Ole Kristian Holme

Improving Vortex-Induced Vibration Prediction Using Hybrid Analytic

Master’s thesis in Marine Technology
Supervisor: Dong Trong Nguyen
Co-supervisor: Svein Sævik, Jie Wu
June 2023

Ole Kristian Holme

Improving Vortex-Induced Vibration Prediction Using Hybrid Analytic

Master’s thesis in Marine Technology
Supervisor: Dong Trong Nguyen
Co-supervisor: Svein Sævik, Jie Wu
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

MSC THESIS DESCRIPTION

Name of the candidate:	Ole Kristian Holme
Field of study:	Marine Cybernetics
Thesis title (Norwegian):	Forbedre virvel indusert vibrasjon (VIV) prediksjon, kombinere fysisk-basert og maskin læring metoder.
Thesis title (English):	Improve vortex-induced vibrations (VIV) prediction by combining physics-based model and machine learning methods.

Background
Vortex-Induced Vibrations (VIV) can lead to fast accumulation of fatigue damage to offshore slender structures, such as power cables to offshore floating turbines and offshore marine risers. A new empirical method for time domain (TD) calculation of VIV has been developed by NTNU. This method is capable of accounting for structural non-linearity and time-varying flow compared to the traditional frequency domain analyses. However, the mathematic formulation of the load model is still considered simplified compared to the complex underlying physical process. By gradually introducing machine learning-based terms in the existing model and continually assessing performance, the already achieved understanding of the underlying model will be preserved, while the accuracy and precision increase. The objective of this study is to improve VIV response prediction by investigating a hybrid model, which combines physics-based prediction model and data-drive machine learning methods.

Work description
1. Perform a background and literature review to provide information and relevant references on:

- Offshore marine risers with a focus on the identification of relevant failure modes and monitoring techniques
- Power cable for floating wind turbines with a focus on the identification of relevant failure modes and monitoring techniques
- Fundamental theory of VIV and prediction tool in time domain (VIVANA-TD)
- Appropriate machine learning methods
- Methods for hybrid modeling
- Digital twins

2. Investigate a Physics-Informed Neural Network (PINN) on a constructed problem, exploring:

- Forward-solving capabilities
- Inverse-solving capabilities
- Improving simplified model with measurement data

3. Utilize PINN with hydrodynamic force from SIMA to examine:

- Forward-solving capabilities
- Inverse-solving capabilities
- Improving simplified model with measurement data

4. Use PINN with the VIVANA-TD equation, incorporating parts of the force from SIMA:

- Forward-solving capabilities
- Inverse-solving capabilities
- Improving simplified model with measurement data
- Extend to include to beam model in the PINN (tentative)

5. Summary and recommendation for future work

Specifications
The scope of work may prove to be larger than initially anticipated. By the approval from the supervisor, described topics may be deleted or reduced in extent without consequences with regard to grading.

The candidate shall present personal contribution to the resolution of problems within the scope of work. Theories and conclusions should be based on mathematical derivations and logic reasoning identifying the various steps in the deduction.

The report shall be organized in a logical structure to give a clear exposition of background, results, assessments, and conclusions. The text should be brief and to the point, with a clear language. Rigorous mathematical deductions and illustrating figures are preferred over lengthy textual descriptions. The report shall have font size 11 pts., and it is not expected to be longer than 60-80 A4 pages, from introduction to conclusion, unless otherwise agreed upon. It shall be written in English (preferably US) and contain the following elements: Title page, abstract, acknowledgements, thesis specification, list of symbols and acronyms, table of contents, introduction with objective, background, and scope and delimitations, main body with problem formulations, derivations/developments and results, conclusions with recommendations for further work, references, and optional appendices. All figures, tables, and equations shall be numerated. The original contribution of the candidate and material taken from other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a Harvard citation style (e.g. *natbib* Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct. Such practice is taken very seriously by the university and will have consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed upon.

The thesis shall be submitted with a printed and electronic copy to the main supervisor, with the printed copy signed by the candidate. The final revised version of this thesis description must be included. The report must be submitted according to NTNU procedures. Computer code, pictures, videos, data series, and a PDF version of the report shall be included electronically with all submitted versions.

Start date:	9 January 2023	**Due date:**	11 June 2023

Supervisor:	Dong Trong Nguyen
Co-advisor(s):	Svein Sævik, Jie Wu

Trondheim 09.06.2023, Ole Kristian Holme

[illegible]
Digitally signed by
Dong Trong Nguyen
Date: 2023.06.09
15:47:08 +02'00'

Dong Trong Nguyen
Supervisor

Preface

Master's thesis in Marine Technology, spring 2023. The thesis explores the application of machine learning methods to improve the vortex-induced vibration prediction model VIVANA-TD. Specifically, the thesis will focus on utilizing neural networks and integrating them with prior knowledge to achieve more precise predictions. Various approaches will be discussed, and a seamless method for combining physical models and measurement data will be tested. This thesis builds upon the project thesis conducted in fall 2022, with chapters 1 and 2 serving as an extended version of the project thesis (TMR4510, 7.5 credits).

I express my sincere gratitude to Prof. Dong Trong Nguyen (NTNU), Prof. Svein Sævik (NTNU), and Jie Wu (Sintef) for their guidance and support. Their expertise and assistance have contributed to my understanding, approach, and motivation. Additionally, I would like to thank Jie for his assistance with the SIMA simulation software and expertise in vortex-induced vibrations.

"All models are wrong, but some are useful"- George E.P.Box

Ole Kristian Holme

Trondheim 07.06.2023

Abstract

Predictions of vortex-induced vibrations (VIV) are of great importance in design and lifetime calculations for slender structures in the ocean. The semi-empirical models used today are still considered simplified compared to the complex fluid-structure interaction. This thesis investigates the potential of combining machine learning techniques with the semi-empirical time-domain model VIVANA-TD to improve VIV prediction. The thesis begins by presenting the theoretical background of VIV and machine learning. Then, different approaches combining physics-based models with machine learning will be discussed.

Furthermore, the thesis investigates how Physics-informed Neural Networks (PINNs) can improve predictions by incorporating mathematical models and measurement data. PINNs introduce an additional term in the loss function that describes the difference between the predictions of the neural network and the physics-based model. This approach enables the discovery of a solution that aligns with the mathematical model. Moreover, the approach facilitates the adjustment of unknown parameters within the physics-based equations, leading to a model that satisfies both the measurement data and the mathematical model. Two simplified problems are used to evaluate the method.

Results from the simplified problems demonstrate its potential to improve mathematical models by incorporating measurement data. However, the optimization process becomes more complex due to an additional physics-based term in the loss function of the neural network. Especially when the parameters of the mathematical models are large, the model fails to converge to the correct solution. Although modifying the neural network's structure improves its ability to converge to the correct solution, further improvements and understanding are necessary for it to become a practical and effective method in engineering.

Sammendrag

Prediksjoner av virvel-induserte vibrasjoner (VIV) er av stor betydning i design- og levetidsberegninger for slanke strukturer i havet. Modellene brukt i dag, har flere forenklinger som gjør at de ikke klarer å beskrive den komplekse fluid strukturen helt nøyaktig. Oppgaven har som mål å utforske potensialet ved å kombinere maskinlæringsteknikker med tids-domene modellen VIVANA-TD, for å forbedre VIV-prediksjonen. Først vil bakgrunnsteori for VIV og maskinlæring bli presentert. Deretter vil tilnærminger for å kombinere fysikk-baserte modeller med maskinlæring bli diskutert. Spesielt vil bruken av nevrale nettverk kombinert med tidligere kunnskap i form av matematiske modeller bli undersøkt.

Oppgaven utforsker videre metoden fysikk-baserte nevrale nettverk (PINN). PINN er et nevralt nettverk med et ekstra ledd i tapfunksjonen, som beskriver forskjellen på den fysikk-baserte modellen og prediksjonen til det nevrale nettverket. Denne metoden gjør det mulig å tilpasse seg måledata og samtidig som det er mulig å finne ukjente parameter i den fysikk-baserte modellen. Dette gjør at PINN kan benytte seg av matematiske modellene vi har i dag og forbedre dem med måledata. Metoden blir testet på to forenklet fysikk-baserte modeller.

Resultatet for PINN, er at metoden i noen tilfeller forbedrer prediksjonen. Men ved å inkludere et ekstra ledd i tapfunksjonen, blir optimaliseringen mer utfordrende. I noen tilfeller får modellen helt feil resultat. Dette er spesielt når parameterne i den matematiske modellen blir store. Ved å endre på strukturen i nevral nettverket blir metoden mer robust. Videre utbedring og forståelse av hvorfor det i visse tilfeller ikke fungerer, må gjøres for at det skal bli en effektiv metode.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Adam	Adaptive Moment Estimation.
AI	Artificial Intelligence.
BC	Boundary Condition.
CFD	Computational Fluid Dynamics.
DDM	Data-Driven Model.
DNN	Deep Neural Network.
IC	Initial Condition.
LSTM	Long-Short-Term-Memory.
MIT	Massachusetts Institute of Technology.
ML	Machine Learning.
NN	Neural Network.
PBM	Physics-Based Model.
PIML	Physics-Informed Machine Learning.
PINN	Physics-Informed Neural Networks.
SciML	Scientific Machine Learning.
VIM	Vortex-Induced Motions.
VIV	Vortex-Induced Vibrations.

Chapter 1

Introduction

This chapter covers the motivation, problem description, literature study, and thesis structure.

1.1 Motivation

In recent years, there has been a growing interest in utilizing the ocean space for renewable energy. With most of the world's surface area covered in oceans, it holds immense potential for resolving the energy transition by technology such as wind turbines, wave power, and floating solar panels. All these installations require significant power cables and mooring lines at deep sea, which contributes considerably to the development costs (DNV n.d.). Offshore structures must withstand harsh weather conditions, including waves, currents, and strong winds. Therefore, having accurate and precise dynamic response models of ocean cables for practical engineering and operations is crucial. Vortex-induced vibrations (VIV) can occur for slender structures in flowing fluids, leading to fatigue damage. Structural integrity analyses must therefore account for VIV. Today semi-empirical methods are used to calculate VIV fatigue damage. However, the mathematical formulation of the hydrodynamic load model is still considered simplified compared to the complex physical process (Wu et al. 2020). A simplified model leads to uncertainty in response prediction, resulting in conservative fatigue prediction and shorter design life. As cost is one of the limiting factors of offshore renewable energy installation, getting more accurate models will elongate the design life reducing the life cycle cost. This thesis will focus on riser modeling because most VIV research is for risers. The study on risers is also relevant to power cables and other slender flexible structures. Figure 1.1 illustrates a semi-submersible and the riser connecting the wellhead to the platform. The white dotted line represents the oscillatory motion of the riser in the event of VIV.

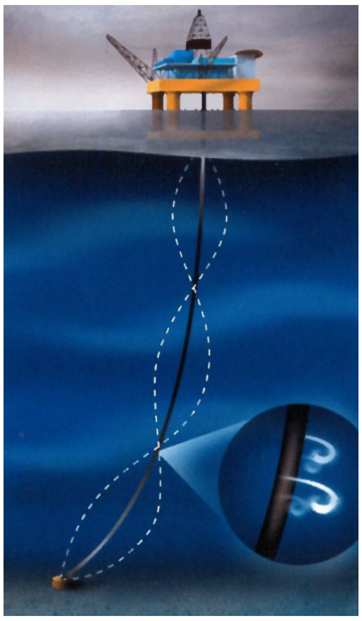


Figure 1.1: Riser and VIV, courtesy Havromsteknologi (Myrhaug and Pettersen 2021)

Machine learning has gained significant attention in recent years due to more available computing power, large amounts of available data, and efficient algorithms. Machine learning has demonstrated impressive performance in various applications, including protein structure prediction (AlQuraishi 2019), image recognition (Szegedy et al. 2014), and self-driving cars. Machine learning excels at capturing the full complexity of a system and effectively representing the systems using deep neural networks. However, while machine learning models excel at capturing the full complexity of a system, they can be limited by the availability of data. The motivation for this thesis is that the abilities of machine learning models should be combined with the prior knowledge from science and engineering such that the advantages of machine learning are utilized while maintaining the generalizability and trustworthiness of the physics models.

1.2 Problem description

Buff bodies exposed to flowing fluid may experience oscillating lift and drag forces. These forces are caused by flow separation and vortex shedding. For flexible structures, the oscillating forces will induce structural vibrations, known as vortex-induced vibrations (DNV-GL 2016a). VIV can rapidly accumulate fatigue damage in slender offshore structures, such as risers, pipelines, mooring lines, and power cables. Fatigue damage poses a significant concern for operations and structural integrity. Therefore, an accurate model is crucial for condition monitoring and can help reduce operating costs while extending the structure's design life. Additionally, vibrations in the cross-flow direction increase drag force, which is particularly relevant during the laying of slender structures on the seabed or in static calculations. Over the past few decades, extensive research has been conducted to improve mathematical models of VIV (DNV-GL 2016a). However, due to the complexity of fluid-structure phenomena, these mathematical models are idealized approximations and cannot fully capture the underlying physics due to various simplifications. This thesis aims to investigate the potential implementation of machine learning in conjunction with the existing mathematical model to achieve more accurate predictions. Furthermore, digital solutions, such as digital twins, will be discussed, where real-time data from sensors is combined with the model to monitor the displacement and integrity of the structure. Lastly, a novel approach will be implemented and tested on simulation data, incorporating prior knowledge directly into the neural network.

1.3 Literature review machine learning implemented in VIV

Some of the most relevant research will be introduced to serve as an inspiration and identify the approaches worth exploring. Three typical types of machine learning (ML) applications in structural engineering are system identification, damage detection, and dynamic response (Sadeghi Eshkevari et al. 2021). While many approaches can be interesting for fatigue damage in risers, such as cameras/sensors detecting structural fatigue, this thesis will focus on improving the dynamic response model.

1.3.1 Machine learning and VIV models

The semi-empirical vortex-induced vibration models usually rely on empirical coefficients to estimate the hydrodynamic forces. The accuracy of these models depends on selecting the empirical coefficients, which necessitates conducting several model tests to cover all possible conditions. The coefficients are determined based on the non-dimensional frequency and amplitude of the response. However, other factors such as the Reynolds number, in-line motions, surface roughness, and inflow conditions also influence the coefficients (Kharazmi, Z. Wang et al. 2021). Considering the expensive and time-consuming nature of model tests, optimizing the selection of these tests is of great interest. Introducing an automatic model tester combined with an artificial intelligence (AI) program could revolutionize the model testing process. This combination would enable the AI to conduct model tests and optimize the setup for subsequent tests. This approach was implemented in the intelligent towing tank by the Massachusetts Institute of Technology (MIT) riser program (Kharazmi, Z. Wang et al. 2021), resulting in a more robust empirical coefficient database. Another approach to improving the coefficients involves using Bayesian machine learning algorithms. Andersen et al. 2022 utilized Bayesian machine learning to identify optimal coefficients based on response measurements. Both approaches aim to estimate improved coefficients, reducing uncertainties associated with hydrodynamic coefficients. However, they still rely on the semi-empirical model, with machine learning used to optimize the parameters. Another approach is integrating the limited sensor data with an existing mathematical model to refine predictions. This approach has been effectively implemented by 4Subsea (Nilsen-Aas et al. 2019), resulting in significant reductions in uncertainties and improved riser fatigue predictions. In this case, 4Subsea utilized an ML model to estimate the heading of an FPSO.

Another ML approach replaces the semi-empirical model with an ML model, known as surrogate modeling. The advantage of the surrogate model is that once trained, the model is often computationally fast and can be used in real-time problems like digital twins. For example, ML has proven to reconstruct riser motion using limited measurements. MIT's riser program Kharazmi, Z. Wang et al. (2021) proposed an innovative neural network framework called LSTM-ModNet, which combines a Long Short-Term Memory (LSTM) network with a neural network (NN) to reconstruct riser motion. The model utilizes strain measurements from 24 sensors and employs modal decomposition to reconstruct the response along the entire riser. The result is a fast model combining different sensors to predict the complete displacement. H2Offshore has adopted a similar strategy (Sundararaman et al. 2018), where the NN is trained on simulation data. The result was a model that could use limited real sensors and accurate prediction of the displacement in real time.

Knowing the damping coefficient can be difficult, and additionally, fatigue damage can lead to changes in the material properties. Therefore, having an estimation of the structural parameters is of interest. Kharazmi, Fan et al. (2021) introduced a physics-informed neural network (PINN) for estimating structural parameters. The paper utilized PINN to estimate the damping coefficients given hydrodynamic force and displacement measurements. The surrogate models so far use a neural network to represent the solution. Another approach is also to use the neural network to represent the operator. The MIT Riser program Kharazmi, Z. Wang et al. (2021) suggested a two-part neural network: a "branch net" that learns the operator based on the current profile and a "trunk net" that takes in the coordinates to evaluate the output. The model returns the amplitude along the spatial coordinate inputs. The method successfully learned the current profile mapping to the amplitude by training on simulation data from the semi-empirical model. The prediction is good when the current profile is close to or inside the training range. Resulting in a fast prediction tool that can match the accuracy of the semi-empirical model

So far, it is possible to divide it into three categories. The first is improving the semi-empirical models using ML-based optimization to improve the coefficients. The second approach uses a pure ML model to reconstruct the riser motion using sensor data. The third approach uses the ML model as a replacement for the semi-empirical model while retaining prior knowledge through the use of simulation data in training. The last category falls under the area of Physics Informed Machine Learning (PIML)(G. E. Karniadakis et al. 2021), which is a part of the Scientific Machine Learning (SciML) community (Christopher Rackauckas 2019). The combination of physics-based and machine-learning models also falls under Hybrid Analytic methods (Riemer-Sørensen 2023; Staff 2021).

1.3.2 Hybrid models

In recent years, significant research has been conducted on combining physical and machine learning models. This field is broad and rapidly evolving. For a comprehensive review, please refer to the following sources: Rai and Sahu 2020; Rueden et al. 2021; Pawar et al. 2021; Frank et al. 2020; G. E. Karniadakis et al. 2021. One approach that has gained popularity is the Physics-informed Neural Network (PINN) approach (Cuomo et al. 2022). This method was first introduced in 2019 (Raissi et al. 2019) and has since been applied to various problems and developed further by a team of researchers at Brown University called the CRUNCH group (G. Karniadakis 2023). The team is led by Professor George Em Karniadakis, one of the original authors of the method (Raissi et al. 2019). PINN is a method that utilizes machine learning to solve differential equations. While sophisticated numerical solvers exist to solve differential equations, PINN introduces the capability to estimate unknown parameters in the differential equation while finding a solution. As most physical models are formulated as differential equations, PINN provides a seamless way to combine data with missing physical equations. In addition, the method has demonstrated robustness to noise and limited data (Raissi et al. 2019). PINN has been successfully applied to solve challenging problems, such as the incompressible Navier-Stokes equation (Raissi et al. 2019) and the coupling of the Navier-Stokes equation with the heat equation (Cai et al. 2021). In mechanics, PINN has been employed for structural health monitoring of a beam. By equipping the beam with sensors along its length, the PINN model is utilized to reconstruct the entire displacement profile of the beam (Yuan et al. 2020; Moradi et al.

2023).

PINNs utilize a deep neural network (DNN) as a surrogate model while simultaneously fitting the observed data and the physical equations. One advantage highlighted by L. Lu, Meng et al. (2021) is that the derivatives are obtained through automatic differentiation, eliminating the need for a mesh that traditional numerical differentiation requires. However, a drawback is that the PINN method is currently slower than the traditional numerical method for finding the forward solution of a differential equation. Additionally, finding a suitable neural network architecture for the method relies on empirical approaches (L. Lu, Meng et al. 2021). The PINN approach adds additional terms to the loss functions, resulting in a more challenging optimization landscape (Krishnapriyan et al. 2021; Cuomo et al. 2022; S. Wang, Teng et al. 2020). However, ongoing research is being conducted on automated neural network architecture (Géron 2017; L. Lu, Meng et al. 2021) and improved optimization routines for multi-objective optimization (Cuomo et al. 2022).

1.4 Research gap

To my knowledge, prior research has yet to be conducted on improving the semi-empirical VIV model by solving it using a PINN approach. This novel approach aims to achieve a solution that aligns with the available data while determining the empirical coefficient required for the semi-empirical model to match the data accurately. Additionally, the proposed model can estimate the structural parameters. E. Kharazmseveralho conducted research utilizing the PINN approach to accurately estimate the structural parameters of a vibrating cylinder caused by VIV (Kharazmi, Fan et al. 2021). Similarly, C. Cheng explored similar research in this area (C. Chen et al. 2021). However, it is essential to note a fundamental difference between these studies and this thesis. In the mentioned research, the hydrodynamic forces were obtained from computational fluid dynamics (CFD) simulations. In contrast, in this study, the forces are obtained using the semi-empirical time-domain model VIVANA-TD. This makes it possible to improve the prediction (adjust the empirical coefficient) and infer structural parameters based on the available measurement data.

1.5 Objective

This thesis aims to improve VIV prediction by using a hybrid analytic method. The research question is whether a physics-informed neural network can solve the VIVANA-TD equation and estimate the correct coefficients based on available measurement data.

The PINN methodology will be examined in a simplified constructed forced mass-spring-damper system. This analysis will provide insights into the capabilities and limitations of the PINN approach. Subsequently, the PINN approach will be applied to the VIVANA-TD equation, utilizing simulated measurement data obtained from SIMA (VIVANA-TD) (Sintef n.d.). This application will enable an exploration of the effectiveness of the PINN in addressing the complex dynamics of the VIVANA-TD equation and accurately estimating the associated coefficients and structural parameters.

1.6 Structure of the report

The Introduction motivates the research and presents some of the approaches explored. Chapter 2 explores theories on risers, VIV, VIV model VIVANA-TD, ML, digital twins, hybrid analytics, and PINNs. Chapter 3 explains the problem setup and implementation in Python. Chapter 4 presents and discuss the results. Chapter 5 introduces improvements in PINNs. Finally, chapter 6 presents the conclusion and further work.

Chapter 2

Background theory

This chapter will introduce risers technology, fundamental VIV theory, VIV modeling, ML, digital twins, Hybrid Analytics, and Physics-informed Neural Networks. The chapter is meant to provide information to understand the underlying issues with the physical models and how digitization and technology can be utilized to improve models.

2.1 Riser

A marine riser is a pipe used in the oil and gas industry. Risers connect the subsea wellhead to a floater, allowing for the transmission of fluids between the well and the surface. There are different risers for various marine operations. Marine risers are typically made of steel or other durable materials. They must withstand offshore environmental conditions, including high pressures, temperature fluctuations, and the forces from currents and waves. Marine risers are essential to oil and gas operations, and their design and operation must be carefully planned and executed to ensure safety and efficiency. Failures can have significant consequences for the environment and economy. Risers are placed in deep water far from land, where the environment is challenging and harsh. There are different types of configurations and also different types of pipes. In this thesis, flexible pipes will be the focus. The different configurations are shown in figure (2.1).

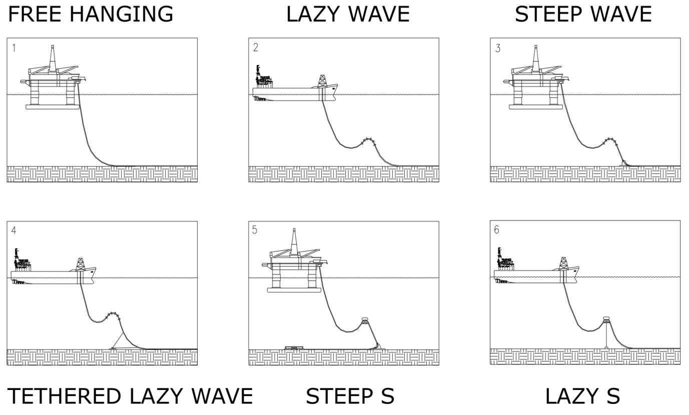


Figure 2.1: Flexible pipe configurations, courtesy Sundararaman et al. 2018

Although risers are used today to transport hydrocarbons, the technology is still relevant for the ongoing transition to renewable energy. One suggestion is that hydrogen is produced by floating offshore wind turbines and stored on the sea bottom (TechnipFMC 2023). The VIV model is also relevant for electrical power cables, an essential part of offshore renewable energy. However, using direct measurements of the vibrations of the riser is often not feasible due to the high initial costs and operational costs to reliably monitor a riser over the whole service life (Sundararaman et al. 2018).

