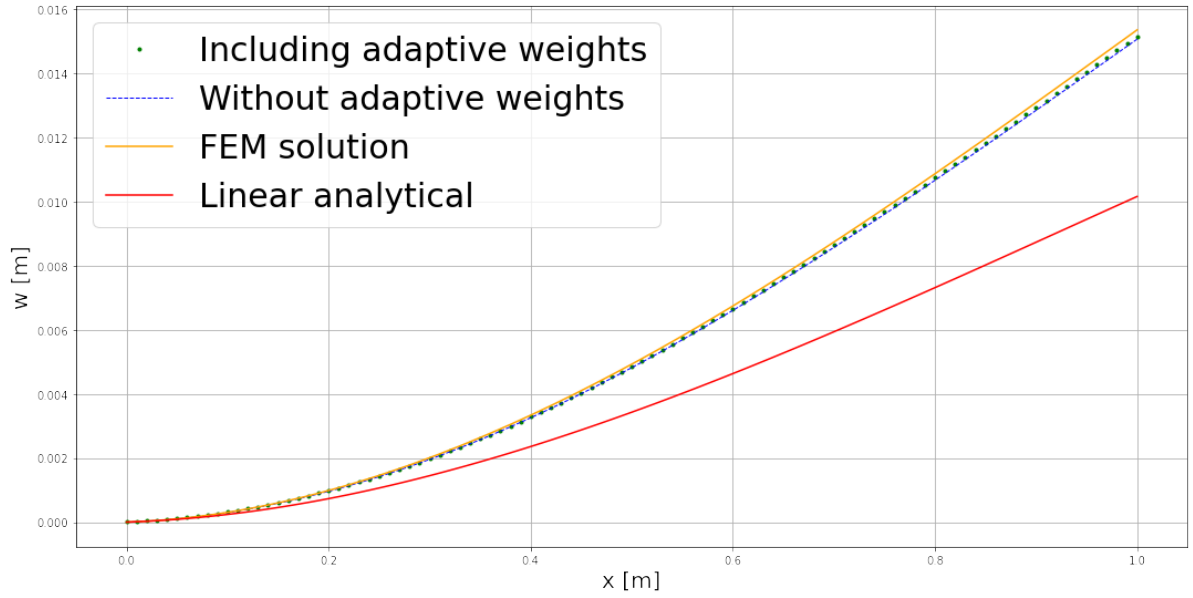


Figure 7.2: The output after training the model with and without the adaptive weight loss algorithm.

Optimizer	**Relative error [%]**
Adam without adapative weights	1.93
Adam with adapative weights	1.22
RMSprop without adaptive weights	28.89
RMSprop with adaptive weights	2.82
Adagrad without adaptive weights	395.93
Adagrad with adaptive weights	153.45

Table 7.1: Different optimizers utilizing the adaptive weight loss algorithm compared. The weights are updated every 500^{th} iteration. The learning rate is set at 0.001.

The inclusion of the adaptive weight loss algorithm resulted in a similar, if not slightly better, output after training with the Adam optimizer. This may be attributed to the fact that the Adam optimizer can handle continuously varying gradients well, due to the fact that it normalizes the updates using the mean and variance of the gradients (also called the first and second moment, respectively). RMSprop also handles gradients similar to Adam. It does not keep track of the first moment, however, unlike Adam.

Adagrad is significantly less accurate, and this can be attributed to the way it handles past gradients. Adagrad accumulates the sum of the squares of all past gradients for each parameter and adjusts its learning rate based on these accumulated values. As a result, parameters associated with large gradients over time experience a significant decrease in their learning rates, leading to slower updates.

7.2. Adaptive Activation Functions

Jagtap et. al (2020) proposed the notion of the activation function containing a learnable parameter, like the weights and biases of the model, to accelerate convergence. The motivation for this is that backpropagation and so also the vanishing and exploding gradient phenomenon are dependent on the activation function. Finding and utilizing an optimal activation function therefore avoids these pathologies and accelerates convergence.

Some common activation functions would then be expressed as follows:

$$\begin{aligned}
&\text{Hyperbolic tangent} : \frac{e^{ax} - e^{-ax}}{e^{ax} + e^{-ax}}, \\
&\text{Sigmoid} : \frac{1}{1 + e^{-ax}}, \\
&\text{ReLU} : \max(0, ax),
\end{aligned}$$

where a is the learnable parameter which continuously adapts during training of the model. Jagtap et. al (2020) also proposed adding a fixed scaling factor n meant to enhance the effects of the parameter a. The aforementioned activation functions would then be described as:

$$\begin{aligned}
&\text{Hyperbolic tangent} : \frac{e^{nax} - e^{-nax}}{e^{nax} + e^{-nax}}, \\
&\text{Sigmoid} : \frac{1}{1 + e^{-nax}}, \\
&\text{ReLU} : \max(0, nax).
\end{aligned}$$

The PINN model has been investigated with the adaptive activation function utilizing the hyperbolic tangent using $n = 1$ and $n = 10$ and are compared with the model with the standard fixed counterpart. The results of the output are shown in Figure 7.3.