2.1.1 Flexible pipe

Flexible pipes are composed of several layers and complex combinations of steel and polymers. The pipes have multiple degradation mechanisms and failure modes (Fergestad et al. 2017). Due to their complexity and harsh environment, several failures have been reported over the last few years. The failures have led to considerable interest in the industry to build a better understanding and detect failures before they happen (Fergestad et al. 2017). A digital solution with screening and monitoring methods can be essential to understand the failure modes and improve the understanding, leading to better prediction of why they happen and how. DNV has reviewed how digital tools and solutions can improve subsea integrity monitoring (DNV-GL 2016b). DNV suggests that digital solutions will be a key component to increasing the understanding of physical phenomena. The fatigue life of the flexible riser is calculated in the design phase using historical data to predict the environment (Nilsen-Aas et al. 2019).

2.1.2 Failure modes

There are several potential failure modes related to flexible pipes. Some failure modes are mechanical damage from installation, corrosion, and fatigue load from VIV. Mechanical failure can occur due to excessive bending, twisting, or crushing of the pipe, weakening the material and making it more susceptible to ruptures or leaks. Corrosion can occur when the materials are exposed to chemicals or corrosive agents, which degrade the material leading to leaks or ruptures. This thesis limits its scope to fatigue damage from VIV. The magnitude of VIV is not very large (cross-flow $\approx 1 \times D$ and in-line$\approx \times 0.3D$), so the bending stress is typically much lower than the yield strength. The issue is that loads from VIV are cyclic. The material can begin to degrade after many cycles, meaning that on a microscopic scale, the material can get cracks and imperfections. Over time this leads to larger cracks and the riser failing. The degradation over cyclic loads is called fatigue. Fatigue damage is typically calculated using a fatigue curve, which gives the relationship between stress and the number of cycles the pipe can withstand before it reaches its fatigue limit and fails. DNV-GL (2016a) mentions two methods to calculate fatigue damage: The Rain Flow counting method and the Miner's rule. Due to the complex cross-sections, fatigue damage calculation is extra difficult. In complete fatigue estimation, temperature and pressure should also be included in the analysis, as rapid changes in temperature will affect the materials. There is also the constant development of materials used in the riser, so using data from old risers past their design life might not be valid for the risers being built today.

2.1.3 Inspection and monitoring methods

During the design phase, the design life is calculated, but due to the large uncertainty in the environments and modeling, this includes a significant safety factor. Since replacing or maintaining a flexible pipe is expensive, the response should be modeled as accurately as possible such that the riser's expected lifetime is accurate. Thus we want to have integrity management of the pipes. Due to its complexity, inspection and integrity monitoring involve many disciplines. In the Riser handbook by Fergestad et al. (2017) table C1.2: there is a whole list with inspection and monitoring lists and explanations on how common they are in practice. However, only a few monitoring methods focus directly on VIV. As the flexible risers typically span several hundred meters down in the ocean, installing and maintaining sensors are challenging and expensive. Transferring and getting electrical power are typical issues. Visual inspection and scanning can also provide reasonable indications of the status of the riser regarding fatigue. There are some examples where a few sensors are placed on the riser. GE Oil and Gas have suggested a method on how acceleration sensors can be installed and used to collect data from a drilling riser, illustrated in Figure 2.2

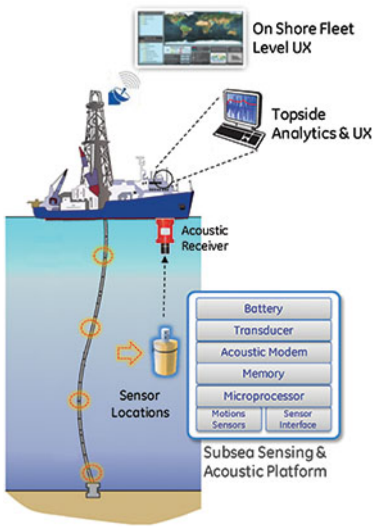


Figure 2.2: Drilling riser, sensor placement, images from Myers 2017

The sensors measure the acceleration and the ocean current. The data can be used in a digital twin to get a complete description of the dynamic response of the whole system. There are a couple of ways this can improve the model: The data can provide more accurate environmental conditions. The data can validate the models. Lastly, by using a hybrid analytic method, the collected data can further optimize the model and represent the physics not included. In addition, this approach requires only a few sensors, making it cost-effective. Therefore, the additional cost may be justified as it contributes to better integrity insight, ultimately leading to an extended lifetime.

2.1.4 Lifetime assessment

Lifetime assessment for flexible pipes involves evaluating the condition over time to determine their fitness for continued use. This assessment would consider factors such as the materials used in their construction, the amount of use and wear they have undergone, and any potential damage or degradation that may have occurred. The assessment results would then determine whether the flexible pipes are still suitable for use or need to be repaired or replaced. Due to the limited historical data due to manufacturers making changes and evolving the flexible pipes, assessing if the flexible pipe can safely operate after sustained damages or anomalies is challenging. Fergestad et al. (2017) states that visual inspection and scanning can be expensive and lead to downtime. Thus having continuous monitoring over the flexible pipe is of significant interest.

2.1.5 Power cables

In the offshore industry, subsea power cables are used in operations such as oil and gas platforms and wind farms. For oil and gas platforms, these power cables are used to transfer electricity to various subsea components and in the electrification of the platform itself. In wind farms, power cables are utilized to transfer energy from wind turbines to the power grid. Power cables share many characteristics with risers: they are flexible, round, slender, and placed in a similar environment for approximately the same lifetime. Therefore, they also exhibit similar failure modes (Delizisis et al. 2022). VIV fatigue is a critical failure mode that can occur in power cables. However, the depth of research on VIV fatigue in power cables is less extensive than in risers. Delizisis et al. (2022) conducted a study examining how VIV models perform on a power cable, specifically focusing on hydrodynamic parameters. They found that by adjusting the hydrodynamic parameters, they could replicate all relevant VIV behavior observed in a small-scale experimental power cable test. This suggests that current models can accurately predict hydrodynamic forces.

However, power cable cross-sections are complex. Factors such as the number of conduits, type of insulation, armor, and others all influence the cable's fatigue life and structural properties. In practical terms, the fatigue life calculation for a power cable is similar to that for a riser, where an S-N curve derived from experiments is used (Nasution et al. 2014).

2.1.6 Summary

It is important to predict the lifetime of risers or slender structures in the ocean. The lifetime and condition of these slender structures can be determined by inspection, screening, and using models. By using models, we can gain a better understanding of the system and its operations. Digitization can allow for a better connection between all the different methods and the incorporation of measurement data, ultimately resulting in a more comprehensive understanding of the system. Fatigue damage is mainly due to cyclic loads from VIV. The model needs to be further improved to obtain a more accurate lifetime assessment, either through a digital twin approach or a more accurate model formulation.

2.2 Vortex-Induced Vibrations

When buff structures are exposed to moving fluid, we may get vortex shedding, leading to oscillating lift and drag forces. For flexible pipes, the forces will lead to structural oscillations. The phenomena are distinguished into vortex-induced motion (VIM) and vibrations (VIV). VIM refers to vortex-induced rigid body motions. A typical example is the VIM of a spar-type floater for a wind turbine or oil/gas production platform. Due to a significantly larger diameter than risers/cables, the shedding frequency is much lower. If the vortex shedding force is close to the slender structure's natural frequencies, the resulting response is amplified, called resonance (DNV-GL 2016a). Risers often have complex cross-sections, buoyancy modules, bending stiffness, and strakes, increasing complexity further. In sheared current, local vortex shedding will also lead to the structure interacting along the length; parts of the structure will transfer energy to the structure, and parts will dampen the response. The literature study is based on DNV VIV Best Practice (DNV-GL 2016a). VIV research aims to get the best understanding, prediction, and prevention of VIV. Due to the complexity of VIV physical and numerical experiments, theoretical analyses and physical insights all are necessary for understanding and modeling. The most important theoretical and experimental findings in VIV prediction for flexible pipes will be presented.

2.2.1 Vortex shedding

Vortex shedding is one of the fundamental aspects of VIV, and it is the underlying reason for the alternating force. Vortex shedding occurs because, in actual fluid flow, the no-slip boundary condition must be satisfied. No-slip boundary condition means there cannot be a sudden change in fluid flow. For flow around a cylinder, the fluid velocity close to the cylinder must be the same as the velocity of the cylinder. A thin boundary layer, denoted as δ in Figure 2.3, formed near the surface due to no slip. Figure 2.3 figure shows how the flow will increase from zero on the cylinder (or the velocity of the cylinder) to a velocity determined by the current and the shape of the cylinder. The pressure gradient will decrease, moving downstream until it reaches the separation point where the pressure gradient is zero. Beyond the separation point, the pressure gradient will start to increase. Figure 2.3 illustrates that a shear layer develops after the separation point, and the flow contains significant vorticity. The vorticity causes the shear layer to roll into a vortex with a sign identical to the incoming vorticity (vortex A in Figure 2.3). Similarly, a vortex B forms, rotating on the opposite side of the cylinder (vortex B) Sumer and Fredsøe 2006.

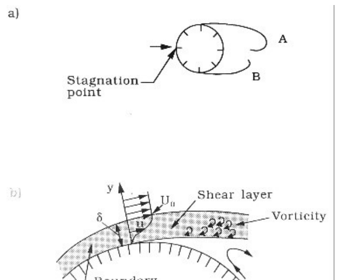


Figure 2.3: Boundary layer, courtesy of Sumer and Fredsøe 2006

From Figure 2.3, it is clear that inertial and viscous forces are important in describing vortex shedding. The relation of these forces can be described with the Reynolds number, a non-dimensional parameter.

$$Re = \frac{U \cdot D}{\nu} \tag{2.1}$$

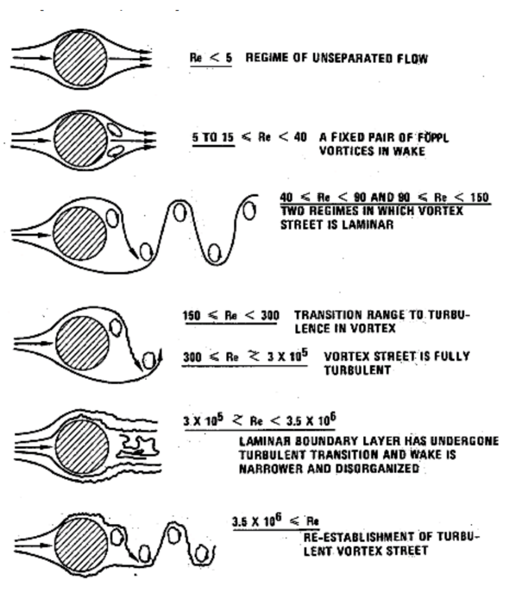
ν is the kinematic viscosity of the fluid, U is the flow velocity, and D is the pipe diameter. According to the value of Re, we can distinguish four flow regimes, assuming a smooth cylinder: A crude partition of Reynolds number is presented based on Lecture notes Sea loads (Greco 2022).

- *Subcritical flow regime*: $\mathrm{Re} \leq 2 \times 10^5$
- *Critical flow regime*: $2 \times 10^5 \leq \mathrm{Re} \leq 5 \times 10^5$
- *Supercritical flow regime*: $5 \times 10^5 \leq \mathrm{Re} \leq 3 \times 10^6$
- *Transcritical flow regime*: $\mathrm{Re} > 3 \times 10^6$

The vortex shedding frequency (f_s) can be expressed with Strouhal's number for a fixed cylinder in steady uniform flow. Strouhals number is a function of the Reynolds number.

$$S_t = \frac{D \cdot f_s}{U} \tag{2.2}$$

In Equation 2.2, D is the diameter, U is the flow velocity, and f_s is the vortex shedding frequency. Investigating the flow around a circular cylinder in a steady current. Figure 2.4 illustrates the patterns behind a fixed, rigid pipe for different Reynolds numbers. Reynolds number higher than 40, we get the formation of a pattern of vortices that shed alternately at either side of the cylinder at a frequency.



Re < 5 REGIME OF UNSEPARATED FLOW
5 TO 15 ⩽ Re < 40 A FIXED PAIR OF FÖPPL VORTICES IN WAKE
40 ⩽ Re < 90 AND 90 ⩽ Re < 150 TWO REGIMES IN WHICH VORTEX STREET IS LAMINAR
150 ⩽ Re < 300 TRANSITION RANGE TO TURBULENCE IN VORTEX
300 ⩽ Re ≿ 3 X 10^5 VORTEX STREET IS FULLY TURBULENT
3 X 10^5 ≿ Re < 3.5 X 10^6 LAMINAR BOUNDARY LAYER HAS UNDERGONE TURBULENT TRANSITION AND WAKE IS NARROWER AND DISORGANIZED
3.5 X 10^6 ⩽ Re RE-ESTABLISHMENT OF TURBULENT VORTEX STREET

Figure 2.4: Vortex shedding, courtesy of Lienhard, 1966

Laminar vortex shedding occurs in the range of Reynolds numbers from 40 to 150, forming what is known as a Karman vortex street. As the Reynolds number increases above 150, turbulence appears in the vortex street. At Reynolds numbers above 300, the vortex street becomes fully turbulent, and the flow is referred to as subcritical up to $3 \cdot 10^5$. Most VIV-related experiments are conducted in the subcritical flow regime, but the flow will enter the critical or supercritical regimes in full-scale cases. This will introduce errors, but it is generally understood that experimental data from the subcritical regime tends to overestimate VIV response when applied to cases with higher Reynolds numbers, as noted in the DNV Best Practice guidelines (DNV-GL 2016a).

In addition to being dependent on the Reynolds number, vortex shedding is also influenced by factors such as surface roughness, cross-sectional shape, incoming turbulence, and the shear in the incoming flow (Triantafyllou et al. 2016). For example, as illustrated in Figure 2.5, the Strouhal number largely varies with surface roughness for a fixed pipe.

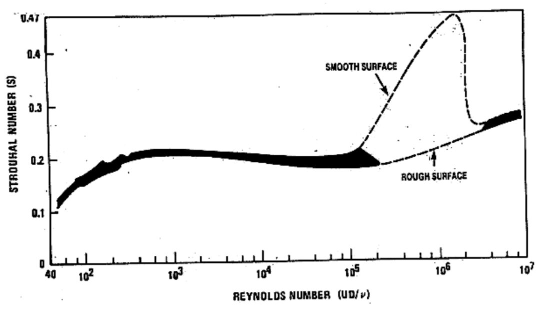


Figure 2.5: Strouhal number, from Lienhard, 1966

The Strouhals number is close to 0.2 in sub-critical flow, and the vortex shedding process is nearly constant. Rough and smooth differences become significant when the Strouhals number approaches the critical flow regime. Usually, offshore risers have a sufficiently rough surface to give a stable vortex shedding (DNV-GL 2016a).

2.2.2 Forces due to vortex shedding

The alternating vortex shedding produces a hydrodynamic force in the cross-flow direction and in-line. The force is found by integrating the pressure along the surface of the cylinder. Assuming harmonic motion, the cross-flow component of the pressure resultant, F_H, is described by,

$$F_H = F_{H0}\sin(\omega t - \epsilon) = -F_A \sin(\omega_{CF} t) - F_E \cos(\omega_{CF} t) \tag{2.3}$$

The first part of the equation represents the added mass force and is in phase with the acceleration. The second part represents the excitation force and is in phase with velocity. A similar expression can be found for the in-line force. For the in-line force, the shedding frequency will be two times the cross-flow frequency due to both vortices affecting the same side. There is also a constant friction force term named drag force. Assuming harmonic oscillation, it is possible to describe the amplitude of the excitation force in phase with cross-flow as,

$$F_E = \frac{1}{2}\rho C_E D L U^2 \tag{2.4}$$

In Equation 2.4, ρ is the density of the fluid, D is the diameter of the cylinder, L is the unit length, and C_E is the excitation force coefficient. This part is in phase with the velocity and controls the energy transfer between the fluid and the cylinder. The excitation coefficient is dependent on amplitude and frequency. To find the coefficients, two experiments are conducted: elastic mounted and forced oscillations.

2.2.3 Elastically mounted rigid cylinders subjected to cross-flow motions

The cylinder is mounted on linear springs and is free to move as the moving fluid exerts forces on the cylinder. The cylinder will start to oscillate if the flow is in a regime where vortex shedding occurs. The still water natural frequency described in Equation 2.5 characterizes the dynamic model. When it starts to oscillate, the added mass term changes and the new frequency term is denoted f_{osc} occurs.

$$f_0 = \frac{1}{2\pi}\sqrt{\frac{k}{m + m_{a,0}}} \tag{2.5}$$

$$f_{osc} = \frac{1}{2\pi}\sqrt{\frac{k}{m + m_a}} \tag{2.6}$$

m_a is added mass, k spring stiffness, and m is mass. The added mass term is a part of the hydrodynamic force and is in phase with the cross-flow acceleration of the cylinder. The cylinder oscillation will further influence the vortex shedding frequency. The oscillating frequency can further be described as non-dimensional by $\hat{f}$; this uses the reduced velocity U_R. By using non-dimensional parameters, it is easier to compare and analyze across different systems or scales.

$$U_R = \frac{U}{D \cdot f_0} \tag{2.7}$$

$$\hat{f} = \frac{f_{osc} \cdot D}{U} \tag{2.8}$$

The top of the Figure 2.6 shows the non-dimensional amplitude with respect to the reduced velocity. This figure differs from a standard mass-spring system with an oscillating force instead of having an amplitude peak when the force approaches the system's natural frequency. The response amplitude increases quickly as the frequency approaches the beam's natural frequency. Still, when the response frequently increases further, the response amplitude is the same until a reduced velocity of 18. This phenomenon is known as lock-in. In this range, the frequency does not follow the Strouhals shedding frequency. This comes from the added mass term being in phase with the motion of the pipe and not the total hydrodynamic force, despite being a part of the hydrodynamic force (DNV-GL 2016a). This also gives the possibility for a negative added mass. The variability of the added mass to adjust the structure's natural frequency is one of the fundamental features of VIV. The response depends on the mass of the pipe; a greater mass can dominate the added mass term. It is worth noting that the natural frequency defined in Equation 2.5 illustrates that the influence of the added mass variation on the pipe will depend on the structure's mass. the mass ratio m* in Equation 2.9 is used to illustrate this,

$$m^* = \frac{4 \cdot m}{\pi \rho D} \tag{2.9}$$

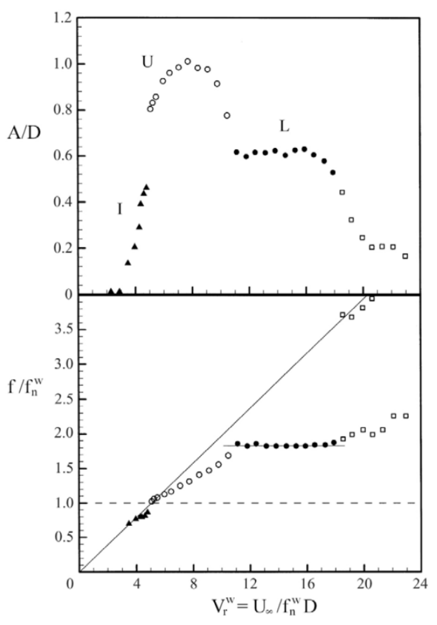


Figure 2.6: Cross-flow VIV response of an elastically supported pipe with low structural damping and low mass ratio (Govardan & Williamson, 2000). L denotes the lock-in.

Another essential aspect of VIV is that it is a resonance phenomenon with self-limiting vibration amplitudes. The self-limiting amplitude is clearly illustrated in Figure 2.7. For a larger initial amplitude than a steady state, the response amplitude gradually decreases until it reaches a steady state value. On the other hand, suppose the initial amplitude is smaller than the steady state. Then, the amplitude gradually increases until it reaches a steady state since the excitation coefficient depends on the response amplitude and frequency. Finding the coefficient in a free oscillation test is difficult because then the transient phase needs to be studied. Thus forced oscillation experiments are more common practice (DNV-GL 2016a).

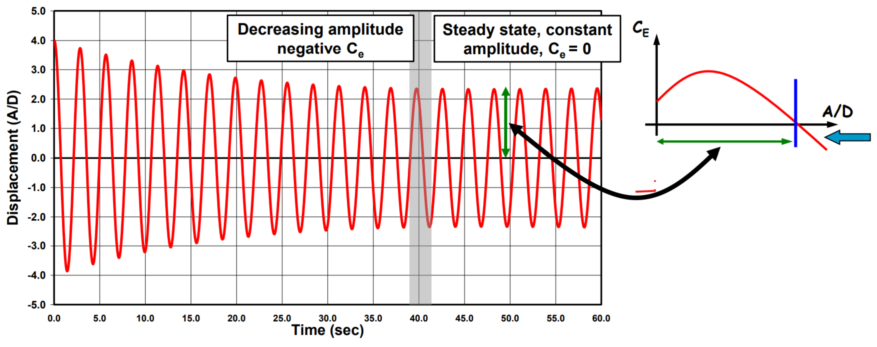


Figure 2.7: Response oscillation for cylinder, from Wu 2022

2.2.4 Forced oscillations

In a forced oscillation experiment, a cylinder undergoes a prescribed motion, which can be inline, cross-flow, or a combination of both. The driving force is the measured force and can be separated into two components: one in phase with acceleration (added mass force) and another in phase with velocity (excitation force). The motion is described by the equation $x(t) = x_0 \sin(\omega t)$, where x represents the displacement, ω denotes the frequency, and x_0 represents the displacement amplitude. The dynamic equilibrium can be expressed by the following equation Wu 2022.

$$(m + m_a)[-\omega^2 A sin(\omega t)] + F_E cos(\omega t) = \text{Driving force} \tag{2.10}$$

From Equation 2.10, the added mass m_a and the excitation force F_e can be calculated. This makes it possible to find the excitation force and added mass coefficient for a chosen combination of amplitude and frequency. There have been several attempts to systematical find the coefficient for both the inline and cross-flow response by a systematic variation of controlling parameters (DNV-GL 2016a). However, because the force coefficients are highly sensitive to phase angle, performing the needed amount of experiments is not feasible. The results from forced oscillation are presented in contour plots where the amplitude ratio is on the y-axis, and the nondimensional frequency is on the x-axis. The added mass and excitation force is found when the coefficient is zero (no excitation).

2.2.5 Flexible pipes

When using flexible pipes, there are infinitely many natural frequencies. In dynamic analysis, it is common to represent the displacement as a sum of mode shapes.

$$r(x,t) = \sum \phi_i(x) y_i(t) \tag{2.11}$$

where ϕ_i is the mode shape, which is the variation of the response in space, and y_i is the scaling factor that varies with time. The mode shape has to satisfy the boundary conditions. It gives the mode shape total response in time (Langen and Sigbjørnsson 1986). Figure 2.8 shows the first ten-mode shapes.