The output after training results in an almost identical accuracy for all three investigated combinations. The difference is in the convergence rate as shown in Figure 7.4. Using $n = 10$ results in the model achieving a significantly improved convergence rate, while with $n = 1$ the convergence rate is similar to using the fixed activation function. In Figure 7.5, the history of the parameter a is shown during training.

From Figure 7.6 it is evident that the adaptive activation function is converging to a hyperbolic tangent function with larger gradients around the origin for $n = 10$. The product na has a final value of 9.09 after training has ended which results in the aforementioned shape of the function, whereas with $n = 1$ this product is equal to 1.239.

The steepness of the activation function has an effect on the gradient dynamics. A steeper activation function ($na > 1$) correlates to higher gradients which can accelerate convergence, but may also increase the risk of encountering exploding gradients. In the first few thousand iterations the convergence is significantly superior for $n = 10$ and this can be traced to the activation function which is steeper than the initial function during this same iteration period.

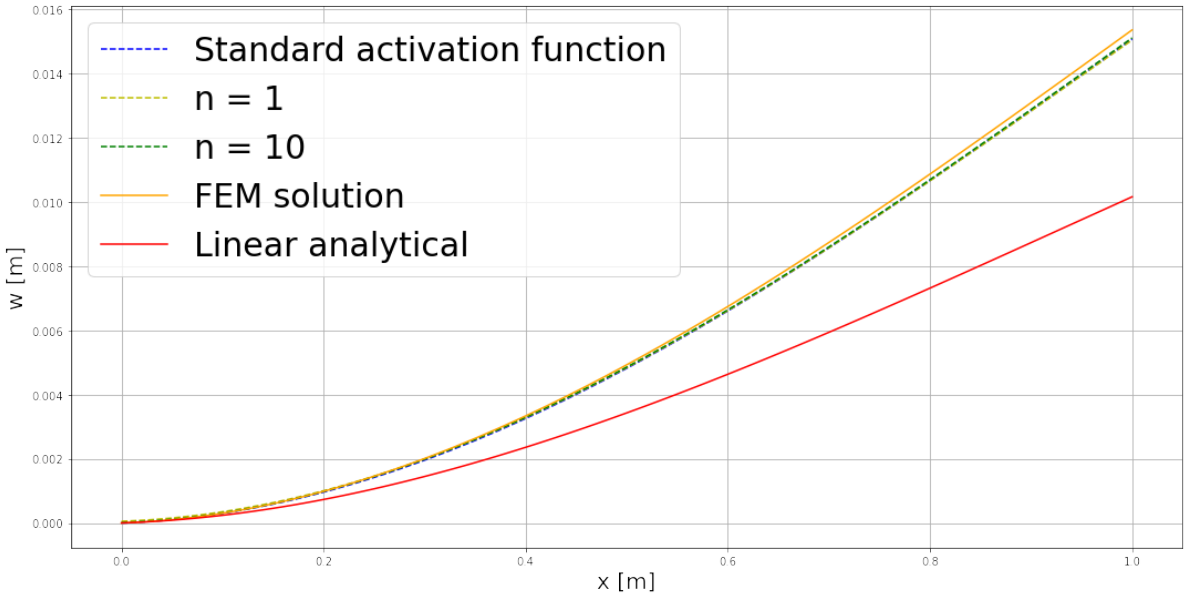


Figure 7.3: The results of using the adaptive hyperbolic tangent function after training the model for 50000 iterations.

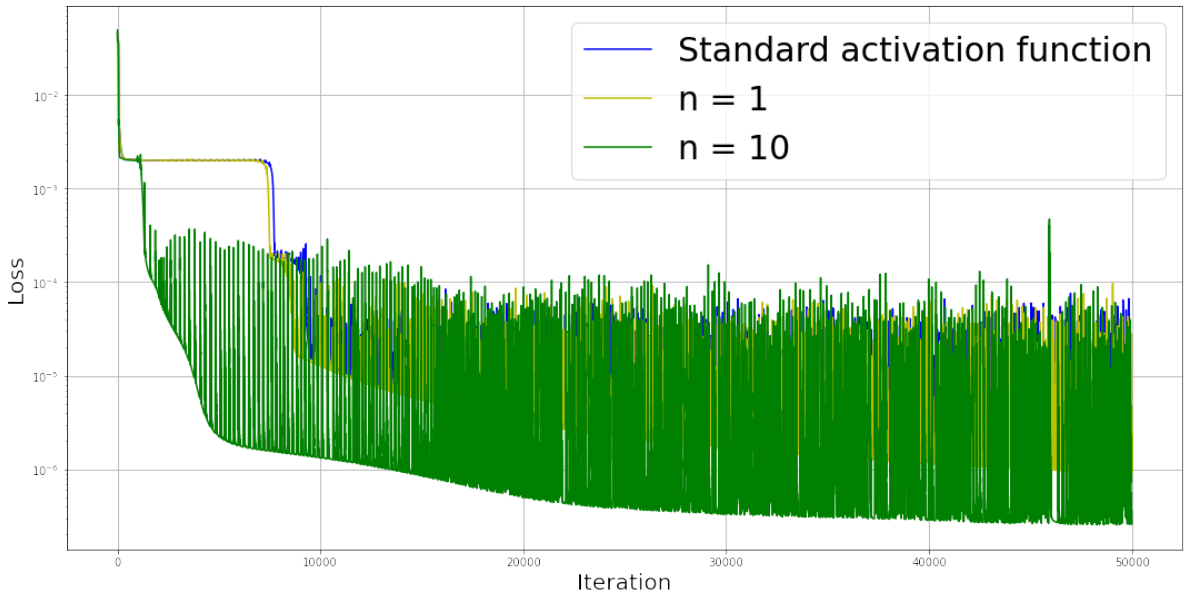


Figure 7.4: The loss history for the different adaptive hyperbolic tangent functions.

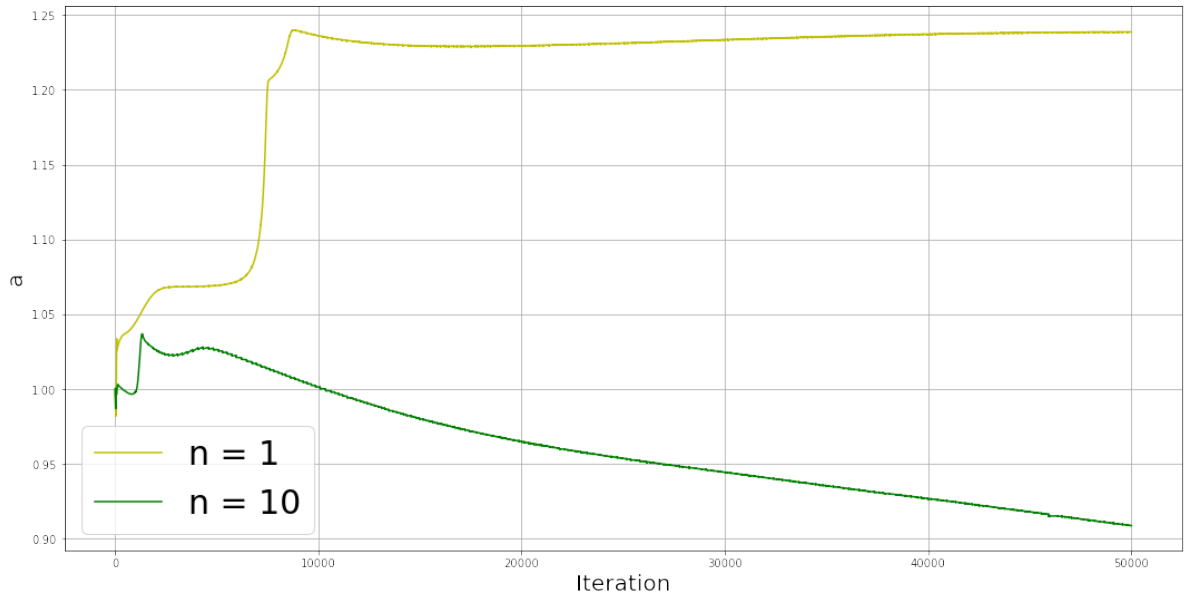


Figure 7.5: The history of the parameter a during training with the adaptive hyperbolic tangent function.