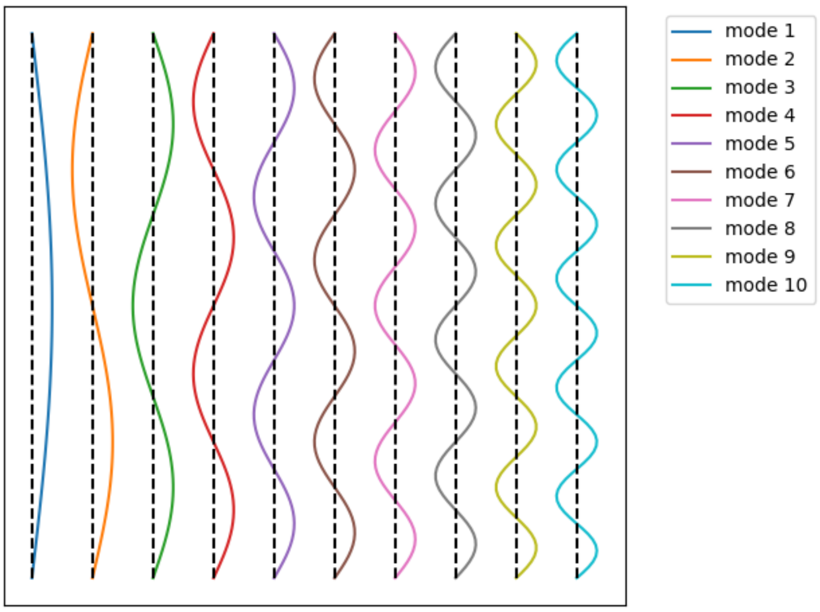


Figure 2.8: Modeshapes

Each mode shape corresponds to one of the structure's natural frequencies. The forces from vortex shedding can synchronize with any of these frequencies, resulting in lock-in.

Increasing current velocity might lead to the synchronization skipping to the next lock-in frequency. In calculating the reduced velocity parameter, the natural frequency is used in the calculation of reduced velocity. The net energy decides the vibration amplitude in the structure. The net energy is a sum of the hydrodynamic excitation, hydrodynamic damping, and structural damping forces (Wu 2022). If the amplitude stays the same, but with a higher frequency, the fatigue damage will increase. In real situations, the flow will vary in space and time. Space and time-varying flow lead to many eigenmodes being exited simultaneously, resulting in a multi-mode frequency response. Risers are not always uniform in cross-section. This result in different vortex shedding, leading to a multi-mode frequency response. Inline vibration also needs to be considered, and higher modes will be excited due to the shedding frequency being twice the frequency. The amplitude is typically 40 % of the CF amplitude (Wu 2022), but they can be equally crucial in fatigue calculations due to the higher modes.

2.3 VIV modeling

To model VIV, there is a need for a hydrodynamic model that calculates the force induced on the structure and a model that calculates the structure's response. Today, accurate finite element solvers are used to calculate the structure's response. The challenge lies in calculating the hydrodynamic force. CFD and semi-empirical methods are the two main approaches to estimating the hydrodynamic force in VIV analyses.

CFD relies on the Navier-Stokes equations to describe the motion of fluids. These equations are then solved by discretizing the fluid and applying the Navier-Stokes equation to each element. Due to the complexity of these equations and the need for fine mesh, CFD is currently limited to simplified shapes and low Reynolds numbers. The method is too computationally heavy for large, complex shapes like a flexible riser. In practice, semi-empirical methods are used (DNV-GL 2016a). Semi-empirical methods are based on a combination of fundamental theory and experiments observing the system, leading to mathematical descriptions. These mathematical descriptions are established on several simplifications. Traditionally, VIV models are based on empirical models that respond at one or a set of discrete frequencies, such as SHEAR7, VIVA, and VIVANA-FD (DNV-GL 2016a). This frequency-based approach has to assume a structure model that is linear. A linear structure model limits the ability to account for non-steady VIV responses and non-linear structural changes, such as tension variation (Sang Woo Kim et al. 2021). To address this limitation, researchers at NTNU have developed a time domain model, VIVANA-TD. The time-domain model can account for non-steady VIV response and non-linear structural changes.

2.3.1 VIVANA-TD

VIVANA-TD is a time-domain hydrodynamic load model developed at NTNU and later improved and validated by Sintef (Wu et al. 2020). The model is based on strip theory, which divides the riser into finite amounts of independent strips. The force for each strip is described by,

$$F = (C_A + 1)\rho\frac{\pi D^2}{4}\dot{u}_n - C_A\rho\frac{\pi D^2}{4}\ddot{x}_n + \frac{1}{2}\rho C_D D v_n|v_n| + \frac{1}{2}\rho D C_{v,CF}|v_n|(j_3 \times v_n)cos\phi_{v,CF} + \frac{1}{2}\rho D C_{v,IL}|v_n|v_n cos\phi_{v,IL} \tag{2.12}$$

The first three terms originate from the Morrison equation. C_A represents the added mass coefficient, C_D is the drag coefficient, $C_{v,CF}$ is the vortex-shedding force coefficient in the cross-flow direction, and $C_{v,IL}$ is the vortex-shedding force coefficient in the inline direction. The instantaneous phases of the vortex shedding forces in the inline and cross-flow directions are represented by $\phi_{v,IL}$, and $\phi_{v,CF}$, respectively. The fluid density is denoted by ρ, and D is the cylinder diameter. Figure 2.9 illustrates the current, hydrodynamic force, and response vectors. v_n is the normal direction of the relative structural velocity. The current vector is denoted by u, where u_n is the normal direction and u_t is the tangential direction with respect to the cylinder strip. j_3 is pointing in the direction of the cylinder axis. By neglecting the tangential direction of the current, the hydrodynamic force can be described in the local j_1, j_2 plane. $F_{v,x}$ is the in-line vortex shedding force, and $F_{v,y}$ is the cross-flow shedding force.

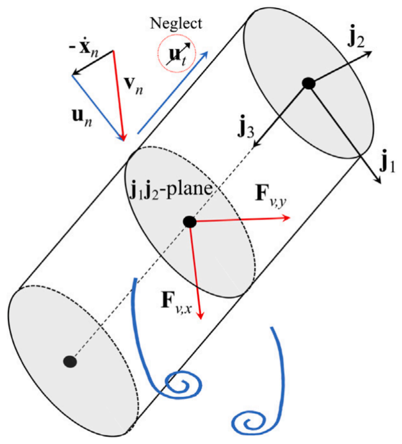


Figure 2.9: Cylinder strip with relevant vectors and local coordinate system, from Sang Woo Kim et al. 2021

This model aims to describe VIV hydrodynamics as simply as possible while maintaining the underlying physics. It is considered semi-empirical because certain parts of the Morrison terms are derived from the first fluid mechanics principles. In order to account for the observed phenomenon of synchronization of VIV loads and structural responses, an additional term needs to be included. The coefficient is determined empirically through experimentation. The parameters ($C_{v,CF}$,$C_{v,IL}$, C_D) are based on empirical observations. They can be influenced by additional factors such as Reynolds number and surface roughness. The VIVANA-TD model reasonably represents VIV loads, and the predicted results agree with general measurements Wu et al. 2020. However, the model still has its limitations, and like all empirical models, it requires extensive model test data to determine the appropriate coefficients.

2.3.2 Structural model

The structural model used in VIVANA-TD uses finite elements. The structure model is governed by the theoretical equation of motion $M\ddot{r}+C\dot{r}+Kr = F$, where M represents the structural and hydrodynamic added mass, C is the structural and hydrodynamic damping, K is the stiffness matrix, and F represents the external force. Since M, C, and R are functions of the response r, an iterative solution scheme is required (DNV-GL 2016a).

2.3.3 Limitations

The workflow of a semi-empirical VIV model is illustrated in figure 2.10.

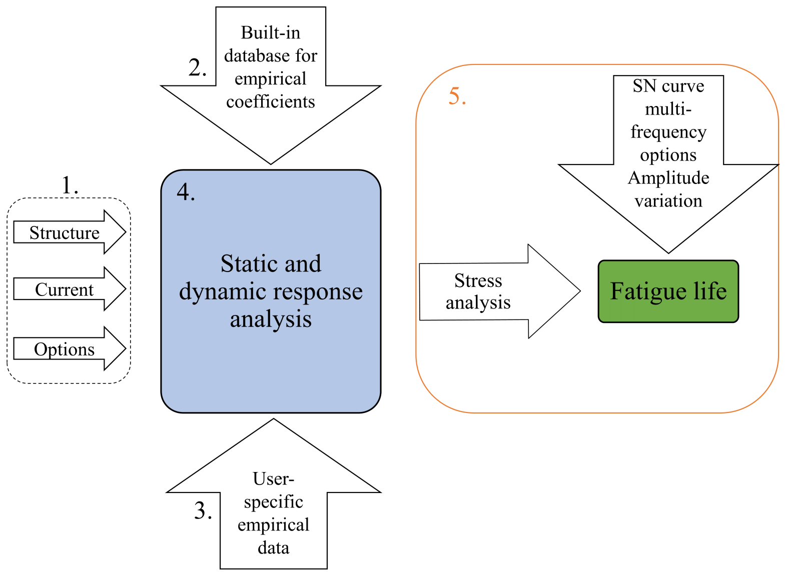

2.
Built-in database for empirical coefficients
1.
Structure
Current
Options
4.
Static and dynamic response analysis
3.
User-specific empirical data
5.
SN curve multi-frequency options Amplitude variation
Stress analysis
Fatigue life

Figure 2.10: Semi-empirical work-flow, adapted from DNV best practice DNV-GL 2016a

The results from the semi-empirical model depend on the accuracy of the influencing parameters, which are labeled as 1 and 3 in Figure 2.10. The validity of the database of empirical coefficients, which serves as the foundation for the load and response model, is indicated as 2 in Figure 2.10. Some significant assumptions in the empirical database include simplified added mass models and non-coupled IL and CF response models. In addition, the scaling approach constrains excitation coefficient data at different Reynolds numbers and surface roughness ratios (DNV-GL 2016a). The accuracy of the model also relies on the simplifications made, identified as 4 in Figure 2.10. According to DNV best practice (DNV-GL 2016a), the model's validity domain must be confirmed based on the experimental database. Additionally, a stress analysis must be conducted considering the response, indicated with 5 in Figure 2.10 to determine the fatigue life. The ocean current will also vary in direction and speed, leading to different vortex shedding and a more complex response. This model does not account for three-dimensional effects.

DNV best practice states, "The present understanding and capability of engineering software for VIV makes it infeasible to carry out analyses that in detail can reproduce VIV as we can observe the phenomenon on real structures" (DNV-GL 2016a). This leads to that large safety margins that are expensive. The models are also insufficient to use together with various sensor data. This raises the question of whether it is possible to implement machine learning techniques in conjunction with measurements to improve the model.

2.3.4 Summary

Vortex shedding is a fundamental aspect of VIV, causing alternating forces on structures in fluid flow. It occurs due to the no-slip boundary condition, where flow around a cylinder forms vortices on either side. Vortex shedding is influenced by Strouhal's number, Reynolds number, surface roughness, and incoming flow characteristics. The hydrodynamic forces from vortex shedding include oscillating cross-flow and inline forces, which can be estimated using semi-empirical methods. VIV models, like VIVANA-TD, aim to capture the physics of VIV while considering simplified mathematical descriptions. However, due to uncertainties in the simplified model, influencing parameters, and relying on empirical coefficients, a large safety margin is typically included in the analysis. In order to reduce the safety margin in the VIV model, a better understanding of the relationship between the response and parameters is needed. Additionally, new approaches incorporating more data can help reduce uncertainties and thus reduce safety margins.

2.4 Machine learning

ML is a part of AI that uses algorithms to learn from data and make predictions or decisions without being explicitly programmed. NN is a subset of ML and arguably the heart of ML. NN are popular because of their ability to learn and model complex patterns and relationships.

2.4.1 Neural Networks Architecture

NN is inspired by the way that biological neurons in the human brain communicate with each other. This is why NNs are often referred to as artificial neural networks (Goodfellow et al. 2016). However, while the initial concept of NNs was influenced by biology, machine learning has since evolved in many different directions. Today's NNs are often highly abstracted and optimized for specific tasks rather than being direct replicas of biological processes. Thus NNs should, instead of being thought of as a representation of the brain, be thought of as a new way of representing and modeling complex functions. NNs are a large function that is well-suited for optimizing. The basic computation unit in a neural network is the Neurons, also called nodes or units. The structure of a neuron is illustrated in Figure 2.11; each neuron has a set of weighted input links (ω), a simple function (a), and output links. The neurons can be connected arbitrarily, allowing for many different architectures. It is common to organize the neurons into groups called layers connected in a series. The most basic neural network is fully connected. In a fully connected neural network, all the neurons in each layer are connected to all the neurons in the next layer; a feed-forward neural network with four layers is illustrated in Figure 2.12. The circles represent the neurons, and the links represent the weights. Each layer is a function of the layer that preceded it. For example, three layers l_3, l_2, and l_1 form the mapping $f_{\mathcal{N}}(x) = f_{l3}(f_{l2}(f_{l1}(x))))$ where each layer represents a function and $\mathbf{x}$ is the input. A network with more than one hidden layer is called a deep neural network (Goodfellow et al. 2016).

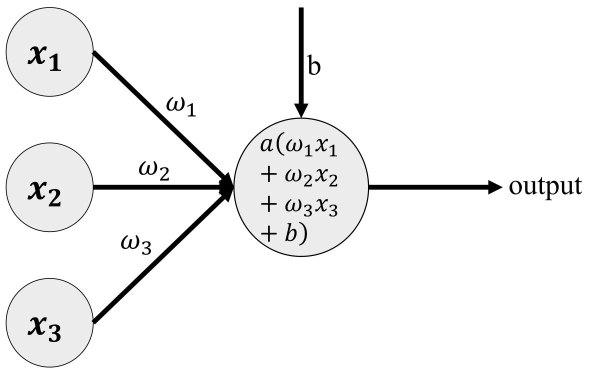


Figure 2.11: Neuron

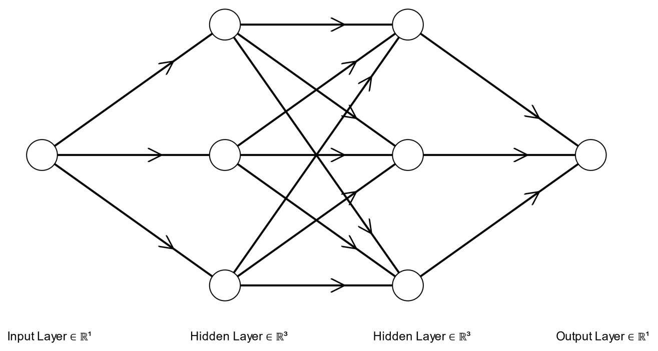


Figure 2.12: Feed-forward Neural Network, generated using (Lenail 2023)

The equation of one layer becomes:

$$\mathbf{f_l} = a(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.13}$$

$\mathbf{x}$ is the input vector to the layer, $\mathbf{W} \in \mathbb{R}^{f_l \times f_{l-1}}$ is the weight matrix, $\mathbf{b} \in \mathbb{R}^{f_l}$ is the bias vector that is added to the weighted sum, and a is the activation function that is applied element-wise to the resulting vector. In DeepXDE paper (L. Lu, Meng et al. 2021), the NN is denoted $\mathcal{N}^L$ is L- is the total number of layers, the resulting feed-forward neural network $\mathcal{N}$ is then

$$\text{input layer:} \quad \mathcal{N}^0(\boldsymbol{x}) = \boldsymbol{x} \in \mathbb{R}^{d_{in}} \tag{2.14}$$

$$\text{hidden layers:} \quad \mathcal{N}^l(\boldsymbol{x}) = a^l(\boldsymbol{W}^l \mathcal{N}^{l-1}(\boldsymbol{x}) + \boldsymbol{b}^l) \in \mathbb{R}^{N_l} \quad \text{for } 1 \leq l \leq L-1 \tag{2.15}$$

$$\text{output layer:} \quad \mathcal{N}^L(\boldsymbol{x}) = \boldsymbol{W}^L \mathcal{N}^{L-1}(\boldsymbol{x}) + \boldsymbol{b}^L \in \mathbb{R}^{d_{out}} \tag{2.16}$$

The neural network represents a function where the parameters $\boldsymbol{\Theta}$ are represented by the weights $\mathbf{W}$ and bias $\mathbf{b}$. Which becomes the following function $\mathbf{f}[\mathbf{x}, \Theta] = \mathbf{y}$, where $\mathbf{y}$ is output and $\mathbf{x}$ is input. The weights represent the strength of the connection of the neurons and bias how often it is active. When designing an NNs, the amount of layers and how many neurons per layer one should choose is a difficult task. Generally, a deeper (more layers) network requires fewer neurons per layer and fewer total parameters (Goodfellow et al. 2016). However, deeper nets are harder to train. The choice of activation function is an important part of the capabilities of the neural network. The number of parameters decides how complex functions can be represented. The ability to express complex functions is called expressibility. The activation function is the part that introduces nonlinearity capabilities into the model; thus, choosing a linear activation function yields a model that cannot describe nonlinear functions. If a linear activation function is used, the whole network could be reduced to one layer since the input and output are just matrix multiplications. The most common activation functions used today are ReLu, Tanh, and Sigmoid. ReLu is the most popular today, mainly because it does not have a problem with vanishing gradients, and it works well in a large number of cases (Goodfellow et al. 2016). Activation functions have a big effect on the training of neural networks because traditional training requires the gradient of the neural network. Since ReLu is not a smooth function differentiating two times yields zero, this can be an issue in cases where the double derivative is required for training or optimizing. Figure 2.13 illustrates the three different activation functions and its derivative.

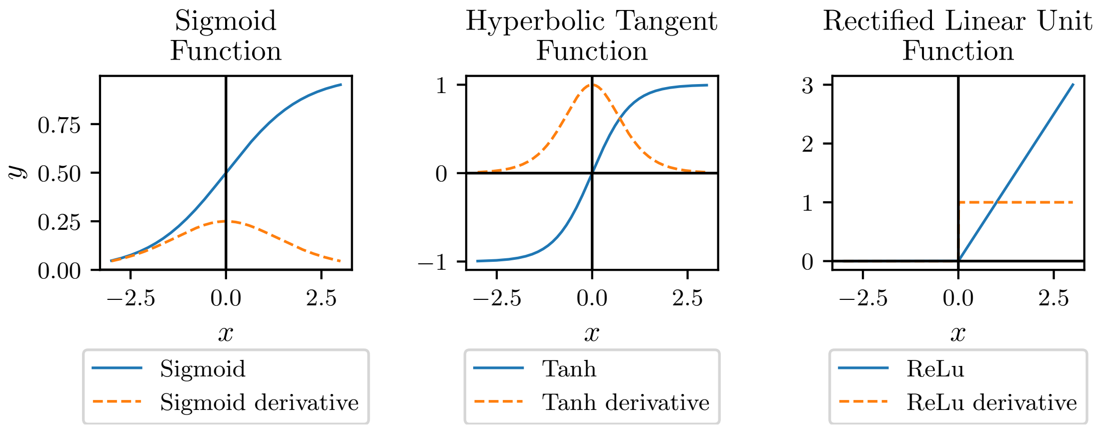


Figure 2.13: Activation function and its derivatives

The parameters that determine the neural network structure is called hyperparameters. Neural networks can be designed in many different ways due to the extensive amount of hyperparameters. Finding the most effective and capable NN can be challenging, often requiring an extensive search (Géron 2017). A lot of research is going into how to select hyperparameters for a neural network. There are optimization routines that can make this process more efficient compared to random searches (Géron 2017). A common practice is to select a more extensive neural network than necessary (e.g. more layers). While this approach can save time, it also increases the risk of overfitting, which is when the model performs poorly on unseen data. However, there exist measures that prevent overfitting. So far, only feed-forward neural networks have been considered. However, there are also other types of NNs, such as convolution neural networks, which are suitable for image recognition, and recurrent neural networks, which are used to capture history. These types of NNs have unique characteristics that make them more suitable for specific tasks. Research efforts focus on discovering new structures, and in recent years, there have been notable breakthroughs, such as the transformers used in the highly debated GPT-3.5 model (network architecture used in ChatGPT)(Wikipedia 2023). While there are countless ways to design a neural network, the most important aspect of a neural network is its capability to learn and represent a desired function.

2.4.2 Feed-forward Deep Neural Networks Training

When training a neural network, the goal is to minimize the difference between the real function and the NN model. In order to minimize the difference, the optimal parameters ω^* and b^* must be found. The loss function describes the mismatch between the model output $\mathbf{u} = \mathbf{f}[\mathbf{x}, \Theta]$ and the ground truth u^*. The loss function outputs a single scalar value used in training to minimize the loss. In addition to having a loss value, it is also important for the NN to represent the function between the measurement points. Thus, in some cases, the training data is divided into a validation set to check how well the model matches the function between the data points. If no validation is used, there is a risk of overfitting the data, meaning that the data points are perfectly matched, but the NN might be completely wrong between the points. Finding the right parameters becomes an optimization problem where the loss function is the objective function. The choice of the loss function can greatly impact the optimizer's capability to find the best parameters. One of the most common cost functions is the mean squared error (MSE). MSE is defined as,

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y_i} - y_i)^2 \tag{2.17}$$

In Equation 2.17, i is the number of N training data. In most cases, this performs well as it penalizes the samples further away exponentially more. However, in cases where the output is very small scale, the loss might become too small; then the absolute value might be a better option than squared.

An optimizer is necessary to determine the weights and biases that best represent a function. The optimizer's role is to locate the minimum value of the loss function. Most optimizers require the first-order derivative (gradient) to find the minima. These gradients are computed from the outputs towards the inputs using the chain rule. Due to the direction from outputs to inputs, this process is called backpropagation. A significant issue with this approach is the potential for gradients to progressively decrease as the algorithm moves through the layers of the neural network. This can result in the first layers remaining unchanged, a phenomenon called the vanishing gradient problem. Oppositely, in some cases, the gradients can become excessively large in the initial layers, a problem known as exploding gradients (Géron 2017). This issue presented a significant obstacle in the machine learning field for a long time. However, in 2010, Xavier Glorot et al. published a paper Glorot and Bengio (2010) in which they discovered that the initialization of weights significantly impacted training, and problems with vanishing and exploding gradients became less common. Glorot suggested a scheme for initializing the weights that were tailored to the different activation functions.

Many of the most commonly used optimization algorithms are further extensions of gradient descent. Gradient descent function by calculating the gradient direction and taking a step proportional to the negative gradient. At each step, the weights are updated based on the following rules:

$$w = w - \alpha\frac{\partial C}{\partial w} \tag{2.18}$$

$$b = b - \alpha\frac{\partial C}{\partial b} \tag{2.19}$$

In these equations, w represents the weight, b is the bias, α is the learning rate, and C is the objective function. If the learning rate is too large, there is a risk of overshooting and oscillating around the minimum. This is illustrated in Figure 2.14, where the red arrows show a learning rate of 0.1 and the blue arrows show a learning rate of 0.8. The learning rate of 0.8 overshoots and starts to oscillate. On the other hand, a small learning rate results in slow convergence. Thus, finding the optimal learning rate is crucial for gradient descent. However, this is a simplified, convex example. For NN, the cost function is non-convex, leading to potential issues with the algorithm settling at local minima instead of the global minimum or getting stuck on a plateau. This is illustrated in Figure 2.15.

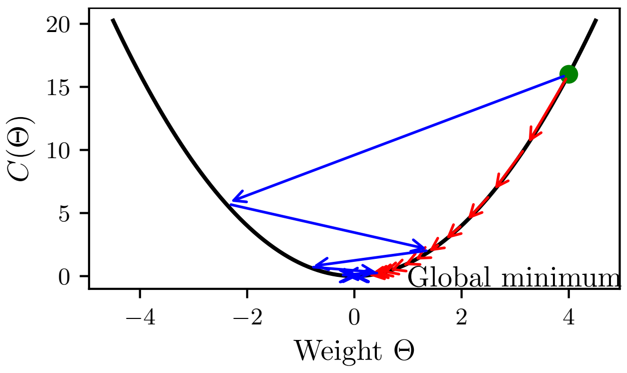


Figure 2.14: Gradient descent, function $C(\theta) = \theta^2$, red arrows learning rate α=0.1, blue arrows learning rate $\alpha = 0.8$

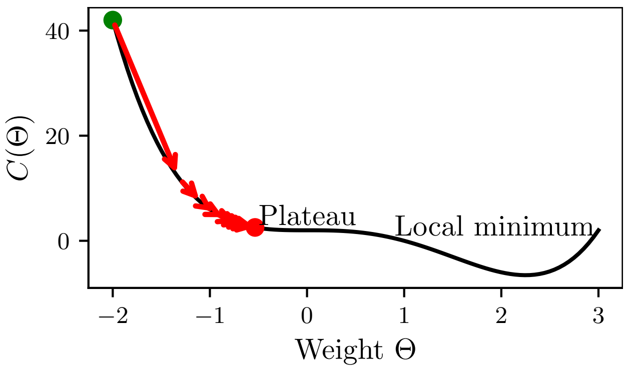


Figure 2.15: Gradient descent. Challenging optimization.

It is important to note that in gradient descent, the objective function represents the entire training set. The dataset might be divided into random batches to expedite training, a method known as stochastic gradient descent. This can lead to faster convergence and the possibility of training on larger datasets (Géron 2017). Adam is an extension of the stochastic gradient descent optimization algorithm. The name Adam stands for Adaptive Moment Estimation. The optimizer improves on stochastic gradient descent with two techniques: momentum and adaptive learning rate. Momentum uses moving averages to accelerate the movement in the relevant direction. Much like a ball rolling down a hill, gradually picking up speed until it reaches terminal velocity, the same principle applies to the momentum optimization technique. It leverages the history of gradients to compute a moving average, which speeds up the training. Using momentum helps the optimizer to get past plateaus, and it can also escape local minima (Géron 2017). The algorithm uses the momentum to calculate adaptive learning rates for each weight in the network (Géron 2017).

There exists also optimization techniques that use second-order derivatives. These methods are less prevalent in machine learning because the second-order derivatives (the Hessian) are difficult to calculate. There are n^2 Hessians for each output, while for the first-order derivatives, there are only n per output (Géron 2017). Despite this, there are second-order derivative methods that use some tricks to make it feasible. L-BFGS is one such example. L-BFGS is a quasi-Newton optimization algorithm that approximates the Hessian matrix to find the direction of the steepest descent. L-BFGS is a batch optimization algorithm that processes the entire training dataset in each iteration. For applications like PINN, using Adam optimizer first, then training with L-BFGS is recommended (Raissi et al. 2019).

2.4.3 Backpropagation

Traditionally derivatives are found by finite differences or symbolic derivatives. For finding derivatives in neural networks, a version of automatic differentiation is used called backpropagation (L. Lu, Meng et al. 2021). Automatic differentiation utilizes that all numerical computations are the sum of a finite set of operations in which the derivatives are known. The backpropagation algorithm first performs a forward pass through the NN. Then, it calculates the error between the actual output and the neural network's output. The algorithm then proceeds from the output layer to the input layer, measuring how much each connection contributes to the error. This information is then used to adjust the weights to reduce the error.

2.4.4 Approximation property of neural networks

The universal approximation theorem states that ”a neural network can approximate any smooth function arbitrarily close provided a sufficient number of neurons N_n.” Hornik et al. 1989. It has since been shown that we can do it with fewer neurons when using a deep neural network with a nonlinear activation function (Z. Lu et al. 2017). This indicates that when designing a neural network, more expressibility by using a deeper neural network instead of many neurons per layer. It is important to remember that the universal approximation theorem only states that it exists a neural network that can represent any function, not that it is possible to find the neural network parameters. Finding the right parameters includes solving a non-convex optimization. There is no guarantee that the solution found is the right one (global minima).

2.4.5 Summary

A neural network is an interesting model due to its expressive capability, well-defined optimization algorithms, and computationally efficient data libraries. A neural network is suited for computers, offering many possibilities.

2.5 Digital twin

There has been increased interest in integrating real-time data into physical models, known as a digital twin. IBM defines a digital twin as ”a virtual representation of an object or system that spans its lifecycle, is updated from real-time data, and uses simulation, machine learning, and reasoning to help decision-making” (IBM 2022). Having a virtual representation fed by live data may result in models providing real-time predictions of the flexible pipe. The idea is that the unmodeled physics will be included in the sensor data. The generated data can offer a better lifetime assessment due to improved insights. These insights can enhance the maintenance program and operational decisions, detect failure modes, and extend its lifetime (DNV-GL 2016b). A high-value asset such as a flexible pipe, reducing downtime and extending life by a few percent, can yield significant economic gains.

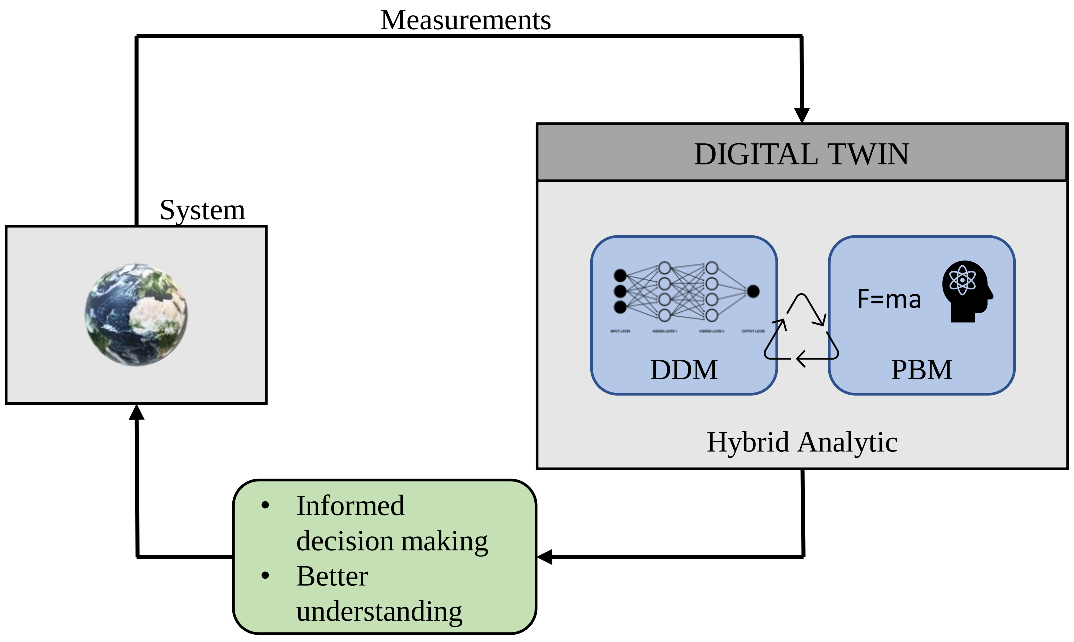


Figure 2.16: Digital Twin Schematics

A digital twin necessitates a robust model capable of utilizing the available data to provide higher resolution and insights. In Stadtman et al. (2023), the most desirable characteristics of the modeling approach include:

- **Accuracy and certainty**: Accuracy denotes the model's ability to model the "ground truth" as closely as possible, and certainty refers to confidence in its correctness. If a model delivers accurate results but relies on simplifications and assumptions, its results cannot be trusted in all cases. Hence, uncertainty studies are important.
- **Computational efficiency**: For running a model in real-time, computational efficiency is required.
- **Generalizability**: The model's ability to solve a broad spectrum of problems without the need to change the model. Theoretical models are often generalizable, e.g., for the gravitational laws $F = ma$, only a needs to be changed going from planet Earth to the moon.
- **Self-evolution** describes the model's ability to learn and improve over time.
- **Interpretability and trustworthiness**: Interpretability is the ease with which humans can understand the reasoning behind predictions and decisions. Trustworthiness indicates the reliability of these interpretations and outcomes.

- **Robustness and stability**: Robustness is related to the model's response to perturbations or noise. Stability is essential to prevent model failure or collapse.

Stadtman et al. (2023) separates into three different types of models: physics-based models (PBM), data-driven models (DDM), and hybrid models.

2.5.1 Physics-based model

PBM is based on mathematical models that utilize principles from physics to simulate and predict real-world phenomena. Since the model is founded on well-established principles and theories, it is reliable and accurate. PBM is often also generalizable to similar problems since the model is built on physics and reasoning. One of the disadvantages of PBM is that the model can be computationally intensive, especially for large systems with high fidelity. Computational intensive models limit their use in real-time models such as digital twins, and often reduced models must be used to compute the models in a reasonable time. A physics-based model is also challenging to develop and calibrate, especially for poorly understood systems. VIV is such a phenomenon; the Navier-Stokes equations that describe the forces are too computationally intensive. Thus, a semi-empirical model that combines theoretical principles with empirical data is used in practice. The advantage of a semi-empirical model is that it is easier to incorporate the observed data to improve the model by tuning the coefficient base. Since they are based on empirical data, they might capture physics that is not understood or possible to model. However, semi-empirical models are often complex and expensive to develop as they require experiments and observations, and a complete underlying understanding may not be possible. Since they are often made as simple as possible, this leads to less accuracy. In addition, the coefficients are often based on small-scale experiments, leading to uncertainties.

2.5.2 Data-driven model

DDM is purely based on available data without any underlying theoretical principles. DDM can be very accurate as the observation it is based on captures the "ground truth" instead of the physical models based on abstract theoretical principles. This makes them suitable for complex systems where we do not have a well-understood theoretical model. Modern ML has caused a revolution in the application of DDM. With DNN, complex systems can be learned only by using data. Model built upon data, it is easy to update and improve. However, the models are constructed entirely from data and observations and may lack physical interpretation due to their complexity. The DNN does not incorporate any reasoning and operates as a "black box," making it challenging to trust their outputs and generalize to similar systems. "Black box" models are a key issue, especially in safety-critical applications. Additionally, DNN need large amounts of high-quality data, which can be hard to obtain. When the system needs to exist before it is possible to collect data, DDM quickly becomes an insufficient modeling strategy. However, DDMs are often very computationally effective, an essential aspect of digital twins. It is worth noting that DDMs have increased substantially in popularity, mainly because of cheaper, better quality, and more available sensors, cameras, and data-gathering devices. The data collected can be used to develop and improve models. Many advances have been made in machine learning algorithms and neural network architectures.

2.5.3 Hybrid model

A hybrid model combines DDM and PBM, utilizing the best of both worlds. The idea behind the hybrid method is to use the knowledge we already have as much as possible and combine them with data-driven models to get the best of both models. The goal is that combining different models can ultimately lead to accurate, efficient, trustworthy, and generalizable models. In many ways, the VIVANA-TD model is a hybrid model as it combines theoretical models (finite elements and potential theory) and observations from model tests to describe the phenomena. However, further improvements could still be possible by integrating this semi-empirical model with machine learning models. This area of research goes under the name of hybrid analytics.

2.6 Hybrid analytic

Hybrid analytics is a new and rising field of research. Hybrid analytics combines physical models with machine learning. Sintef predicts that hybrid analytics to become the forefront of practical engineering (Riemer-Sørensen 2023). Combining machine learning (ML) with physical models is an extensive field containing many ML techniques and physical models. However, this thesis focuses explicitly on utilizing ML in the form of NN, narrowing the scope to this particular approach.

2.6.1 Introduction

Developing a digital twin for the riser is a good solution for the integrity monitoring of a riser. However, fatigue damage is something that happens over time. Therefore is optional to use a model that runs in real-time. Instead, the ML could be used to improve the simulation. One promising approach is residual modeling. This approach uses a simulator to calculate a prediction then the difference between the prediction and real measurement is learned using an ML model. An interesting possibility for VIV is to use multi-fidelity neural networks (NN), which combine low- and high-fidelity data. The method uses simulated data and sparse data from real measurements. The neural network is able to discover and exploit nonlinear and linear correlations between the high- and low-fidelity data, improving the prediction. By combining the strengths of physical models and neural networks, multi-fidelity NN is an interesting method for bridging the gap between real life and simulations. In a paper by Meng and G. E. Karniadakis (2020), the authors publish a neural network that can combine low and high-fidelity data and extend it to contain physics equations to learn an unknown parameter.

Another promising approach is to use a neural network as an extra term in the differential equation. This term is trained to represent unmodeled physics. This approach goes under Neural ODE or Universal differential equations (Christopher Rackauckas et al. 2021). It is also possible to use symbolic regression on the neural network to discover a symbolic equation (mathematical equation). This can help discover nonlinear models that might be difficult to discover using traditional techniques.

Much research is going into merging physical knowledge into ML models. This area of research goes under Physics-Informed Machine Learning (PIML). PIML aims to enhance the accuracy, interpretability, and generalization of models. This can provide new possibilities in modeling complex phenomena such as VIV.

2.7 Physics-informed machine learning

PIML is an approach that leverages knowledge from empirical, physical, or mathematical descriptions to improve the performance of ML models (G. E. Karniadakis et al. 2021). There are three ways of shaping the ML model using prior knowledge.

The first and simplest method is by generating training data through simulation or by preprocessing the training data and removing data that does not comply with the prior knowledge. This informing of ML is called observation bias by (G. E. Karniadakis et al. 2021). This has yielded promising results, especially in situations where a fast model is needed (Sung Wook Kim et al. 2021). After the model is trained, the result is a fast model that can run in real-time. For example, it has been used in medicine to develop an ML model to find the aortic wall stress distributions. The model was trained on finite element analysis data, and the result is that the ML model can provide instant results instead of waiting for analysis each time (Liang et al. 2018). The same approach has also been used in fluid mechanics and construction (L.-W. Chen and Thuerey 2021; Aloísio et al. 2013).

The second approach for incorporating physical knowledge into ML models involves designing customized architectures that strictly enforce physical laws. However, this method can be challenging to implement as it requires the development of a unique architecture that adds complexity to the training and optimization process (G. E. Karniadakis et al. 2021). Nevertheless, despite the difficulties associated with this approach, enforcing physical constraints can help improve the accuracy and robustness of the ML model. In the VIV case, this can be a neural network that is inspired by the modal decomposition of the Fourier series.

The third approach for integrating physical knowledge into machine learning models involves regulating the model with physical equations to ensure that the neural network converges to the solution matching the mathematical equation. When differential equations are used in the training process, the method is called a physics-informed neural network (Raissi et al. 2019). This approach is intriguing because it combines the strengths of neural networks while still adhering to known physical laws. However, it requires a prior understanding of the differential equation that describes the system. Nonetheless, this approach shows great promise in improving the accuracy and robustness of machine learning models in the context of physical systems.

2.7.1 Review of physics-informed neural network (PINN)

The concept of a physics-informed neural network (PINN) was first introduced in 2018 by (Raissi et al. 2019). However, similar approaches have been developed as far back as 1990 (Lee and Kang 1990) and 1998 (Lagaris et al. 1998). The PINN aims to solve two problems: data-driven solutions and data-driven discovery of partial differential equations. Although the method is relatively new, it is constantly being improved and refined, and new discoveries are still being made. George Em Karniadakis is one of the authors of the PINN framework, and he is currently leading a research team at Brown University that is further developing the PINN approach and have, since the original paper, produced a number of papers addressing improvements and implementations in new areas (G. Karniadakis 2023).

2.8 Physics-informed Neural Network

This chapter will present the methodology of physics-informed neural networks (PINNs). The theory is based on the general framework of Maziar Raissi, Paris Perdikaris, and George Em Karniadakis (Raissi et al. 2019). Therefore, the authors will be addressed as the original PINN authors, and the theory presented in the general framework will be considered the original PINN formulation.

2.8.1 Introduction to PINNs

Understanding a complex system through data alone requires a large quantity of it. In certain scenarios, data generation and collection can be both cost-effective and easily available. Image recognition is an example of this. Here, humans are prompted to identify specific items within images, thus creating an extensive repository of labeled data that can be used in training neural networks. With today's advanced computing power and algorithms, learning the objects in the images and interpreting them in new images is possible. However, this process of data acquisition can be more challenging. There are instances, particularly in physics and engineering, where obtaining large data sets can be challenging. For instance, gathering sufficient data from a riser might be financially or practically infeasible. Furthermore, the data collected from a riser is typically sparse, both in time and space, making it impossible to fully capture the entire system's dynamics. However, years of scientific research have expanded our understanding of dynamic systems. This knowledge is often in the form of differential equations. Differential equations illustrate the relationship between variables and how they change over time, providing a framework for understanding complex interactions. This has motivated the development of Physics informed neural networks (PINNs). PINNs integrate differential equations into the training routine of neural networks, ensuring that the model's predictions satisfy the known physical laws and principles. This makes them particularly suited to tackling problems in science and engineering where data may be insufficient, but the underlying physics is well understood. PINNs use a Neural network as a surrogate model, which is then regulated by a loss term using the differential equations and data loss term.

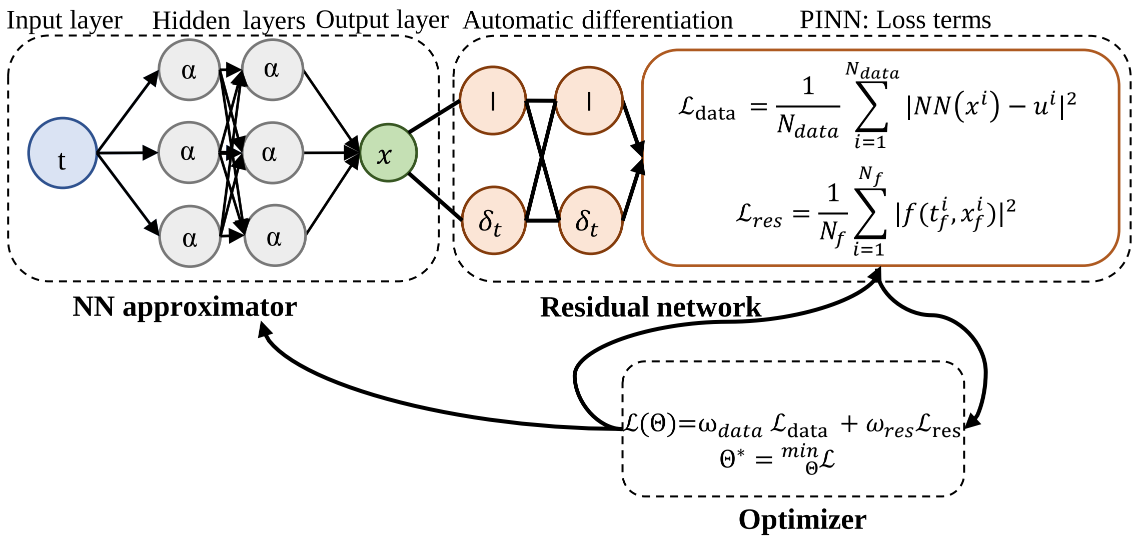
Following the formulation explained by Raissi et al. (2019). The PINN uses the universal function approximation capability of neural network Hornik et al. (1989) to represent the solution to ordinary or partial differential equations of the form:

$$\frac{\partial u}{\partial t} + \mathcal{N}[u;\lambda] = 0, x \in \Omega, t \in [0,T] \tag{2.20}$$

$$\mathcal{B}(u(t,x)) = g(z), z \in \partial\Omega \tag{2.21}$$

defined on the domain Ω with the boundary $\partial\Omega$. u(t,x) is the hidden solution, which depends on time t and a spatial variable x, λ is the parameter related to the physics, $\mathcal{N}$ is a nonlinear differential operator. $\mathcal{B}$ is the operator indicating the initial or boundary condition related to the problem, and then the boundary function g. The equation 2.20 describes the physical system that can be solved forwardly and inversely. The hidden solution $u(x,t)$ is solved forwardly, given the parameters λ and sufficient initial and boundary conditions. In the inverse problem, the unknown parameters λ are to be found together with the hidden solution $u(t,x)$from the data (Raissi et al. 2019).

Solving inversely and forwardly are two very different problems; however, there are minimal differences in implementing the PINN approach. The difference is that in the forward problem, the neural network is defined by the set of parameters Θ; in the inverse problem, the unknown physics parameters λ are included in the set of neural network parameters Θ. For both problems, the neural network is then trained with the loss function $\mathcal{L}$ describing the difference between the labeled data $\mathcal{L}_{data}$, including the initial and boundary condition $\mathcal{L}_{IC}$,$\mathcal{L}_{BC}$ and the residual loss $\mathcal{L}_{res}$. Each of the losses is weighted with a weight ω. Then an optimizer finds the parameters Θ that minimize the loss. The complete overview of the PINN structure is illustrated in Figure 2.17.



Input layer
Hidden layers
Output layer
Automatic differentiation
PINN: Loss terms
t
α
x
I
δ_t
$\mathcal{L}_{\text{data}} = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} |NN(x^i) - u^i|^2$
$\mathcal{L}_{res} = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$
NN approximator
Residual network
$\mathcal{L}(\Theta)=\omega_{data}\,\mathcal{L}_{\text{data}} + \omega_{res}\mathcal{L}_{\text{res}}$
$\Theta^* = \min_{\Theta}\mathcal{L}$
Optimizer

Figure 2.17: Schematic structure of PINN. Three blocks: Neural Network, Residual network, and then the optimizer. The initial condition and boundary conditions are included in the labeled $\mathcal{L}_{data}$ function.

The PINN can be divided into three main parts: a neural network, a residual part, and the optimizer. The neural network maps the input t to the output x in the Figure 2.17. The residual part uses the derivatives of the neural network to find the physical loss, initial and boundary loss. In Figure 2.17, the $f(t, x)$ represents the left-hand side of Equation 2.20, this means: $f(t, x) = \frac{\partial u}{\partial t} + \mathcal{N}[u; \lambda]$ Then the optimizer finds the neural network parameters Θ to minimize the loss.

2.8.2 Neural network model

The neural network is the model that is used to represent the solution. The idea is that it exists an unknown mathematical model $f(x) = u$ where f is the unknown function that takes the input x and outputs the solution u. The problem is that in most physical cases, there is no known exact function f; therefore, the problem becomes finding an approximation to f. Neural networks are popular models due to their approximation capability (Hornik et al. 1989). Since the theorem states that for a large enough neural network, any function can be approximated, the problem becomes finding the parameters for the neural network, which is a well-defined mathematical optimization problem (Chris

Rackauckas 2020). The neural network will then become,

$$NN_\theta(x) \approx f(x) \tag{2.22}$$

Other aspects that make neural networks great surrogate models: are several available high-performance computational libraries like Tensorflow or Pytorch. Furthermore, they tend to overcome the curse of dimensionality and are highly customizable for different tasks (Chris Rackauckas 2020). The neural network architecture described in the Section 2.4.1 leaves much flexibility. Even for the simple feed-forward neural network, the number of layers, neurons per layer, activation function for each layer, and weight initialization are some of the hyperparameters that need to be decided. The authors of the original framework state that for the network to perform well, it should have sufficient expressibility to express the possible solution one expects, given the differential equation (Raissi et al. 2019). However, there is little understanding of how much expressibility a neural network has. Thus there are no other options than exploring different sizes. There exist hyperparameter tuning optimizations such as AutoML (Géron 2017) that explore more efficiently than doing it randomly. In practice, choosing a "too large" model is common. It is also important to highlight the prevalence of transfer learning in these contexts. Transfer learning involves using the parameters from a similar problem that has already been solved. This approach can be highly beneficial as it can speed up the training process for the current problem. Thus, seeking similarities with previously solved issues is always advisable when facing a new problem. In the context of PINNs, the most frequently used neural network is the fully connected feed-forward with the activation function "tanh," coupled with "Glorot" initialization (Raissi et al. 2019; L. Lu, Meng et al. 2021). Choosing the correct activation function is another important consideration. The activation function must be continuous for problems involving second-order differential equations or a second-order optimization like L-BFGS. Furthermore, proper weight initialization is essential for effective training. For this thesis, all models will adhere to the original framework, utilizing "Glorot" initialization and the "tanh" activation function. The original authors acknowledged that more work is needed to identify better, customized architectures suited for PINNs. Some of the new and improved neural network architectures will be discussed in chapter 5. Input normalization is also recommended to ensure that the inputs to the neural network fall within the 'effective range' of the "tanh" function (-1,1) (Raissi et al. 2019).