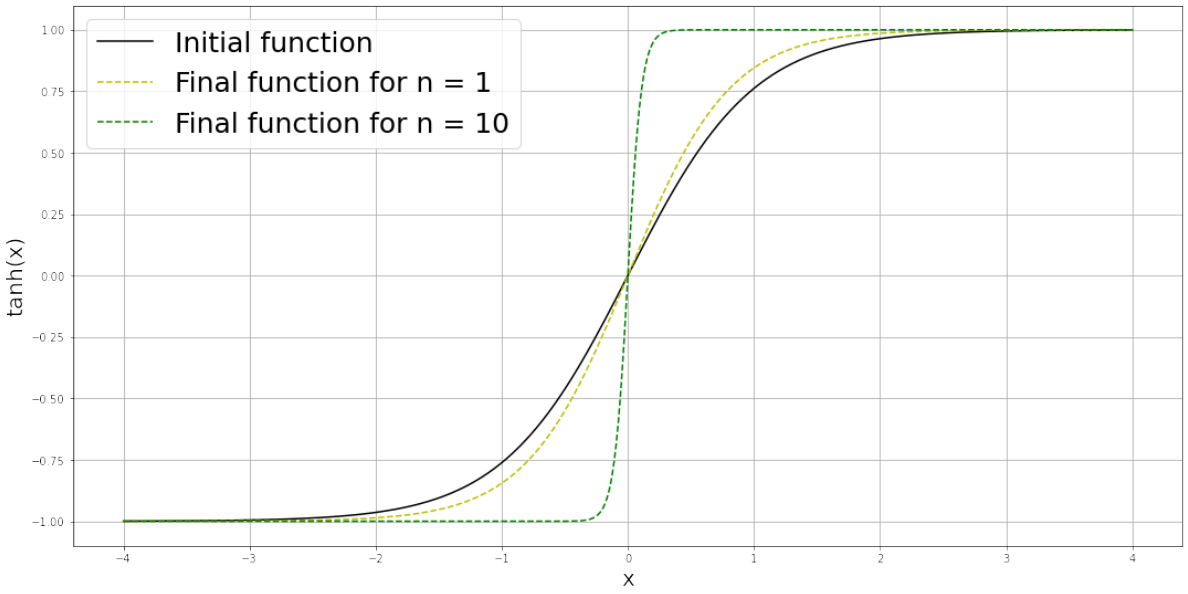


Figure 7.6: The initial hyperbolic tangent function before and after training for n = 1 and n = 10.

8

Parameter Study

In this chapter PINN parameters are investigated that could influence the performance of the geometrically non-linear Timoshenko model. The parameters to be investigated are the activation function and the learning rate. The influence of the amount of hidden layers has already been explored in chapter six.

8.1. Influence of the Activation Function

The activation function is responsible for introducing non-linearity in the model as previously mentioned, and thus directly affects the performance of the model. Several activation functions are investigated using a model containing two hidden layers, containing 16 neurons each. The results are shown in Table 8.1 and the results are visualized in Figure 8.1 .

The ReLU activation function is unable to capture the behavior of the geometrically non-linear Timoshenko beam. This can be attributed to the fact that the second derivative of the ReLU is zero. The computation of the loss residuals becomes saturated. This is further evidenced by the Softplus activation function, which is a logarithmic smooth approximation to the ReLU function. With the Softplus activation function the model is able to accurately predict the solution. A vital part of the model therefore is to select an activation function which has sufficient non-zero derivatives, as the ReLU can become incapable of accurately capturing the behavior (Leng & Thiyagalingam, 2022). This issue can be practically accounted for by selecting an activation function which has infinite non-zero derivatives, such as the Tanh or Sigmoid functions.

8.2. Influence of the Learning Rate

Three different values of the learning rate are investigated where the training of the model ends once a loss of 10^{-5} or lower has been achieved. The results are shown in Table 8.2.

A learning rate of 0.001 turns out to be the most efficient learning rate for this scenario. A learning rate of 0.0001 is too slow to efficiently train the model, whereas a learning rate of 0.01 is too coarse. It is worth noting that the latter is the most efficient learning rate when describing linear problems, as shown in chapter five. For the geometrically non-linear Timoshenko beam problem the most efficient learning

Activation function	**Relative error after training [%]**
Tanh	2.10
Sigmoid	1.65
ReLU	411.20
Softplus	2.20

Table 8.1: Performance of the model using different activation functions.