2.8.3 Physics-part

The physics part constructs a new custom loss function where the residual of the differential equation is evaluated. The expression for the residual defined loss is,

$$f(x,t;\Theta,P) = \frac{\partial}{\partial t}u(x,t;\theta) + \mathcal{N}[u(x,t;\Theta);\lambda] \tag{2.23}$$

In Equation 2.23 f represents the residual of the PDE. $\frac{\partial}{\partial t}u$ is the partial derivative of the solution with respect to time, and $\mathcal{N}$ is a nonlinear differential operator. The derivative is found by automatic differentiation. The automatic differentiation provides the derivative of the neural network concerning their input. The residual can be calculated when the derivative is available at every point chosen in the space-time domain. The authors of the original framework called the residual points collocation points. It is important to notice that the loss term is separate from the labeled data loss. This means it can include different evaluation points. PINN requires the known differential equation to be satisfied at chosen points $x_i, t_i \in \Omega[0, T]$ "collocation points" or "residual points". Satisfying the differential equation means that the residual of the differential equation should be zero.

$$\mathcal{L}_{tot} = \mathcal{L}_{data} + \mathcal{L}_{residual} \tag{2.24}$$

Using the mean squared error loss function $\mathcal{L}_{residual}$ becomes,

$$\mathcal{L}_{residual} = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2 \tag{2.25}$$

The user is responsible for selecting the residual evaluation points. However, enough points must be chosen to cover the entire domain to obtain the correct solution. The original authors propose various sampling strategies, e.g., hyper-cubic sampling. There are also algorithms that select residual points automatically (L. Lu, Meng et al. 2021). Selecting too many residual points can result in high memory usage and slow training while choosing too few can result in an inability to represent the function accurately. Solving the differential equation for a large domain requires more residual points, which might lead to memory problems. However, this can be avoided by dividing the domain into smaller parts and solving consecutively. Dividing into smaller parts has also been shown to increase the performance of PINN (Krishnapriyan et al. 2021). The original paper also proposed a discrete time-stepping method that avoids the need to select residual points. To obtain a unique solution, initial and boundary conditions must be specified when solving differential equations. The solution space is infinitely large without these conditions, and any solution is equally valid. A loss term with initial and boundary conditions is included to constrain the solution space, known as a well-posed problem, to get a unique solution. The loss term is formulated to penalize the difference between the predicted and actual values of the initial and boundary conditions. This leads to a unique solution that satisfies the differential equation and the given conditions. Since in the PINN, the initial condition (IC) and boundary condition (BC) loss term is included as a soft constraint, and there is no guarantee that the solution found is the correct one due to the resulting optimization problem being non-convex, which generally does not have a unique solution. PINN requires that all the hyperparameters, e.g., network architecture, learning rate, and numbers of collocation points, be tuned to find a good solution. According to the original paper, their observation suggests that the solution is correct in most cases. The robustness of PINN is problem-specific, and some problems are much harder to get satisfactory results. PINN also has the advantage over traditional solvers because it can find the correct solution to an ill-posed problem by using data points to shape the solution. This is possible because PINN can learn the underlying dynamics of the system directly from the data without knowing the initial or boundary conditions. This approach is beneficial when the data is sparse or noisy.

The additional loss term comes with its problems: the optimization problem becomes increasingly more challenging. Optimizing the two terms simultaneously can lead to several problems, especially with vanishing gradients. This is likely because the two problems have different scales leading to more challenging optimization landscapes. This can lead to the two terms being sensitive to different hyperparameters, such as learning rate, increasing the difficulties in finding optimal parameters for the problem. The result is that problem is much more prone to end up in local minima. The original paper only suggests the correct weighting of the loss term to overcome this problem. However, later research suggested new setups to overcome these optimization issues. Some of the new approaches will be explained in chapter 5.

2.8.4 Optimizing

The process of determining the neural network parameters Θ is called training. The objective of the training is to find the weights that minimize the loss. Several papers, including (Raissi et al. 2019) and (L. Lu, Meng et al. 2021), suggest a two-step training process: initial training with the Adam optimizer and further training with the L-BFGS optimizer. The Adam optimizer avoids local minima and approximates a good solution, after which L-BFGS refines the weights further. This approach has proven effective in tests conducted for this study, particularly when identifying unknown parameters in the differential equation. Utilizing L-BFGS after Adam consistently increased the accuracy. However, experience in this thesis is that the L-BFGS optimizing tends to be considerably slower than the Adam optimizer.

2.9 Summary

VIV is a complex phenomenon influenced by numerous parameters. The VIV model has been improved and evolved, but it cannot accurately describe the real phenomena due to fundamental assumptions and simplifications. Recent years have seen immense popularity in DDM, especially DDM based on DNN. Due to DNN's excellent pattern recognition and function approximation capabilities, there has been success utilizing this in the industry. With technological advancements, digital twin applications have emerged, requiring efficient, flexible, and accurate models. Hybrid analytics, combining the strengths of ML and PBM, offer a promising approach for digital twins. The end goal is digital twins driven by hybrid analytics, leading to better predictability, understanding of the system, and practical and optimized systems. PINN provides a seamless integration of neural networks and physical models, facilitating the achievement of these goals.

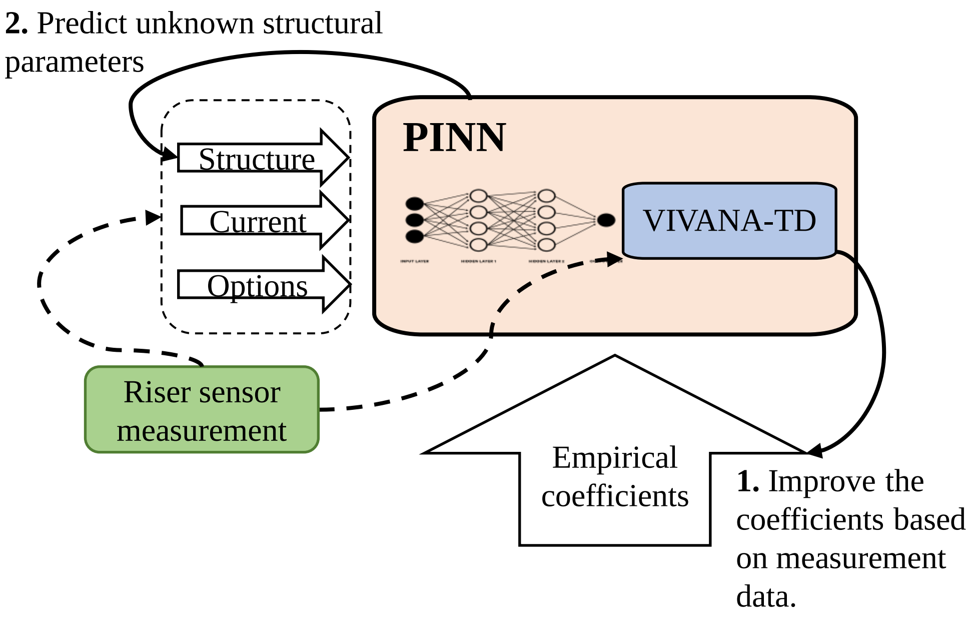
Chapter 3

Problem setup and implementation

In this chapter, the proposed concept will be explained. Then, two simplified cases will be described to assess the feasibility of the concept. The first case is a forced mass-spring-damper system. The second case is a mass-spring system with force data obtained from the semi-empirical VIVANA-TD model. Lastly, the code implementation will be explained.

3.1 Proposed concept

Due to the challenging operational conditions discussed in Section 2.1, vibration sensors are typically scarce on risers. If they exist, they are usually placed in practical locations rather than high-stress areas (Sundararaman et al. 2018). This means there is a need for a framework that can utilize the sensor data given the arbitrary placement and quantity of the sensors. The idea is that using the PINN formulation can solve the equation forwardly while adjusting the empirical data and estimating the unknown parameters based on the available data. This approach utilizes the best of physical and data-driven modeling. The final model is a NN. However, the physical model has to be satisfied for the physical residual of the NN to become zero. In the inverse problem, the estimated parameters provide insight into what solution is learned. Thus some of the physical understanding from the models is kept. Figure 3.1 gives an overview of the PINN approach.



2. Predict unknown structural parameters
PINN
Structure
Current
Options
VIVANA-TD
Riser sensor measurement
Empirical coefficients
1. Improve the coefficients based on measurement data.

Figure 3.1: VIV PINN concept

This approach could work by recording data for three hours, then using the data to find an accurate prediction of the displacement for the whole riser. One apparent downside is that the already-developed numerical solvers are not used. Furthermore, since PINN is a relatively new approach, the method needs are not fully understood, and there needs to be more theory about stability and robustness. Nevertheless, the method offers a flexible approach, and additional terms can be included in the loss function to regularize the model and ensure convergence to the correct solution. There are also possibilities for customization of the NN to represent a riser's displacement better. For example, a NN utilizing modal decomposition can be employed (Raynaud et al. 2022).

3.2 Case 1: Forced mass-spring-damper

3.2.1 Equation and numerical solution

The primary objective of the first case is to validate the functionality of the baseline PINN. Solving this problem will confirm that the implemented PINN code functions as intended. Despite its simplicity, a mass-spring-damper system is interesting due to its presence across various dynamic behaviors. For example, this system is frequently used to model structural vibrations. The system is represented in figure 3.2.

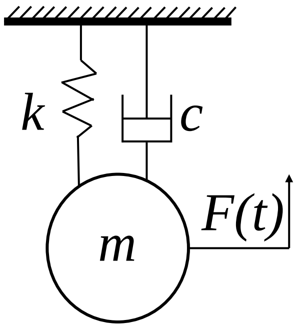


Figure 3.2: Case1: forced mass-spring-damper system

Assuming linear spring, $f_r(t) = ku(t)$, linear viscous damping, $f_d(t) = c\dot{u}(t)$, and system with constant mass, $f_I(t) = m\ddot{u}(t)$. The equation of motion of the system becomes:

$$m\frac{d^2u}{dt^2} + c\frac{du}{dt} + ku = F\sin(\omega t) \tag{3.1}$$

Initial conditions:

$$u = u_0, \dot{u} = \dot{u_0} \tag{3.2}$$

where m represents the mass, c represents the damping coefficient, k represents the spring constant, u represents the displacement of the mass from its equilibrium position, t represents time, F represents the amplitude of the forcing function, and ω represents the angular frequency of the sinusoidal excitation force. This equation describes the motion of a mass attached to a spring and a damper, subject to a periodic external force. It is a second-order ODE involving only the second derivative and the first derivative with respect to one parameter, in this case, the displacement with respect to time. The equation can be solved by expressing it as two first-order ordinary differential equations. To obtain a unique solution, the initial condition must be satisfied. The numerical solver ODEINT from SciPy (n.d.) is utilized to find a solution u^*. Numerical solvers are beyond the scope of this thesis, and the numerical solution will be referred to as the exact solution u^*. The second-order linear differential equation is expressed as two first-order linear ordinary differential equations:

$$\begin{aligned} \dot{u_1} &= u_2 \\ \dot{u_2} &= \frac{1}{m}(F(t) - cu_2 - ku_1) \end{aligned} \tag{3.3}$$

c	0.06
m	0.01
k	0.5
F_A	1
ω	15

Table 3.1: Case1 : Parameters

The solution from $t \in [0, 2]$ with the parameters given in the table Table 3.1 is plotted in figure 3.3

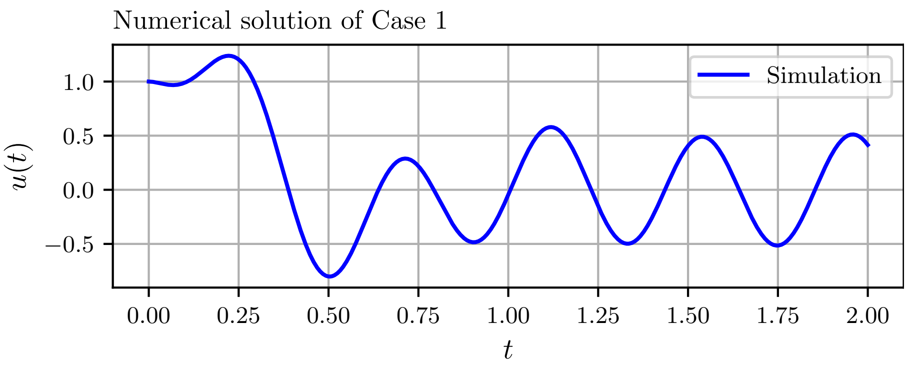


Figure 3.3: Case 1: Numerical solution to equation 3.3 on the interval $t \in [0, 2]$.

3.3 Case 2: VIVANA-TD

In case 2, force and displacement data are collected from the simulation software SIMA (Sintef n.d.), utilizing the VIVANA-TD equation. The results from the simulation program are used in conjunction with the VIVANA-TD equation within the PINN model. SIMA simulates "real measurement data" for the PINN model to identify unknown parameters in the equation. This case study resembles Kharazmi, Fan et al. (2021), with the difference that the data is obtained from the semi-empirical simulation program SIMA instead of CFD calculations. The objective is to identify the coefficient that best describes the measured data while removing uncertainty associated with parameter selection. Additionally, the problem is simplified by considering a stiff cylinder, eliminating the need to model and include spatial equations for the beam. By disregarding the space dimension, the problem transitions from a PDE to an ODE.

SIMA

The SIMA software (Sintef n.d.) is used for simulating the model. The model calculations are done using RIFLEX 4.44, which utilizes VIVANA-TD equations to calculate the hydrodynamic force. The model configuration is depicted in Figure 3.4, consisting of two nodes fixed in all directions except the z-direction. A linear spring is attached to both nodes in the z-direction, and a rigid pipe is connected between the nodes. Movements are initiated using only current. The parameter values for the model are provided in Table 3.2.

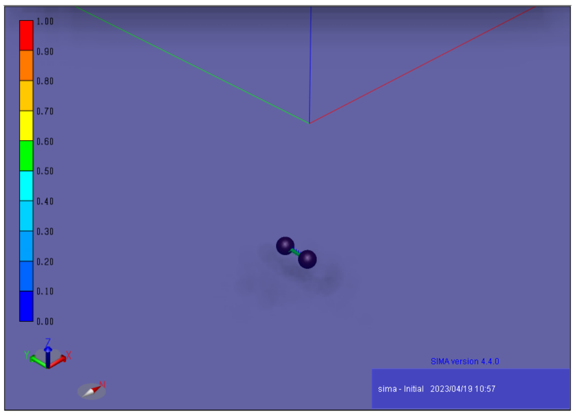


Figure 3.4: SIMA model

Length	1 m
Radius	0.1 m
Mass	13.05 kg
Spring stiffness	598.6 N/m
Current velocity	1 m/s

Table 3.2: SIMA parameters

The resulting displacement and force collected from SIMA are displayed in Figure 3.5.

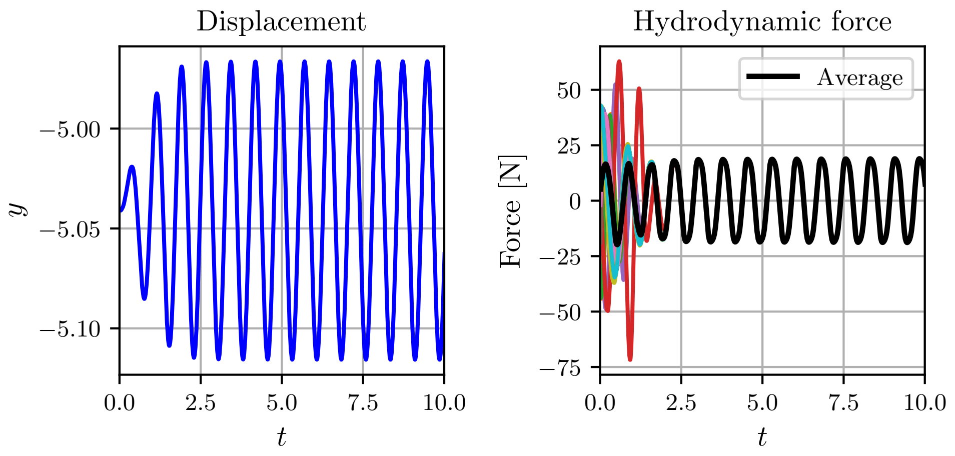


Figure 3.5: VIVANA-TD simulation, with the parameters described in Table 3.2

In its computations, SIMA (Sintef n.d.) divides the pipe into smaller parts to get the displacement along its length. While the flexible pipe is modeled with high bending stiffness, resulting in consistent displacement, the hydrodynamic forces initially vary across different sections of the pipe due to the calculation method. Finally, the average force acting along the cylinder is computed to represent the overall force used in the 1D problem accurately. This is illustrated left in Figure 3.5.

Equation

Assuming that the force term is always available, the resulting residual differential equation becomes:

$$\text{Residual} = (m + A * rho) + k * z - F_{hydrodynamic} \tag{3.4}$$

Given that the force data is only available at discrete steps from the simulation, second-order spline interpolation has been used to determine the force at all timesteps. Combining this equation with the simulation data yields the following results.

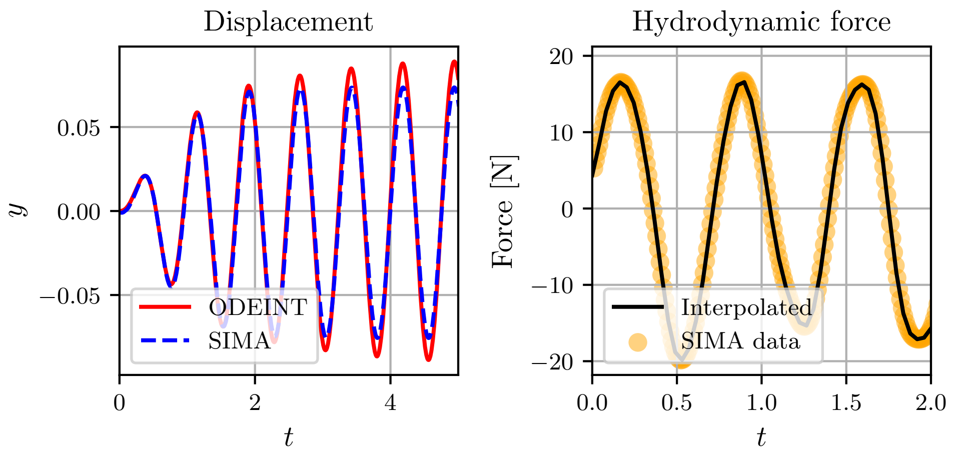


Figure 3.6: Left: SIMA simulation compared to the numerical solution using force from SIMA and equation Equation 3.4. Right: The force from SIMA, data points, and interpolated result

On the left in Figure 3.6, the numerical solution differs from the solution derived from SIMA. These solutions should match perfectly. However, the intricacies of the VIVANA-TD simulation and numerical solver fall outside the scope of this discussion, so the discrepancy is, for now, neglected. When viewed from another perspective, this discrepancy isn't necessarily all negative, as real-world equations often don't perfectly match the data, yet they're usually successful in capturing the main movements. The experiment can be extended by decomposing the hydrodynamic force into the terms from the VIVANA-TD (Equation 2.12). Then it is possible to incorporate the empirical coefficients as unknown parameters and use the PINN method to "fine-tune" the coefficients, for example, C_D from Equation 2.12. There are also possible to estimate the other coefficients from the Equation 2.12. However, C_{v_CF} requires the relative phase angle between the structural velocity and the vortex shedding force $\phi_{v,CF}$ in the term, further complicating the setup. The goal is to have a solution that matches the training data, and the empirical parameter is estimated such that the physical equation residual is also zero.

3.4 Implementation

There are multiple frameworks for PINNs. The original PINN framework (Raissi et al. 2019) is implemented in Python using TensorFlow 1.0. Since the initial publication, multiple high-level libraries have been created to limit the need for low-level implementation of a model in TensorFlow 1.0, including open and popular libraries such as DeepXDE (L. Lu, Meng et al. 2021), Nvidia Modulus (Nvidia 2023), NeuralPDE (Zubov et al. 2021), and SciANN (Haghighat and Juanes 2021). These libraries provide a simpler and more efficient way to implement PINNs. Despite the existence of these libraries, this code has been implemented in Python using Tensorflow 2.0. TensorFlow 2.0 has been chosen to have better control and understanding of the underlying processes. TensorFlow 2.0 is a more widely-supported framework than TensorFlow 1.0, and, most importantly, it enables more transparent and comprehensible code. Pierre Jacquier's and the mentioned libraries have been used for inspiration in implementing the code in TensorFlow 2 (Jacquier 2019). The code is implemented in an object-oriented setup utilizing three distinct classes, each representing one of the building blocks of the PINN. This modular approach offers greater flexibility when tackling various case studies, as each component can be modified independently.

The overview of how the code is implemented is presented in Figure 3.7

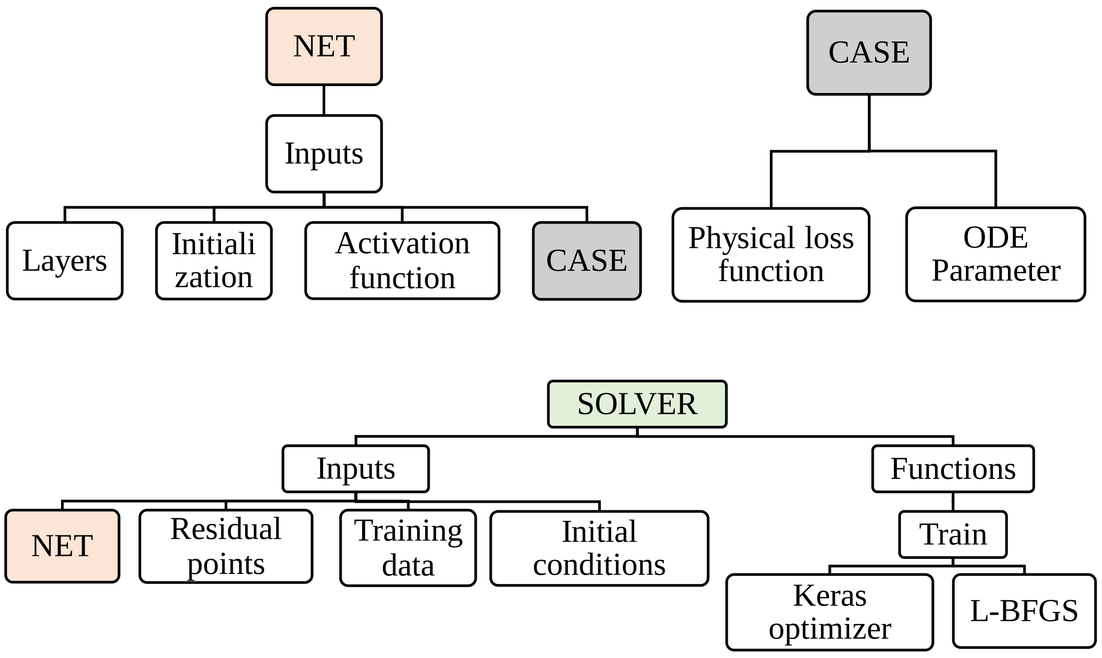


Figure 3.7: Code structure. Main classes and functions.

Dividing the code into modules facilitates the addition of new functions. However, it requires a more general approach to coding.