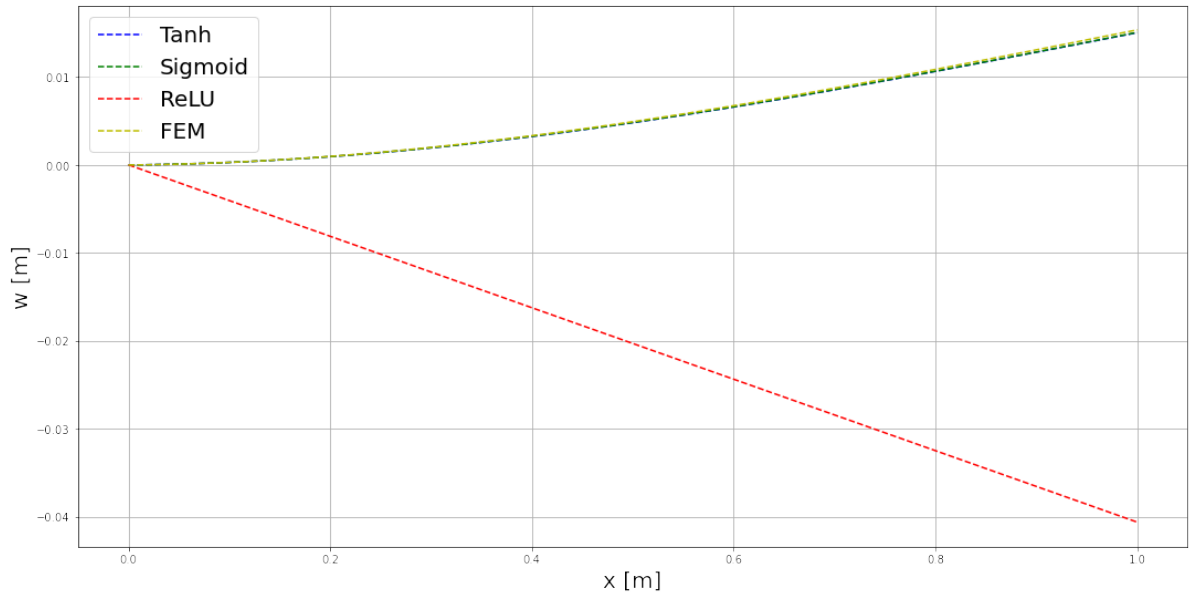


Figure 8.1: Different activation functions compared.

Learning Rate	Iterations
0.01	11033
0.001	9005
0.0001	17553

Table 8.2: The amount of iterations necessary for reaching a loss of $\mathbf{10^{-5}}$ or lower for each learning rate.

rate is around the magnitude of 10^{-3}. This can be attributed to the increased complexity of the model and also the choice of residuals. The latter of which will be discussed in the next chapter.

9 Discussion

In this section, some practical aspects of PINNs will be discussed, such as the limitations of it faced in this thesis and in general, its potential use case in monitoring and higher dimensional implementation.

9.1. Challenges

The one-dimensional geometrically non-linear Timoshenko beam is investigated where the loss term contains a virtual work residual and a force residual. A disadvantage that has been experienced is that the computational expense, for solving the proposed problem, is significantly higher than the FEM, which corresponds to the findings of Grossman et al. (2024). There is evidence that suggest that the PINN method, however, experiences no exponential increase in computational cost if the dimensionality increases when solving for PDEs, as evidenced by Hu et al. (2024), possibly indicating that the PINN method will be more computationally efficient when solving for high dimensional cases.

Another challenge is training stability. There have been techniques derived to tackle this issue, such as non-dimensionalization, which has been applied here with significant success as a result. Proper initialization of the PINN model also affects the training greatly. It is common practice to initialize the model using a Xavier initialization method, as described in chapter six, to avoid the phenomena of vanishing and exploding gradients so as to stabilize training. It is found that using an uniform distribution initialization where the weights and biases are close to zero, as used previously, results in superior convergence than with the Xavier initialization. Based on this, for geometrically non-linear problems using the same residuals as presented here, it is advised to consider using the aforementioned initialization instead of the Xavier initialization.

Furthermore, for each model solving a problem the optimal hyperparameters (hidden layers, amount of layers, activation function, etc.) have to be found using trial-and-error. This is only an issue at the beginning, when building the model. There is literature available that can serve as guidelines to mitigate this issue, such as the work of Wang et al. (2023).

In this thesis the beam is assumed to be perfectly elastic throughout loading. It is shown here that PINNs are able to accurately capture the non-linear behavior of beams. This is because the relevant physics are implemented in the model, meaning the model can capture only what is defined a priori. The model quickly becomes questionable when it only relies on physics which are not understood well. This is another limitation of PINNs. For instance, the assumed dynamics of a structural system may change due to environmental conditions which is not or cannot be accounted for in the PINN model.

In real-world contexts, PINNs can integrate sensor data of the investigated system to compensate for dynamics that are erroneously assumed or simplified to still arrive at an accurate prediction, which ties back to how PINNs can be used for structural health monitoring as well.

9.2. PINNs and Monitoring

In this thesis, the proposed model has been verified with simulated data. This leads to the question on how PINNs can be utilized when real-life (measured) data is present and what role PINNs can have depending on the presence of simulated and real-life data or the lack thereof. In this thesis a forward problem has been addressed, but PINNs can also be used to solve inverse problems. Some examples regarding this will follow.

When investigating a system, PINNs can be trained using real-life data while being constrained to physical laws, which is how data scarcity can be compensated for. For algorithms that do not use any physical laws, distinguishing between alterations observed in a system due to impairment and those caused by changing operational or environmental conditions is challenging (Figueiredo and Brownjohn, 2022). With PINNs, the promise is that this can be extrapolated by utilizing physical laws, thus improving generalization.