NET

The NET class is a sub-class of the $tf.keras.Model$ (Tensorflow 2023) class is responsible for constructing the NN. When creating an instance of the NET class, the user specifies a list of layer sizes, an activation function, and a weight initialization method. Using Keras sub-classing (Tensorflow 2023), the NN is then initialized. This approach provides flexibility for customizing the network and allows for additional trainable parameters to be added directly into the network without defining a custom layer. The NET class implements feed-forward NNs, but it can be easily adapted to other network architectures without modifying the other classes. The NET class has a function that can append a normalization layer to the NN. For the experiments in this project, a standard NN with four layers and 20 neurons is used (see Table 3.3). This configuration has been found to yield good results, although different network sizes were also considered (see Appendix A for details). It is important to note that further research could be conducted to find the optimal NN. However, it is generally safe to choose an over-parameterized NN as it has no significant consequences other than training (Géron 2017), and by having a physical loss term, this will act as a regularizer against overfitting. The activation function is chosen as ”tanh” based on the Raissi et al. (2019) paper, and the initialization method is set to ”Glorot” (Glorot and Bengio 2010).

Number of Hidden Layers	4
Number of Neurons per Layer	20
Activation Function	Tanh
initialization	Glorot uniform

Table 3.3: Hyperparameters, baseline NN

The NN is now a function with over 1000 parameters (weights and biases) that needs to be adjusted to find the approximation to the solution.

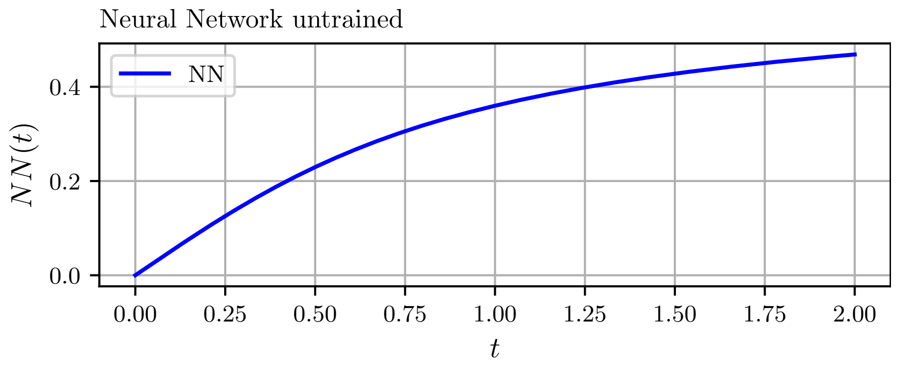


Figure 3.8: Glorot uniform initialization, tanh activation, 4 hidden layers of 20 neurons each

Figure 3.8 illustrates the untrained neural network (NN) model predicting the solution from $t \in [0, 2]$. The predictions of the untrained NN will differ for each initialization due to the randomness in the "Glorot initialization". Thus, the training performance and result will also differ for each NN initialization.

Optimizer	Adam
Epochs	10000
Learning rate	0.001
Residual loss weight	1.0
Initial condition loss weight	1.0
Data loss weight	1.0

Table 3.4: Standard training parameters, the forward problem

Table 3.4 shows the standard training for the forward problem. For inverse problems, the standard is to follow up with L-BFGS until convergence.

CASE

For each specific problem, there exists a corresponding class that inherits from the NET class. This derived class includes additional components such as the physical loss equation, initial loss, and the unknown parameters for the physical loss equation. In the case of the VIVANA-TD problem, the force data is provided as a list of specific residual points. The physical loss function takes a list of residual time points as input and calculates the residual loss at the selected points, returning a list of these losses.

SOLVER

This is the class implemented to find the correct parameters for the neural network. The inputs are the Keras neural network, residual loss points, training data, and initial conditions. The solver can train the NN using a Keras optimizer or L-BFGS. Due to the L-BFGS optimizer not being supported in TensorFlow 2.0, it has been implemented using the tutorial by Pychao (Chuang 2019). Sometimes while training with the Adam optimizer, the loss suddenly gets worse. Therefore, the best loss and corresponding weights will be saved when training with the Keras optimizer. After the Keras training is finished, the best weights will be set to the NN.

The Solver class has also been implemented with post-processing tools. Plotting the loss history, unknown parameters, and solution. The code can be found by following the link in appendix B.

Chapter 4

Result from baseline PINN

In this chapter, the results of the baseline PINN model will be presented and discussed. The analysis will begin by investigating and discussing Case 1, followed by Case 2. The details and descriptions of these cases are found in chapter 3.

4.1 Case 1: Forced mass-spring-damper system

4.1.1 Neural network

To demonstrate the expressibility of the baseline NN described in Table 3.3. The NN is trained on labeled samples with a fixed timestep from the solution. The labeled data is sampled from the numerical solution, with 20 points sampled at a fixed time step. The NN is then trained with 4000 iterations using the Adam optimizer, with a learning rate of 0.001.

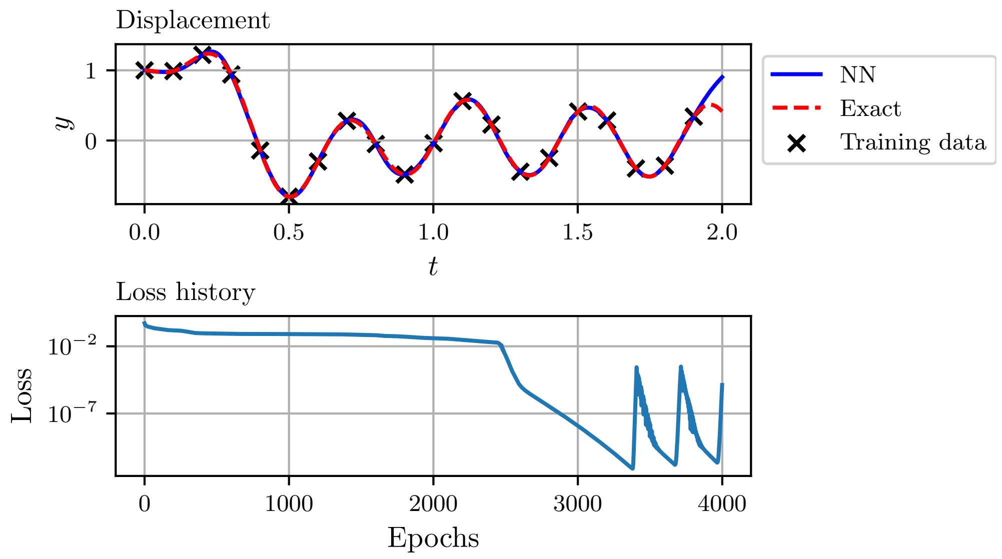


Figure 4.1: Case 1: NN. Top: The displacement prediction from 0 to 2 seconds. 20 training data points were sampled from the exact solution. Bottom: Training loss history. 4000 Epochs optimizer Adam with learning rate=0.001.

Figure 4.1 verifies that the baseline NN can learn the mapping from the input (t) to the output (y) when provided with sufficient training data. Figure 4.1 shows that the NN's expressibility to represent the solution is satisfactory. However, if the solution needs to be obtained for a larger time interval, it may be necessary to increase the NN size to adequately represent the solution. During training, the NN's weights are optimized to minimize the loss between the training data, as shown in Figure 4.1. As a result, the NN's predictions align with the training points, and the approximation also matches the solution between the training points, which is equally important in practice. With the significant amount of training data available in this case, the NN is expected to capture the solution accurately for the entire domain.

It's important to note that the NN's accuracy outside the training domain is not included in the loss function. Hence, there is no guarantee that the NN's prediction matches the solution between the training data. To address this issue, some training data can be set aside as validation or test data. Training can stop when the loss for this validation or test data is low, indicating that the NN is adequately trained. Continuing to train beyond this point can lead to overfitting. The NN's training is limited to the interval between 0 and 2, and it cannot extrapolate beyond this range. Hence, dealing with sparse or limited data presents a challenge.

4.1.2 PINN Forward problem

To increase flexibility, the NN will be trained using the differential equation and initial condition without needing labeled training data. This approach offers greater flexibility, as the model can operate within the desired time domain, unlike traditional training methods, which require input-output pairs. By including a term that describes the difference between the prediction and the equation, the model will not have issues with overfitting. The initial condition must also be included such that the correct solution is found. The total loss function is then defined as follows:

$$\text{loss} = \omega_{IC} L_{IC} + \omega_{res} L_{res} \tag{4.1}$$

$$L_{IC} = (NN(0) - y(0))^2 + (\frac{dNN(0)}{dt} - \dot{y}(0))^2 \tag{4.2}$$

$$L_{res} = \frac{1}{n} \sum_{i=1}^{n} (\frac{d^2NN(t)}{dt^2} m + \frac{dNN(t)}{dt} c + NN(t)k - Fsin(\omega t)) \tag{4.3}$$

In Equation 4.1 ω_{IC} and ω_{res} represent the initial condition weight and ODE residual weight, respectively. In Equation 4.3 F represents the external force amplitude, ω is the angular frequency, and t represents the chosen collocations points. The collocation points are chosen in the interval of $t \in [0, T]$, and the number of collocation points is denoted n. Unlike traditional NN training methods, in this model, no labeled training data is included in the loss function. In order to ensure that the solution satisfies the differential equation at each point in time, the NN residual from the differential equation is evaluated at the chosen collocation points within the interval. The collocation points can either be randomly or regularly spaced. The NN output is then used to calculate the residual of the differential equation at each collocation point which is then evaluated in the loss function. The result is a mapping from the NN to the analytical solution in the time domain of $t \in [0, T]$. Thus, the PINN method results in a NN representing an approximation to the analytical solution on the time domain $t \in [0, T]$. The method can solve differential equations for a wide range of problems with varying complexities. Using

50 fixed collocation points and training for 10000 iterations of Adam optimizer with a learning rate of 0.01, the results are shown in Figure 4.2.

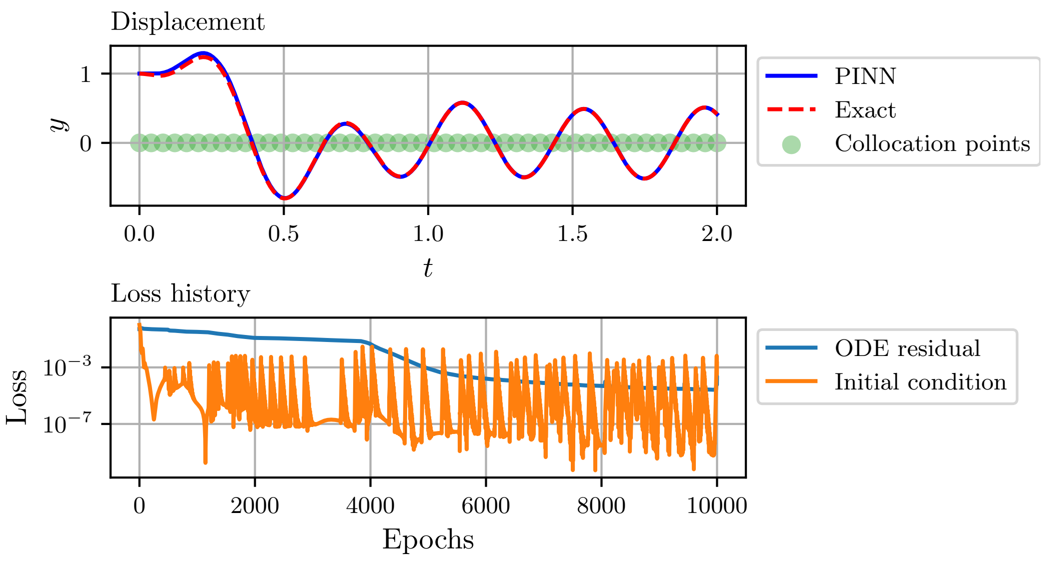


Figure 4.2: Case 1: PINN. Top: displacement y computed with PINN is compared to the numerical solution (Exact). Bottom: the training history illustrating the minimizing of the different loss components for each epoch.

Figure 4.2 shows that the PINN approach can find the correct solution given only the initial conditions. In contrast, the standard NN requires labeled data to learn functions. However, due to the added complexity of having two terms in the loss function, the PINN requires more training iterations compared to the traditional NN, as evident from the loss curve in Figure 4.2. In particular, the PINN takes around 5000 iterations to reach a loss of 10^{-3}, whereas the traditional NN reaches the same loss after 1500 iterations. This observation aligns with the findings by Raissi et al. 2019.

Optimizing using the PINN is more difficult due to the extra complexity of having multiple objectives. Occasionally, the optimizer may concentrate on minimizing one loss before satisfying another, causing the optimizer to find a local minimum instead of a global one. For example, the residual loss gets minimized before the correct initial conditions. In that case, the model might be unable to change to the correct initial conditions since correcting the initial condition will lead to a larger residual loss. However, this can be improved by weighting the initial condition more than the residual loss. Most PINN-related papers, including the original framework (Raissi et al. 2019) and DeepXDE (L. Lu, Meng et al. 2021), manage good results by weighting the residual loss lower and using the Adam optimizer, followed by the L-BFGS optimizer. Despite the advantage of PINNs learning function solutions without labeled data, the optimization is slower, more difficult optimizing and has a higher number of training iterations compared to traditional NNs.

4.1.3 Comparing PINN with normal NN

Traditional NNs are highly effective at solving a wide range of problems, but they require large amounts of labeled data to achieve high levels of accuracy. In contrast, PINNs have been developed specifically to leverage both input and output data, as well as differential equations, in order to solve problems more efficiently. This makes them particularly well-suited for addressing physical problems where observation data is often sparse or difficult to obtain. By defining a standard NN and a PINN and training each with the same set of labeled data. The results are illustrated in Figure 4.3.

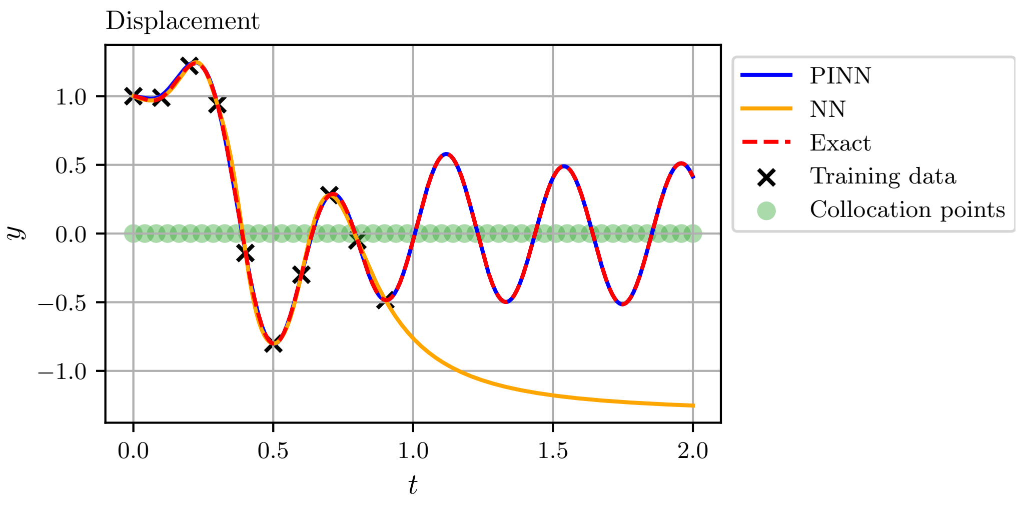


Figure 4.3: Case 1: PINN vs NN. Training points only cover half the time domain.

Figure 4.3 shows that PINNs can extrapolate beyond the available labeled data, while traditional NNs fail to produce accurate results when presented with limited data. Integrating prior knowledge as a differential equation enables the PINNs to be better equipped for solving complex physical problems where the underlying principles are known, even in scenarios with sparsely labeled data. The limitations of a standard NN in extrapolating data are clearly highlighted in Figure 4.3. A PINN operates similarly to a numerical method, capable of solving problems within the desired space. Furthermore, the accuracy of a PINN can be adjusted by selecting the number of residual points, modifying training parameters, and opting for a larger NN. Figure 4.3 shows how the PINN can seamlessly combine the differential equation and measurement points. In a scenario where the differential equation and measurement points differ, a method is needed to decide what should be prioritized. Typical measurement data and initial and boundary conditions are prioritized. Thus to optimize the PINN approach, the loss function should be balanced (4.4). The incorporation of the loss balancing weights further expands the already extensive list of hyperparameters. However, PINNs offer a powerful and promising approach to solving physical problems where data is sparse or incomplete and represent a significant step forward in developing ML techniques for practical engineering applications.

$$\text{loss} = \omega_{IC} L_{IC} + \omega_{res} L_{res} + \omega_{data} L_{data} \tag{4.4}$$

4.1.4 Inverse problem

In the inverse problem, one or more of the parameters in the differential equation are unknown. These unknowns are incorporated as trainable variables within the PINN model and are identified concurrently with the weights and biases of the NN. This approach offers no guarantee of finding the correct parameters, as the NN may converge to a solution that satisfies all the labeled data and a differential equation that aligns with the collocation points. However, according to Raissi et al. 2019, the correct parameters are usually identified given sufficient data points. In order to test the capabilities, different tests will be considered.

Two unknown coefficients, no noise

To illustrate this approach, the same training data as in Figure 4.3 is used in training, but the damping coefficient c and the stiffness k are unknown in the differential equation. The NN was initially trained using the Adam optimizer and further refined with the L-BFGS.

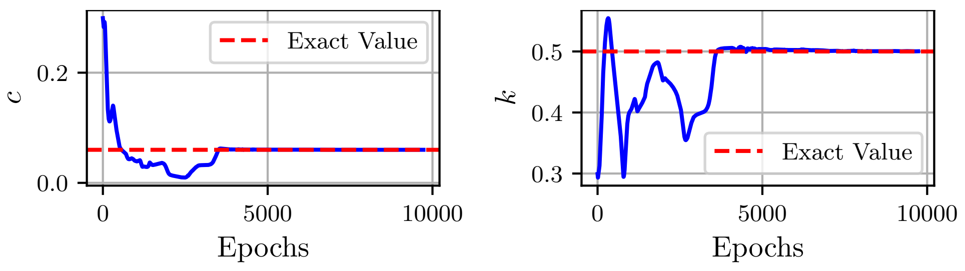


Figure 4.4: Case 1: Parameter identification. The evolution of c and k

Figure 4.4 shows how the estimated values for c and k. The model successfully identifies the correct parameters for the differential equations. This is expected, given the substantial amount of training data available. To further investigate the capacity of the method to solve inverse problems, random training data is selected with varying levels of noise and data quantity.

Randomly selected points, varying levels of noise, and data quantity

Training data were randomly sampled from the interval $t \in [0, 2]$ for the subsequent tests. Each variation of the number of points and the noise level is run three times. This is because the location of the data points considerably affect the value it has for parameter identification. In this test, only the c parameter is unknown. The results are shown in Figure 4.5.

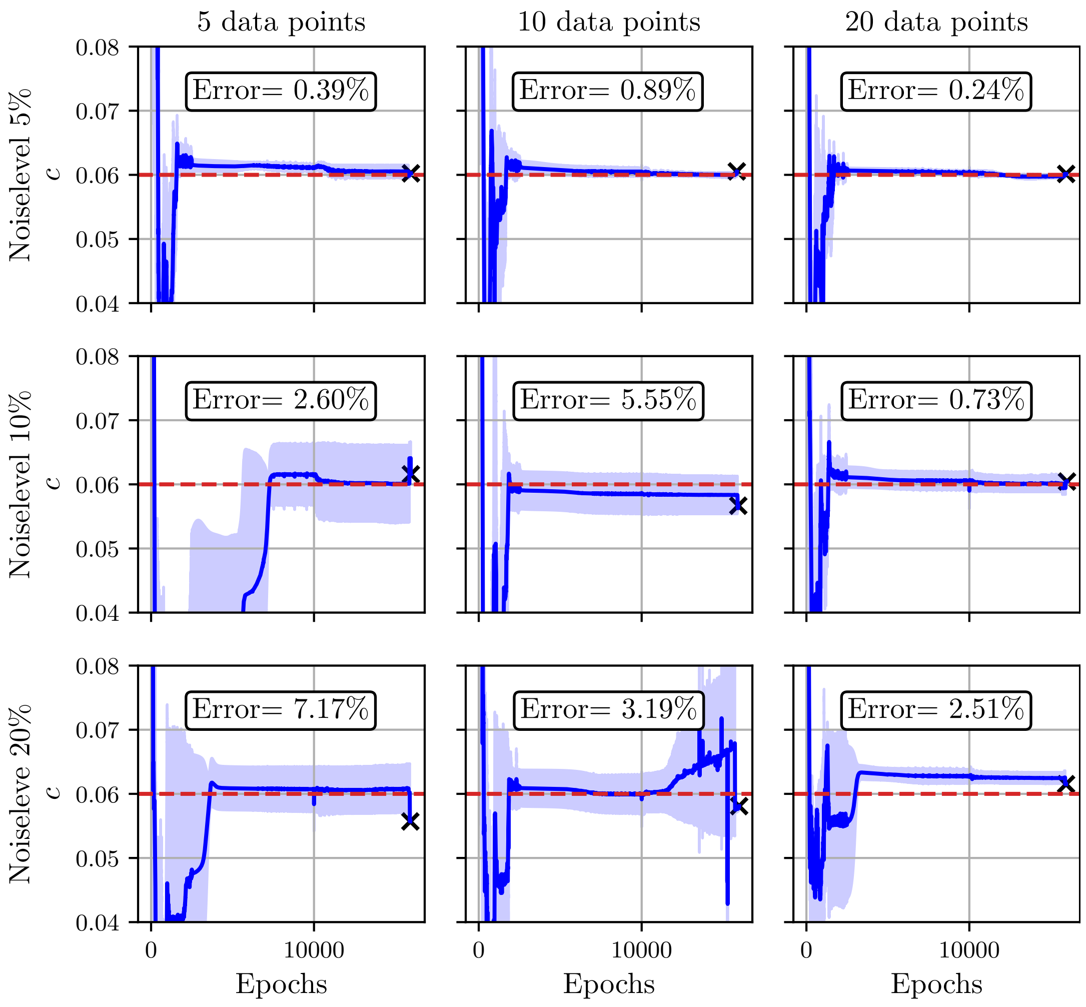


Figure 4.5: Case 1: Parameter identification: The light blue illustrates the standard deviation, and the solid blue illustrates the average for the three iterations. The black cross is the final average prediction

The results in Figure 4.5 demonstrate that the PINN is robust to noise and can accurately determine the exact value. With 5% noise, the model can accurately identify the c value, even with only five randomly selected points. The model successfully identifies a close approximation to the actual c value for all iterations involving random points. This trend continues even with variations in noise and data quantity. However, as noise increases, the average becomes increasingly inaccurate. Increasing the number of data points reduces the standard deviation and the number of iterations required for the method to locate the accurate values. The final value also improves with the increasing amounts of data points. This highlights the method's strength in accurately identifying the value, even with sparse and noisy data points. These findings underscore the robust characteristics and utility of the PINN approach.

Two unknowns with varying amounts of noise

Examine the scenario when both k and c are unknown. This analysis involves a single iteration with unknown coefficients.

Noise level / Number Of Points	2%	5%	10%	20%	30%
5	59.40%	111.23%	6.39 %	18.09 %	13.98%
10	**1.62%**	**0.59%**	4.81%	17.62%	9.65%
20	**2.45%**	5.21%	3.98%	6.70%	24.36%
50	**1.49%**	**1.20%**	3.83%	12.58%	**1.33%**

Table 4.1: The result is the average error percentage for k and c. Bold text indicates the best results.

Table 4.1 presents the average percentage error for k and c. The results with two unknowns vary substantially more than those with only one unknown. This is likely because several combinations of k and c closely align with many of the data points. The data quantity plays a more significant role here, as it is evident that 5 data points are too few to identify the two coefficients accurately. However, the results improve substantially for 10, 20, and 50 data points. Interestingly, the method performs well for low noise levels and 10 or more data points but struggles to find the correct solution with a noise level of 10% or more. However, with 50 data points and a 30% noise level, the model finds a very accurate solution. The results suggest that the placement of data points plays a substantial role. More iterations are necessary to identify clear trends and draw a conclusive interpretation.