Lai et al. (2022) presented a monitoring framework where physics-informed modelling is integrated. A cable-stayed bridge, which is considered here to be linear to mildly non-linear, is investigated using an experimental dataset. It is shown that the hybrid model, the model that utilizes both PINNs and experimental data, outperforms the purely physics-based approach. A limitation they noted is that the performance deteriorates when the dynamic regime deviated significantly from the training data (Lia et al., 2022).

Cross et al. (2021) demonstrated this promise of generalization of PINNs and stated that it is especially useful when data is scarce, which is common in structural monitoring projects. They further commented that the methodology would benefit of a better understanding of fatigue damage accrual.

To date, research on PINNs and structural health monitoring has primarily concentrated on modeling and predicting dynamic behavior, while further studies are needed to assess their effectiveness in quasi-static behavior, which is vital for long-term structural monitoring (Al-Adly & Kripakaran, 2024). To that end, Al-Adly & Kripakaran (2024) investigated a Kirchhoff-Love plate using PINNs and synthetic sensor data including simulated noise. They investigated PINN models, predicting displacements and internal forces, that included PDE-constraints related to physical laws and those without. They concluded that the PINN model that only included the boundary condition constraints and the sensor data outperformed the one that also included the PDE-constraint at relatively low noise levels. This is noteworthy since it implies using the former model is better for generalization while also being less computationally expensive. It is shown from the results that the model with the PDE-constraint included performs better when higher noise levels are present, though, indicating that utilizing PDE-constraints results in an overall more consistent model when high noise levels are present. It is also concluded that the PINNs also are able to accurately predict for inputs outside the range of training data.

Based on the mentioned studies, a schematic illustration is shown in Figure 9.1 used to visualize the role pure data-driven approaches and physics-informed methods can have within structural health monitoring.

9.3. Complex Geometries

Despite PINNs' successful applications, a limitation so far is the application of it on complex geometries. Multiple studies have been conducted on implementing strategies so that PINNs can manage more complex geometries where traditional PINNs, here defined as PINNs utilizing a simple distribution of the collocation points, fail to make accurately predictions. Sukumar & Srivastava (2022) constructed an approach using a signed distance function to describe the domain. By utilizing this, the boundary conditions are satisfied a priori, without utilizing separate loss terms for this. It is concluded that the proposed PINN model consistently outperforms the traditional collocation method (which is used in this thesis), highlighting the importance of satisfying the boundary conditions a priori. The numerical examples presented are mainly focused on one- and two-dimensional shapes.

Costabal et al. (2024) focused on an approach using the Laplace–Beltrami operator, which is used to encode the geometrical information, to create an input space that accurately represents the geometry

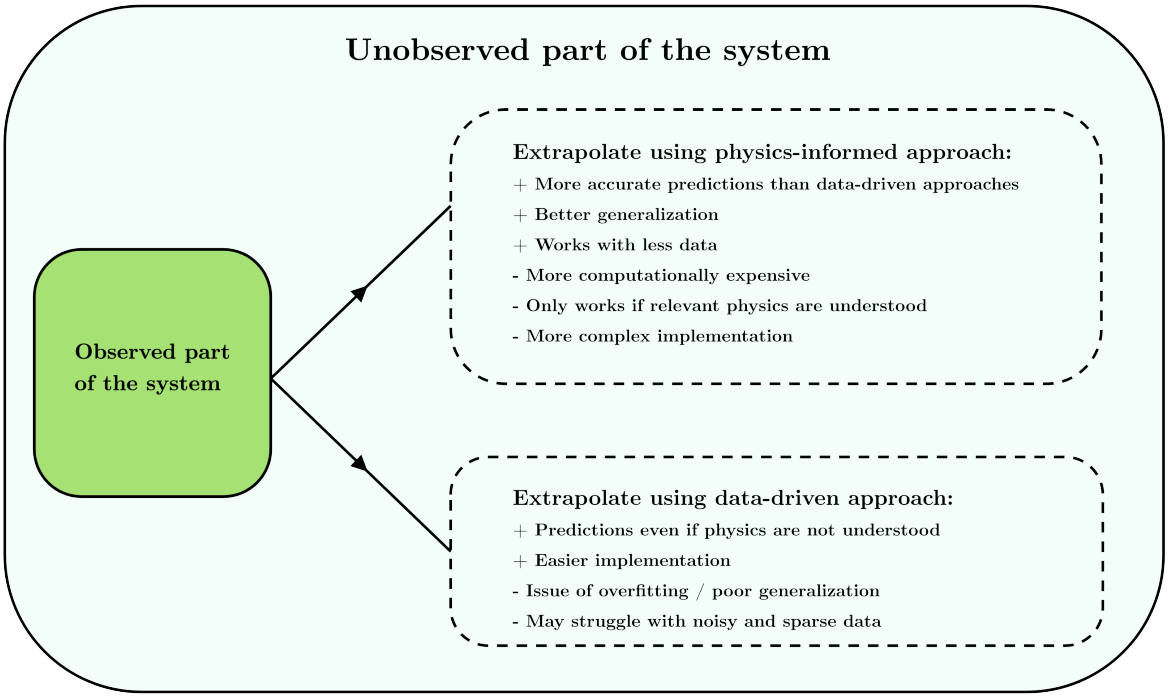


Figure 9.1: The methods used to extrapolate observed data to regions which are unobserved. The advantages and disadvantages of using a purely data-driven approach and a physics informed approach when extrapolating from observed data of a system.