4.1.5 Simplified physical model

While models may not always match real-world measurement data, it is interesting to investigate the impact of using a simplified model in the PINN model. Thus the PINN will be incorporated with a simplified force model. The simplified force term is defined as follows:

$$\text{Force}_{Simplified} = 0.5 \cdot \text{Force}_{True} \tag{4.5}$$

Real measurement data is obtained by sampling from the true solution and implementing these samples in the PINN model. This experiment aims to determine whether the PINN model, combined with a simplified equation, can outperform the simplified equation when integrated with sampled data points. To do this, the PINN will be compared to the simplified solution obtained by a numerical method. The investigation consists of two parts. One has a fixed truncated physical model, and the other by having the F as a trainable variable in the PINN. The first investigation aims to determine if the PINN approach can account for unmodeled physics. The goal is to find out if the measurement data improves the simplified model while keeping the characteristics of the simplified model. The second investigation will determine if PINN can solve forwardly while inferring/calibrating the uncertain parameter.

Fixed physical equation

The first investigation will start by using fixed-time step sampling together with the simplified model. The goal is to investigate the behavior when combining a simplified model with measurement data in the PINN approach.

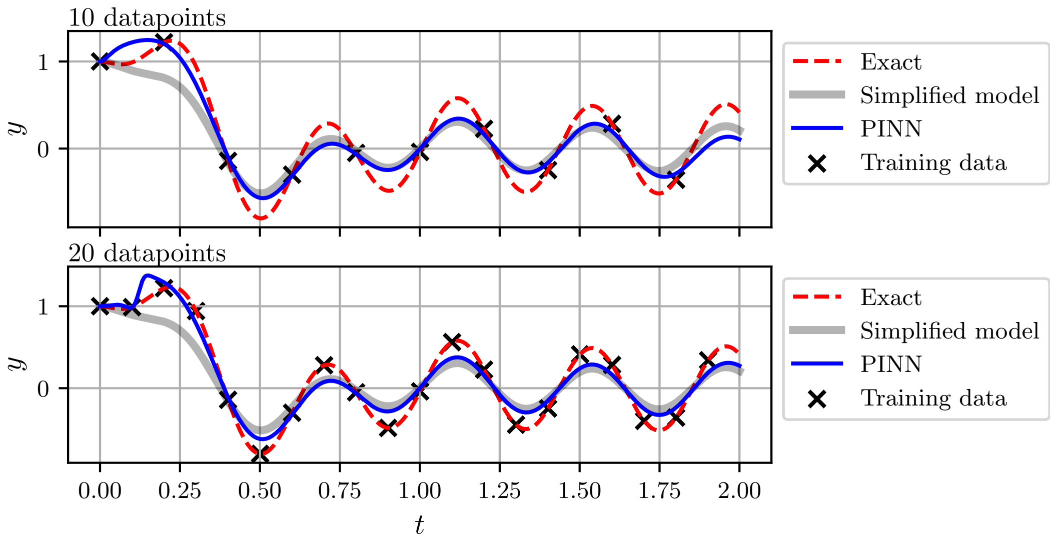


Figure 4.6: Case 1: Simplified model. Fixed time step sampling. Equal loss weights.

In Figure 4.6, the simplified model is the numerical solution using the simplified force term described in equation 4.5 (F=0.5), the exact is the numerical solution to the case 1 Equation 3.3 with F equal to 1, the PINN is the prediction of the PINN model after training. The plots in Figure 4.6 for 10 data points show that the PINN model is very close to the simplified model. This outcome is expected as the training data is situated in areas where the simplified and true solutions are almost indistinguishable. For 20 data points, the PINN prediction is still very close to the simplified model. Here several data points should be able to provide value resulting in more accurate prediction. However, this is mainly due to equal weighting, meaning that the residual loss is equally important as the training loss. Therefore, the residual weighting is set to 0.1 to prioritize the observed data the most, and the same training iteration is run.

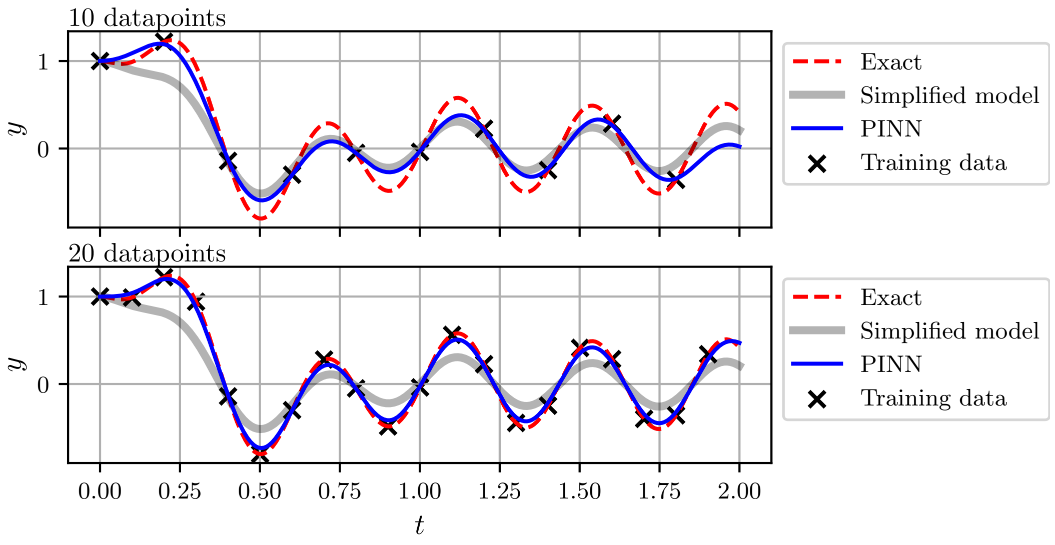


Figure 4.7: Case 1: Simplified model. Fixed time step sampling. Residual loss weight 0.1.

Figure 4.7 illustrates that applying more weight to the training data improves the performance of the PINN in satisfying the training points. With only 10 training points, the model displays noticeable improvement initially, followed by very similar performance as the simplified model, and eventually, the model becomes more inaccurate than the simplified model. However, the model can still predict a solution that conforms to the data points while preserving the physical model characteristics. On the other hand, by using 20 training points, the model is able to precisely represent the exact solution, leading to a significant improvement over the simplified model. It is important to note that, in actual situations, data may be randomly selected instead of equally sampled. Therefore, the same comparison with 20 and 10 randomly selected data points is done.

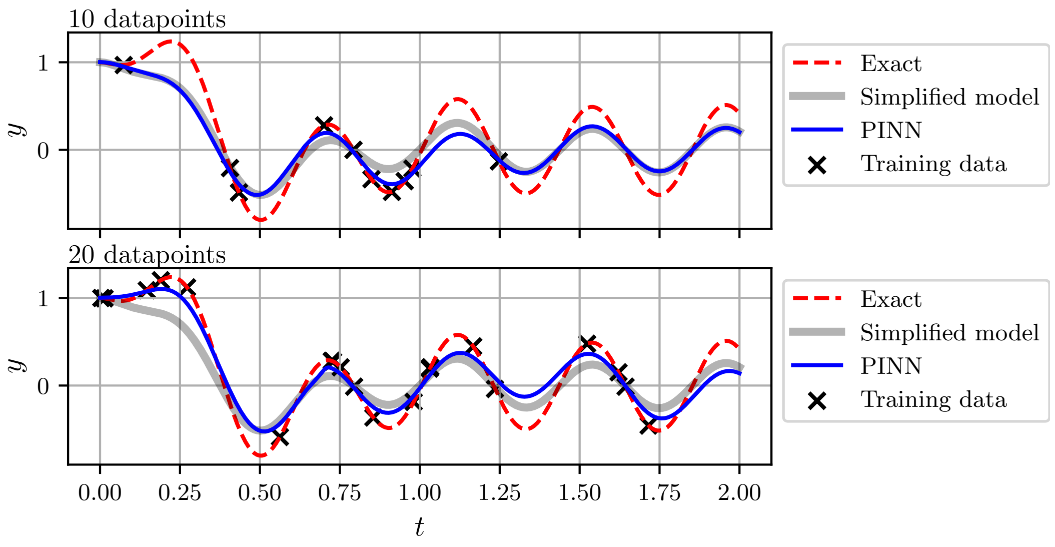


Figure 4.8: Case 1: Simplified model. Randomly sampled. Equal loss weighting.

Figure 4.8 shows that for 10 data points, the model improves when t is between 0.75 and 1. For the rest of the prediction, the model is very close to the simplified model. However, for 20 random points, the model solution is very close to the simplified model and cannot improve over the simplified model. Leading to the same observation as with fixed sampling: the data points should be weighted higher to improve the model's accuracy.

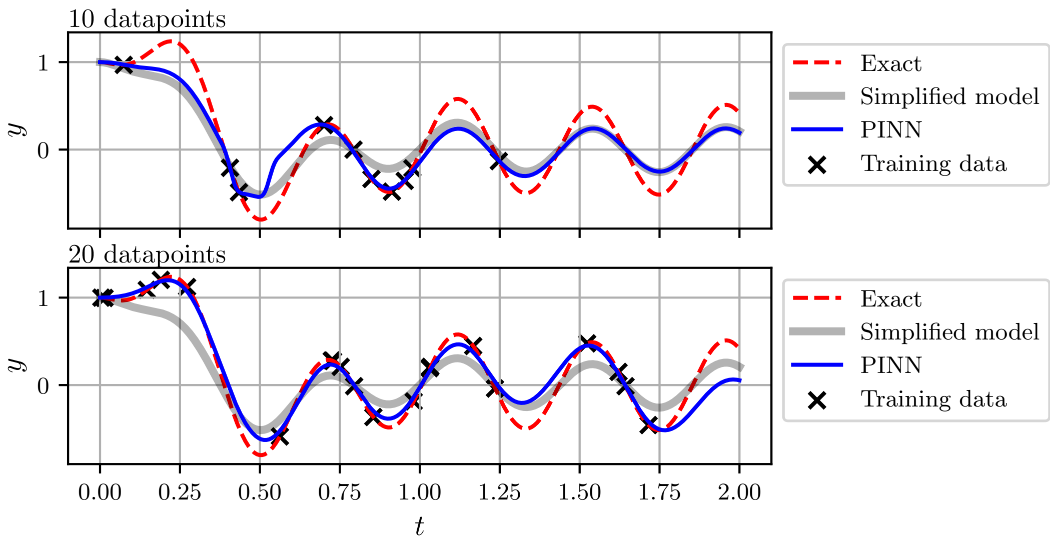


Figure 4.9: Case 1: Simplified model. Randomly sampled. Residual loss weighted 0.1.

Figure 4.9 illustrates that by incorporating weighting into the PINN model, it is possible to satisfy the data points. At the same time, it still keeps the characteristic of the simplified model structure. For 10 data points, there is a strange PINN prediction at t equal to 0.5; this indicates that there might be solutions that might be completely wrong. This needs to be investigated more before use in a real-life application. For 20 data points, the model uses the available data to improve the model. The method is successful in incorporating measurement data to improve the model. However, it requires the correct weighting. Therefore, more development is required in weighting loss terms and sampling residual points effectively.

Trainable parameter

This study set the force amplitude F as an unknown and trainable parameter. This approach requires that the form of the equation is known; this is, however, the case for several engineering problems. The objective is to utilize the PINN approach to identify the correct force amplitude term, thereby enabling the determination of the accurate solution.

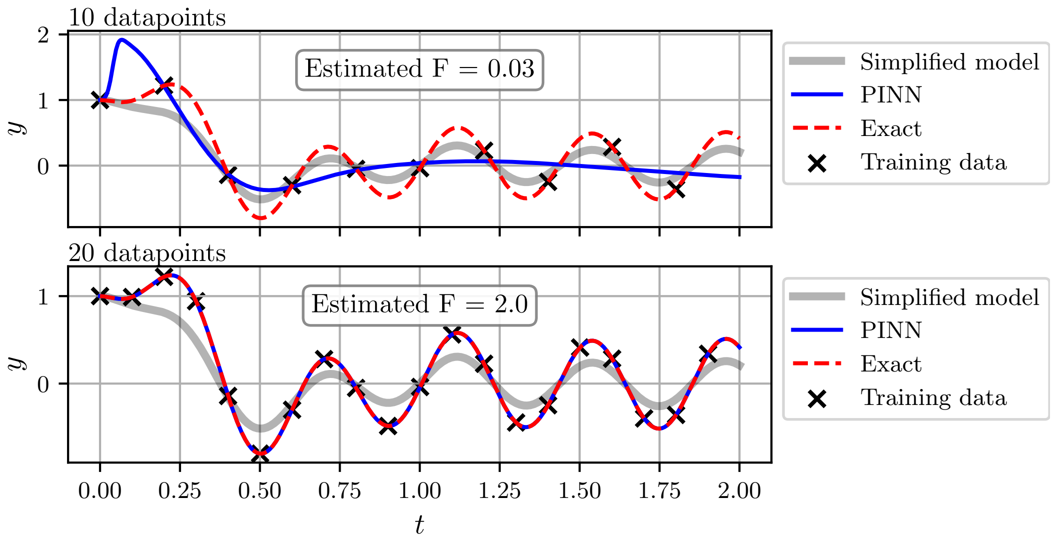


Figure 4.10: Case 1: Simplified model. F unknown parameter. Fixed time sampling. Equal weighting.

Figure 4.10 demonstrates that for 10 data points, the model finds an entirely incorrect estimation of F, resulting in a highly inaccurate prediction. However, with 20 data points, the PINN model accurately estimates the correct F and can predict the exact solution. It is worth noting that the since the parameter is found, the time domain can be extended. This study will be further tested with a lower residual weighting.

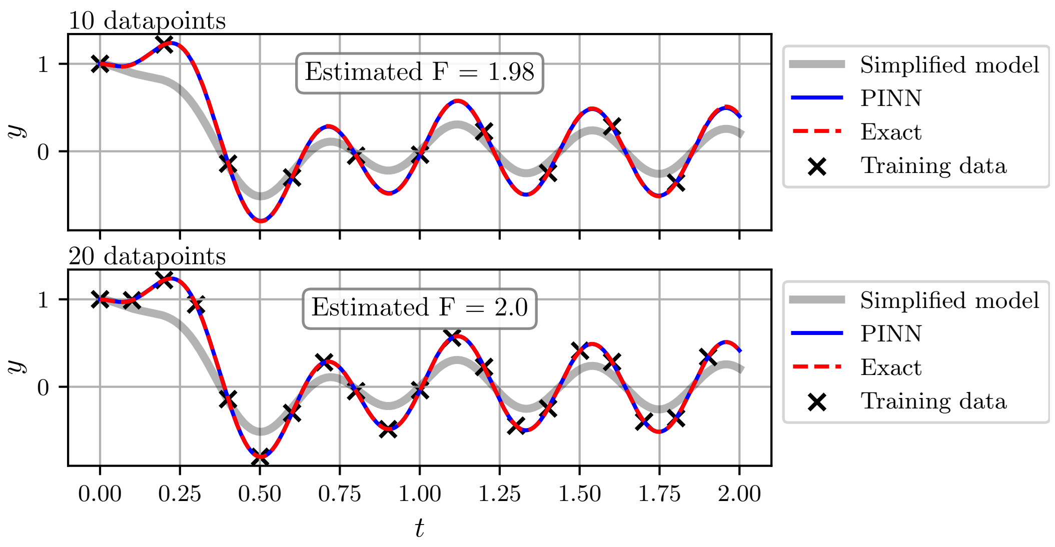


Figure 4.11: Case 1: Simplified model. F unknown parameter. Residual weight = 0.1

Figure 4.11 illustrates that adjusting the residual weight to 0.1 significantly improves the model's performance, enabling accurate estimation of F in both cases and leading to the correct solution. Randomly sampled data points are considered to investigate the model's performance further.

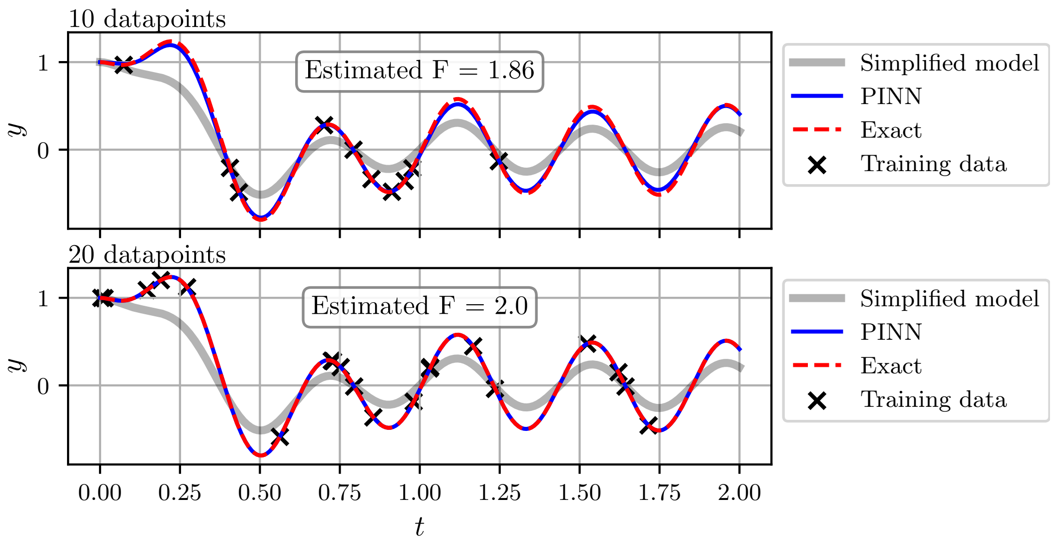


Figure 4.12: Case 1: Simplified model. F unknown parameter. Residual weight = 0.1

Figure 4.12 illustrates that even when using randomly sampled data points, the model can still closely identify the correct value of F. This observation highlights the robustness and generalization ability of the model. The model can utilize the measurements at hand to improve the model. Since the estimated parameter is known, some of the model's interpretability is kept.

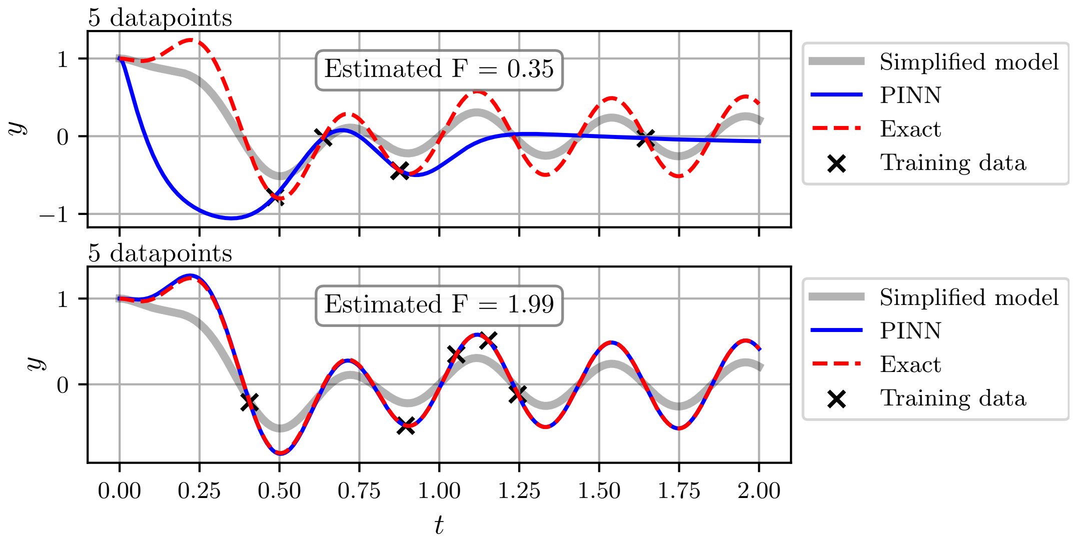


Figure 4.13: Case 1: Simplified model. F unknown parameter. Residual weight = 0.1

Figure 4.13 showed the model's performance when trained with five data points. In this case, the model finds the solution successfully in one out of two possible cases, indicating that the amount and location of training data can influence its performance. However, in engineering situations, there is often an idea of the parameters' value. This can be taken into account in the optimization. For example, the solution is discarded if the value is outside a chosen interval.

4.1.6 Summary

The PINN method demonstrates capability in forward-solving and inverse problem-solving scenarios in exploring the simplified forced-mass-spring-damper system. Traditional numerical solvers are more efficient and accurate when it comes to forward problem-solving. Nevertheless, the PINN methodology remains appealing due to its ease of implementation, potential for various extensions, and flexibility regarding factors such as mesh, initial conditions, and boundary conditions. There is currently a lot of development focused on improving this method. The CRUNCH group led by G. Karniadakis 2023 believes that this approach has the potential to become the next generation of numerical solvers.

The inverse problem-solving capability of PINN proves to be highly effective. This approach is particularly valuable in real-world situations where only sparse measurement data is available, which can be combined with a differential equation with unknown parameters. One notable advantage of the PINN approach in these cases is its robustness against noise and few data points. This feature increases applicability in practical engineering scenarios where high-quality, dense data may not always be accessible. Therefore, the PINN approach can be useful in engineering applications requiring inverse problem-solving. This could, for example, be structural identification and damage detection.

Using PINN with a simplified model demonstrates its potential in combining a simplified model with measurement data. With proper weighting, the model is able to incorporate measurement data with a physical model. It is evident that assigning a lower weight to the residual loss leads to improved results. However, selecting the appropriate weight remains a challenging task. The solution is also a black box since there are no other measurements than the corresponding loss value for the residual and data, the risk of getting strange results as the training with 10 points in Figure 4.9. Thus, the method shows promise, but further research and investigation are needed to optimize the loss weighting and residual points selection. The best performance is when the model estimates unknown differential equation parameters that align with the available data. This requires to know the form of the equation. In most practical engineering problems, the form of the equation is known. In the case of VIV, VIVANA-TD can describe the phenomena if the correct empirical parameters are chosen. Choosing the correct empirical parameters is a challenging task. By using PINN's ability to infer the coefficients that match the data points, the abilities of machine learning and physical models are kept. There is also kept a physical interpretation in the form of an adjusted empirical coefficient.

4.2 Case 2: VIVANA-TD

4.2.1 NN

Representing the solution using a traditional NN. Sampling 20 training points and training for 10.000 iterations with Adam optimizer with a learning rate of 0.001.

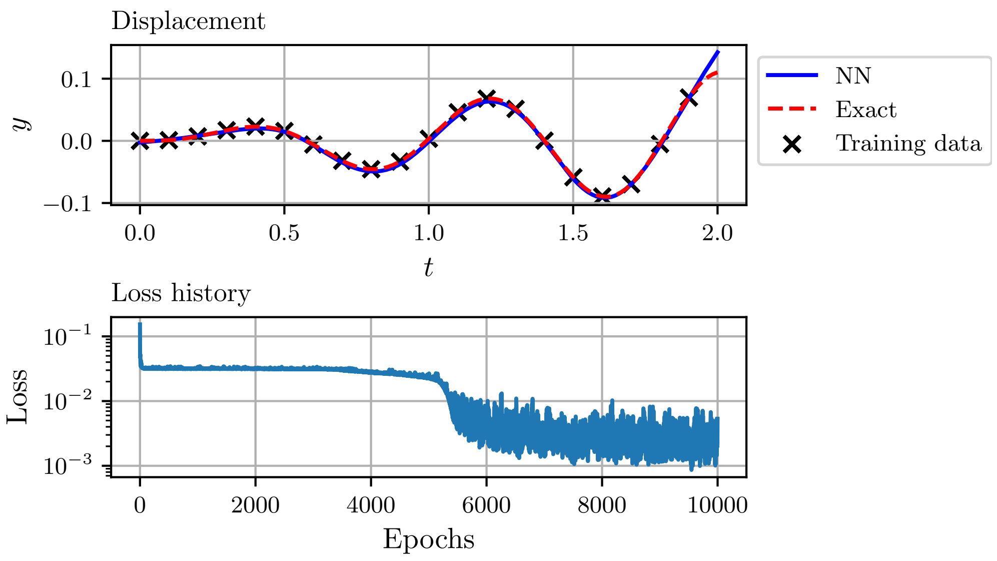


Figure 4.14: Case 2: NN. Top: The displacement prediction from 0 to 2 seconds. 20 training data points were sampled from the exact solution. Bottom: Loss history

Figure 4.14 illustrates that by using a traditional NN, the model is able to learn the solution to the ODE with training data.