discussed. The input vector is, contrary to traditional PINNs, no longer a set of coordinates where the residuals are calculated, but rather the value of eigenfunctions of the Laplace-Beltrami operator at those coordinates with the notion being that more information can be encoded for each point coordinate. With this, complex fully three-dimensional surfaces have been investigated such as a heat sink and a Stanford bunny among others. It is shown that this proposed PINN model can accurately predict the solution whereas the traditional PINN fails to do so.

These examples show that while traditional PINNs may fail with complex geometries, recent advancements enable the accurate modeling of complex shapes using techniques like those previously stated. Depending on the geometry, a simple equidistant grid for the collocation points or an adaptively distributed one as mentioned in chapter three can suffice for relatively simple shapes, while more complex geometry (three-dimensional surfaces) may require more complex processes such as the positional encoding method proposed by Costabal et al. (2024) or the signed distance function proposed by Sukumar & Srivastava (2022).

10

Conclusions and Recommendations

In this chapter, the main findings of the research will be summarized and some recommendations will be made based on this study. In this thesis the viability of using a physics informed neural network to approximate a geometric non-linear Timoshenko beam solution is investigated.

For solving linear beam cases, it suffices to include the differential equation(s) describing the utilized beam theory as a loss term while the remainder loss terms are the boundary conditions of the described problem. It should be noted that the differential equation(s) should be non-dimensionalized to significantly improve the convergence rate.

For the geometrically non-linear case a variational principle, the principle of virtual work, is used in conjunction with the Green-Lagrange strain theory to formulate the geometric non-linear Timoshenko virtual work equations. These virtual work equations form a part of the loss function, but do not suffice on their own to solve the problem, as the model tends to the trivial solution. Two penalty terms are introduced and investigated to alleviate this issue. The first penalty term directly penalizes the model for remaining at the trivial solution which is multiplied by a scaling factor. The issue that arises is the trial-and-error nature that is necessary to find the optimal value of the scaling factor such that it does not dominate the loss function. The second penalty term is a force residual, which is more robust, as there is no trial-and-error method for this penalty term necessary to guarantee convergence.

The PINN model has a single neuron as input, this being the collocation points along the beam. The model has three output neurons, which correspond to the degrees of freedom the Timoshenko beam theory describes, which are the horizontal and vertical translations and the cross-sectional rotation. The amount of hidden layers has not only an influence on the computational cost, but also on the accuracy the model can achieve. A model containing four hidden layers with as little as eight neurons per hidden layer achieves a relative error of sub 2% with respect to the FEM solution.

Two additional algorithms which improve the convergence rate, the adaptive weight loss algorithm and the adaptive activation function, have been investigated. Both algorithms positively influence the convergence rate of the model, though the former significantly increases the computational cost of the training. Based on this study, a non-dimensionalized model coupled with the Adam optimizer is recommended to use which can eventually be supplemented with an adaptive activation function. The adaptive activation function with $n = 10$ can be incorporated into the model to improve the convergence rate significantly without significantly increasing the computational cost of the model.

While this research focused on one-dimensional geometric non-linear Timoshenko beams, future studies could extend the PINN framework to incorporate material non-linearity. This would allow for a more comprehensive model where material behavior is not purely elastic. Future studies could also include extending the framework to model geometrically non-linear plates and frame structures.

References

Al-Adly, A. I., & Kripakaran, P. (2024). Physics-informed neural networks for structural health monitoring: a case study for Kirchhoff–Love plates. *Data-Centric Engineering*, *5*, e6.

Alam, M. I., Kanagarajan, B., & Jana, P. (2019). Optimal design of thin-walled open cross-section column for maximum buckling load. Thin-Walled Structures, 141, 423-434.

Arridge, S., Maass, P., Öktem, O., & Schönlieb, C. B. (2019). Solving inverse problems using data-driven models. Acta Numerica, 28, 1-174.

Bauchau, O. A., & Craig, J. I. (2009). Euler-Bernoulli beam theory. In Structural analysis (pp. 173-221). Dordrecht: Springer Netherlands.

Bazmara, M., Mianroodi, M., & Silani, M. (2023). Application of Physics-informed neural networks for nonlinear buckling analysis of beams. Acta Mechanica Sinica, 39(6), 422438.

Chen, L., Zhang, H. Y., Liu, S. W., & Chan, S. L. (2023). Second-order analysis of beam-columns by machine learning-based structural analysis through physics-informed neural networks. *Advanced Steel Construction*, *19*(4), 411-420.

Costabal, F. S., Pezzuto, S., & Perdikaris, P. (2024). Δ-PINNs: physics-informed neural networks on complex geometries. *Engineering Applications of Artificial Intelligence*, *127*, 107324.

Cross, E. J., Gibson, S. J., Jones, M. R., Pitchforth, D. J., Zhang, S., & Rogers, T. J. (2022). Physics-informed machine learning for structural health monitoring. *Structural Health Monitoring Based on Data Science Techniques*, 347-367.

DIANA FEA B.V. (2021). DIANA FEA (10.5) [Software]. `https://dianafea.com/`

Figueiredo, E., & Brownjohn, J. (2022). Three decades of statistical pattern recognition paradigm for SHM of bridges. *Structural Health Monitoring*, *21*(6), 3018-3054.

Gardner, L., & Nethercot, D. A. (2011). *Designers' guide to Eurocode 3: design of steel buildings* (p. 7). ICE publishing.

Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249-256). JMLR Workshop and Conference Proceedings.

Grossmann, T. G., Komorowska, U. J., Latz, J., & Schönlieb, C. B. (2024). Can physics-informed neural networks beat the finite element method?. *IMA Journal of Applied Mathematics*, hxae011.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359-366.

Hu, Z., Shukla, K., Karniadakis, G. E., & Kawaguchi, K. (2024). Tackling the curse of dimensionality with physics-informed neural networks. *Neural Networks*, *176*, 106369.

Jagtap, A. D., Kawaguchi, K., & Karniadakis, G. E. (2020). Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, *404*, 109136.

Kapoor, T., Wang, H., Núñez, A., & Dollevoet, R. (2023). Physics-informed neural networks for solving forward and inverse problems in complex beam systems. IEEE Transactions on Neural Networks and Learning Systems.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Khodabakhshi, P., & Reddy, J. N. (2017). A unified beam theory with strain gradient effect and the von Kármán nonlinearity. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, *97*(1), 70-91.

Lai, Z., Liu, W., Jian, X., Bacsa, K., Sun, L., & Chatzi, E. (2022). Neural modal ordinary differential equations: Integrating physics-based modeling with neural ordinary differential equations for modeling high-dimensional monitored structures. *Data-Centric Engineering*, *3*, e34.

Lawal, Z. K., Yassin, H., Lai, D. T. C., & Che Idris, A. (2022). Physics-informed neural network (PINN) evolution and beyond: a systematic literature review and bibliometric analysis. Big Data and Cognitive Computing, 6(4), 140.

Leiteritz, R., & Pflüger, D. (2021). How to avoid trivial solutions in physics-informed neural networks. arXiv preprint arXiv:2112.05620.

Leng, K., & Thiyagalingam, J. (2022). On the compatibility between neural networks and partial differential equations for physics-informed learning. *Available at SSRN 4392241*.

Moradi, S., Duran, B., Eftekhar Azam, S., & Mofid, M. (2023). Novel physics-informed artificial neural network architectures for system and input identification of structural dynamics PDEs. *Buildings*, *13*(3), 650.

Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). 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. `https://doi.org/10.1016/j.jcp.2018.10.045`

Ruocco, E., Wang, C., Zhang, H., & Challamel, N. (2017). An approximate model for optimizing Bernoulli columns against buckling. Engineering Structures, 141, 316–327. `https://doi.org/10.1016/j.engstruct.2017.01.077`

Sukumar, N., & Srivastava, A. (2022). Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, *389*, 114333.

Talebi, H., Saputra, A., & Song, C. (2016). Stress analysis of 3D complex geometries using the scaled boundary polyhedral finite elements. Computational Mechanics, 58(4), 697–715. `https://doi.org/10.1007/s00466-016-1312-0`

Timoshenko, S. P. (1921). LXVI. On the correction for shear of the differential equation for transverse vibrations of prismatic bars. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *41*(245), 744-746.

Tsavdaridis, K. D., Kingman, J. J., & Toropov, V. V. (2015). Application of structural topology optimisation to perforated steel beams. Computers & structures, 158, 108-123.

Wang, S., Sankaran, S., Wang, H., & Perdikaris, P. (2023). An expert's guide to training physics-informed neural networks. *arXiv preprint arXiv:2308.08468*.

Wang, S., Teng, Y., & Perdikaris, P. (2021). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, *43*(5), A3055-A3081.

Weihe, S., Kröplin, B., & De Borst, R. (1998). Classification of smeared crack models based on material and structural properties. *International journal of solids and structures*, *35*(12), 1289-1308.