4.2.2 PINN Forward problem

The loss function will now become

$$\text{loss} = \omega_{IC} L_{IC} + \omega_{res} L_{res} \tag{4.6}$$

$$L_{IC} = (NN(0) - y(0))^2 + (\frac{dNN(0)}{dt} - \dot{y}(0))^2 \tag{4.7}$$

$$L_{res} = \frac{1}{n}\sum_{i=1}^{n}(\frac{d^2NN(t)}{dt^2}(m+m_a) + NN(t)k - F_{SIMA}(t)) \tag{4.8}$$

In Equation 4.8 n is the number of residual points. m is the weight of the cylinder, m_a is the added mass, k is the spring stiffness, and F_{SIMA} is the hydrodynamic force from SIMA. The F_{SIMA} could be replaced with the VIVANA-TD Equation 2.12. Then F_{SIMA} would be a function of velocity ($\frac{dNN(t)}{dt}$) and acceleration ($\frac{d^2NN(t)}{dt^2}$), this also requires a model to capture the change in phase angle between structural velocity and vortex shedding. However, the phase angle modeling could be avoided by obtaining it directly from SIMA simulation. Before further analysis, the complete hydrodynamic force is used to assess the PINN model's capability to capture the system's behavior accurately.

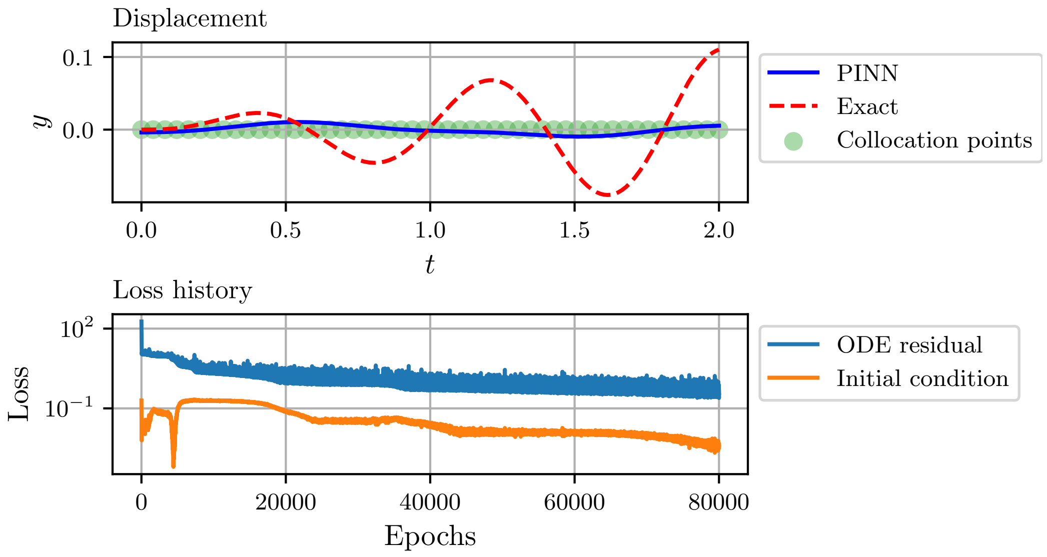


Figure 4.15: Case 2, PINN prediction. Equal weights. Top: the displacement prediction from 0 to 2 seconds. Left: Training loss history

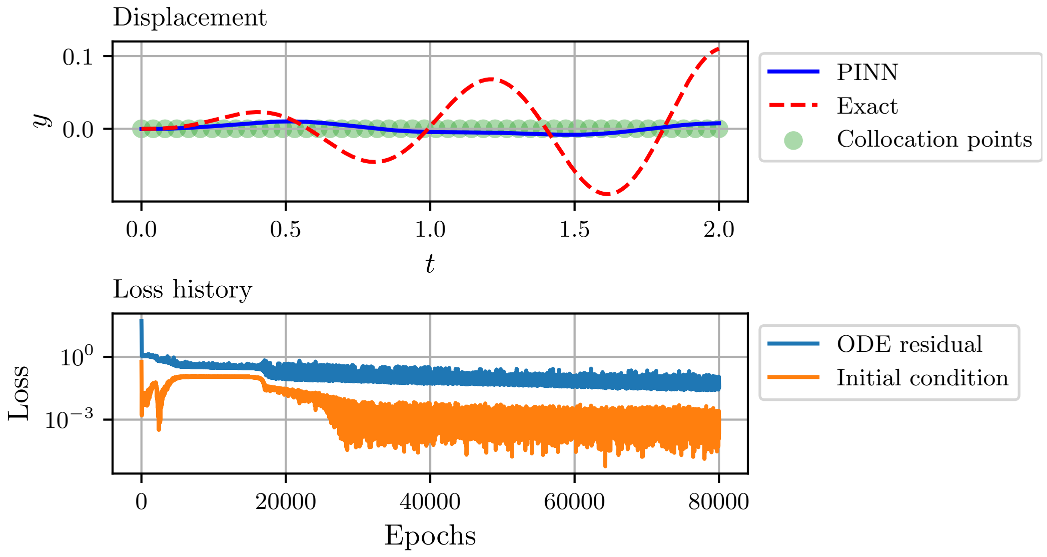


Figure 4.16: Case 2, PINN prediction. Right: the displacement prediction from 0 to 2 seconds. Left: Training loss history. With a residual weight of 0.001

Looking at Figure 4.16, it is clear that the PINN is unable to learn the correct solution. This led to an extended hyperparameter search to get the model to converge to the correct solution. Different loss functions, NN architecture, learning rates, and more were considered. However, the model failed to converge to a correct solution, resulting in highly unsatisfactory results. The suspected issue is with the multi-objective optimization of the losses. This led to a deeper investigation into the failure modes of PINN was conducted (Krishnapriyan et al. 2021; S. Wang, Teng et al. 2020; S. Wang, Sankaran et al. 2022; Basir and Senocak 2022), leading to the belief that the issue likely stemmed from the different scaling of the loss terms. As a result, extensive efforts were made to find the optimal loss weights. However, the model was still unable to converge to a satisfactory solution. This ultimately led to the realization that the problem lay in the baseline formulation of the model and that further extensions to the baseline PINN were necessary to achieve the desired results. In research done by S. Wang, Teng et al. 2020, they found a fundamental failure in physics-informed NNs related to stiffness in the gradient flow dynamics. This leads to an unstable imbalance in the magnitude of the back-propagated gradients during model training using gradient descent, leading to vanishing gradients. To illustrate the problem, the gradients values after training for case 2 and case 1 will be illustrated in a histogram. The gradients will be plotted with respect to the residual loss and initial condition loss.

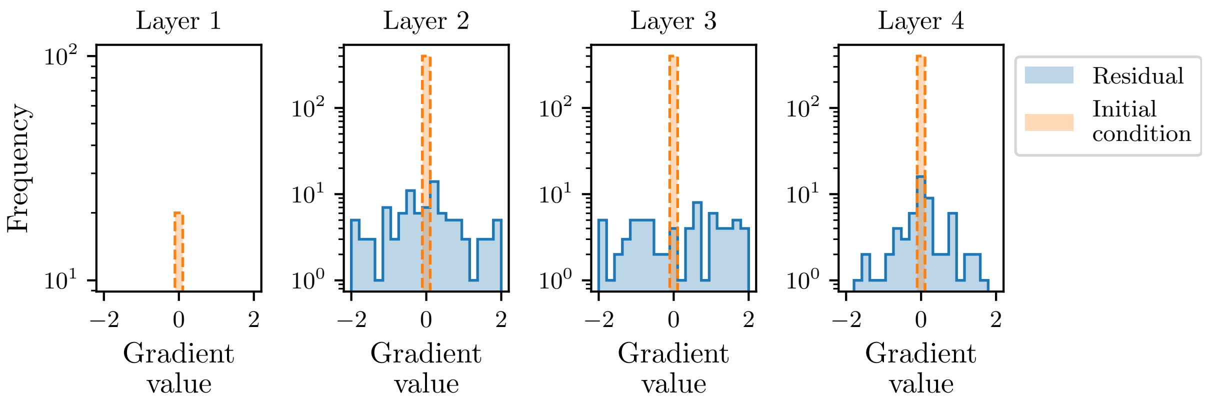


Figure 4.17: Case 2: Gradient histograms, after training

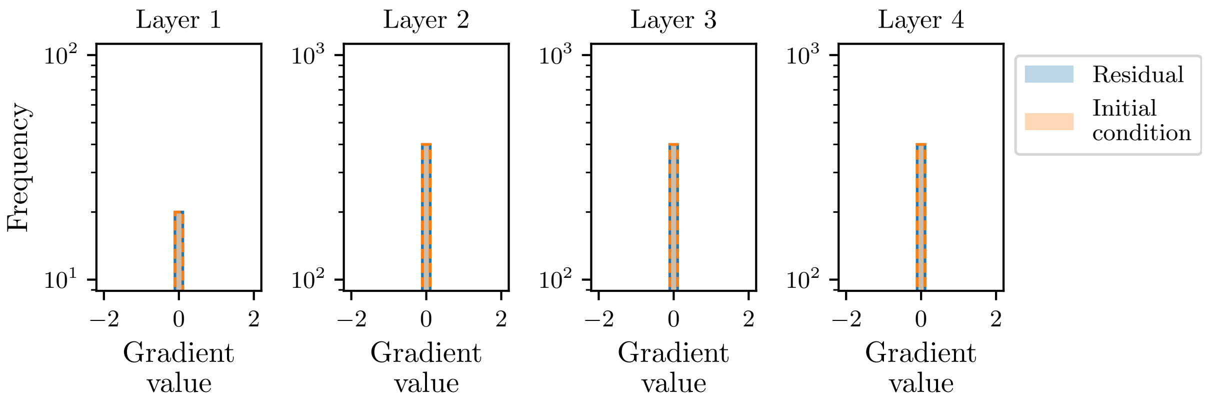


Figure 4.18: Case 1: Gradient histograms after training

Figure 4.17 show the gradient histogram after training case 2. Figure 4.18 shows the gradient histogram after training case 1. Investigating the gradient values in case 2, the gradient value concerning the initial conditions is close to zero, while the gradient values with respect to the residual are larger. The different scale of gradient values leads to an unbalanced optimization. Comparing the gradients from case 2 with case 1, it is clear that in case 1, the gradient values with respect to the different loss terms are more similar. This leads to a more balanced optimization problem, meaning it is more unlikely to end up in local minima. This was also experienced by S. Wang, Teng et al. 2020 when increasing the constant in the 1D Helmholtz equation. Further investigation into PINNs and the problem in case 2 is necessary to overcome this issue.

Chapter 5

PINN extensions

It has become evident that relying solely on the baseline PINN is insufficient. Therefore, it is necessary to conduct further exploration to identify the underlying cause of the problem and implement new extensions that can address these issues. The original authors of PINN Raissi et al. 2019 left several questions unanswered regarding the selection of the neural network model, loss function, and optimization. Since the initial publication of the framework, additional research and development have been published to improve the baseline formulation. There have also been published papers that have researched failure modes of PINN and proposed new and improved extensions to its original formulation. This section will first introduce some of the recent extensions to PINN, followed by a focus on understanding and resolving the problems encountered in case 2.

5.1 Literature study

Several recent studies have identified a potential issue with the additional term in the loss function of physics-informed neural networks. (Basir and Senocak 2022) and (Krishnapriyan et al. 2021) have investigated this phenomenon and found that multiple loss terms can create a complicated loss landscape and lead to potential problems with vanishing gradients. They observed that this occurs when there is a significant difference in scale between the initial conditions and the data points, and a large coefficient in the equation increases the problems. Research done by S. Wang, Teng et al. 2020 has suggested that this problem arises from performing gradient descent on the multi-objective loss function, as the greedy procedure may prioritize one loss term over the others and prevent convergence to the correct solution. To address this problem, researchers have proposed improved neural network architectures more resilient to these issues, such as the architecture proposed by S. Wang, Teng et al. 2020 in "Understanding and mitigating gradient pathologies in physics-informed neural networks" and various loss weighting schemes. Potential improvements could be made by utilizing neural networks suited to represent sine waves, such as ModalPINN (Raynaud et al. 2022) or Fourier neural networks by (Ngom and Marin 2021). Wong et al. 2022 addressed the issue of getting stuck in local minima by introducing a neural network model in their paper titled "Learning in Sinusoidal spaces with physics-informed neural networks". The result was a neural network that increased gradient variability, leading to the model escaping local minima. Several more customized neural network architectures have been proposed to increase the training of PINN. There is also the possibility to enforce the initial condition always to be satisfied, thus removing the multi-loss landscape in the forward problem (L. Lu, Pestourie et al. 2021). Other approaches suggest training in sequence, starting with a simple problem and gradually increasing the challenge (Krishnapriyan et al. 2021). Ji et al. 2021 suggested that stiff ODE was causing problems with the performance of PINN. They suggested converting the equation to a nonstiff form to get better convergence. However, the most discussed extension is changing the weighting (McClenny and Braga-Neto 2022). With many extensions and improvements presented, it becomes challenging to determine the most effective extensions in this particular case. Therefore, more experiments will be conducted to find the central issue in the VIVANA-TD problem.

5.2 Investigating the failure modes

5.2.1 Case 3: Constructed VIVANA-TD

Due to issues getting the VIVANA-TD problem to converge, a modified equation will be used and simulated with a numerical solver (ODEINT) to eliminate potential sources of error. The aim is to represent the VIVANA-TD problem as closely as possible, which will facilitate the testing of different hyperparameters and extensions. The method that yields the best results will then be applied to the data simulation data. Considering the challenges, different modifications and extensions of PINN will be tested. The ODE to be considered is as follows:

$$(m + \rho A)\frac{d^2u}{dt^2} + ku = F\cos(\omega t) \tag{5.1}$$

$$u_0 = 0, \dot{u_0} = 0 \tag{5.2}$$

with the parameters,

m	13.05
k	1197.2
F	18
ω	8
ρ	1000
A	0.007854

Table 5.1: Case 3: Parameters

In Figure 5.1, the solution is plotted alongside the solution using the force data from VIVANA-TD. The constructed case is similar to the VIVANA-TD case, suggesting that the hyperparameters and modifications that work well for the constructed case should also work well for the VIVANA-TD case.

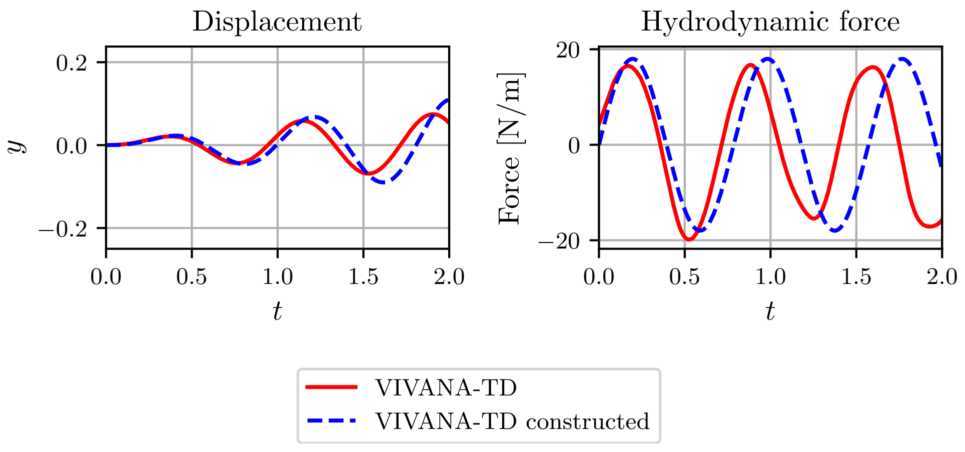


Figure 5.1: Comparing the VIVANA-TD case with the constructed case

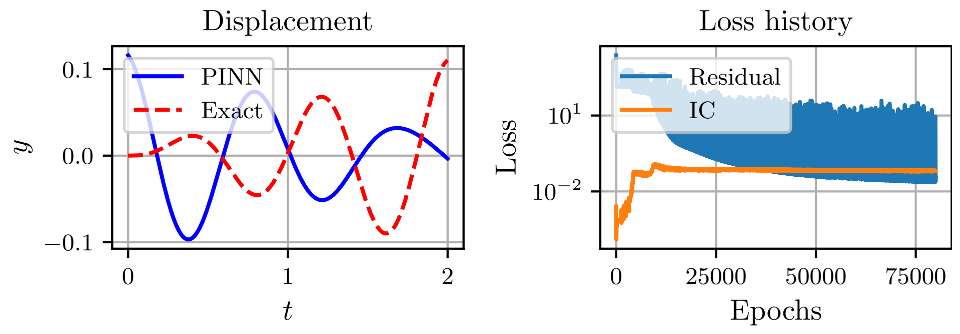


Figure 5.2: Case 3: Results from using the baseline PINN on the constructed equation.

Figure 5.2, shows the results from using the baseline PINN formulation on case 3. It becomes clear that the same problem experienced in case 2 is present. Furthermore, an exhaustive search for optimal weights was conducted. However, the PINN was not able to converge to the correct solution.

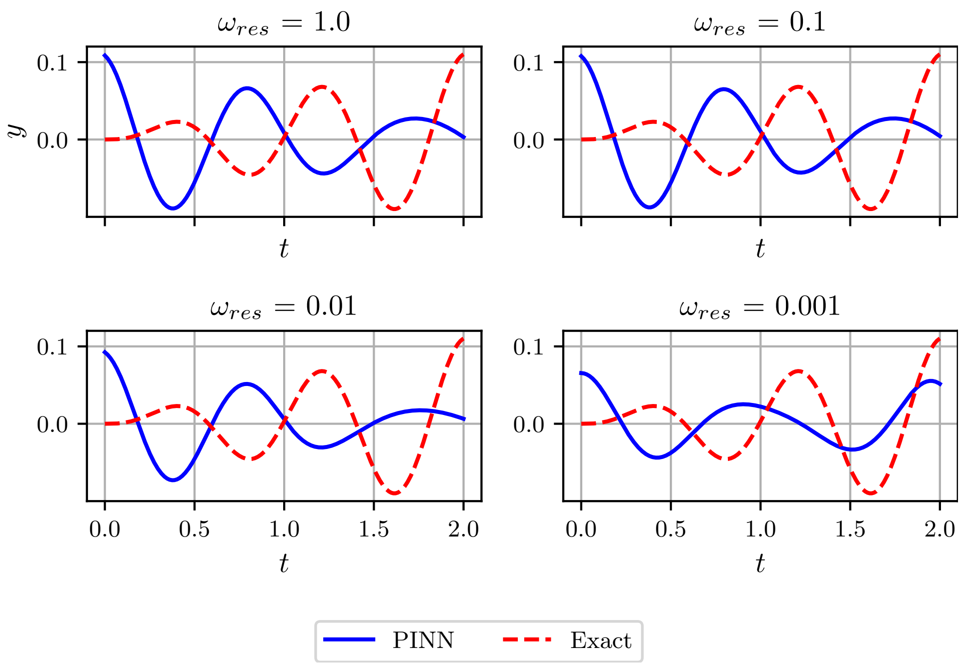


Figure 5.3: Case 3: Different residual loss weighting.

Figure 5.3 illustrates some of the different residual weights tested. As evident in the Figure 5.3, there is little change or improvement when changing the residual weight.

5.2.2 Equation

Solving numerically is not a problem for larger coefficients; however, as pointed out by Krishnapriyan et al. 2021, the PINN approach struggles to find a solution when the coefficients become large. Substituting the coefficients from Table 5.1 into Equation 5.1 results in Equation 5.3. This equation has larger coefficients than case 1, and there is also a substantial scale difference between $\ddot{u}(t)$ and $u(t)$.

$$20.904\ddot{u}(t) + 1197.2u(t) = 18\cos 8t \tag{5.3}$$

The term multiplied by $u(t)$ is almost 60 times larger than the term multiplied by $\ddot{u}(t)$. This discrepancy might pose challenges in achieving convergence during optimization. Krishnapriyan et al. 2021 demonstrated that significant coefficients complicate the optimization landscape, resulting in sub-optimal solutions. Given the substantial difference in initial and residual loss magnitudes, issues can arise during backpropagation through the network S. Wang, Teng et al. 2020. Different values of k will be investigated to see if the problems originate from the significant coefficients.

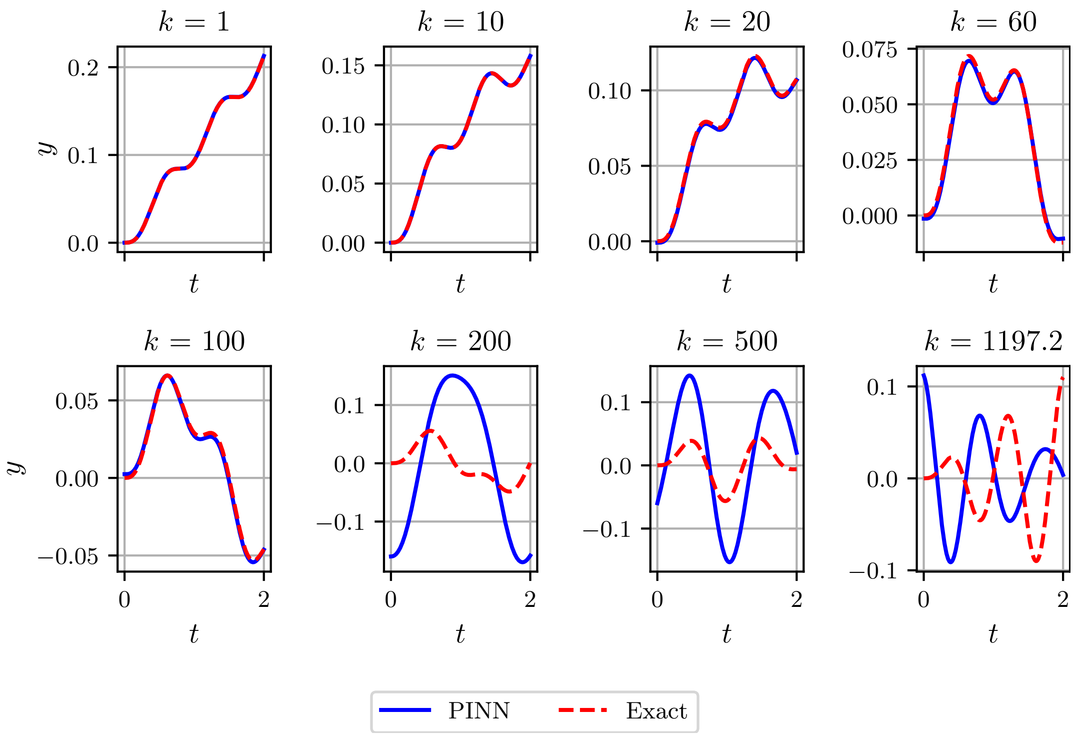


Figure 5.4: Case 3: Different values of k. Trained on 10000 iterations with Adam with learning rate=0.001.

From Figure 5.4, it is evident that complications arise as the k becomes larger, a finding that aligns with what Krishnapriyan et al. 2021 pointed out. However, a positive takeaway is that the method does not seem to struggle with terms being at different scales. To investigate if there is an issue with different eigenfrequencies of the system and the forcing term, different angular frequencies of the force will be explored. Setting the learning rate to 0.001 and the residual loss to 0.1, yields the following results,

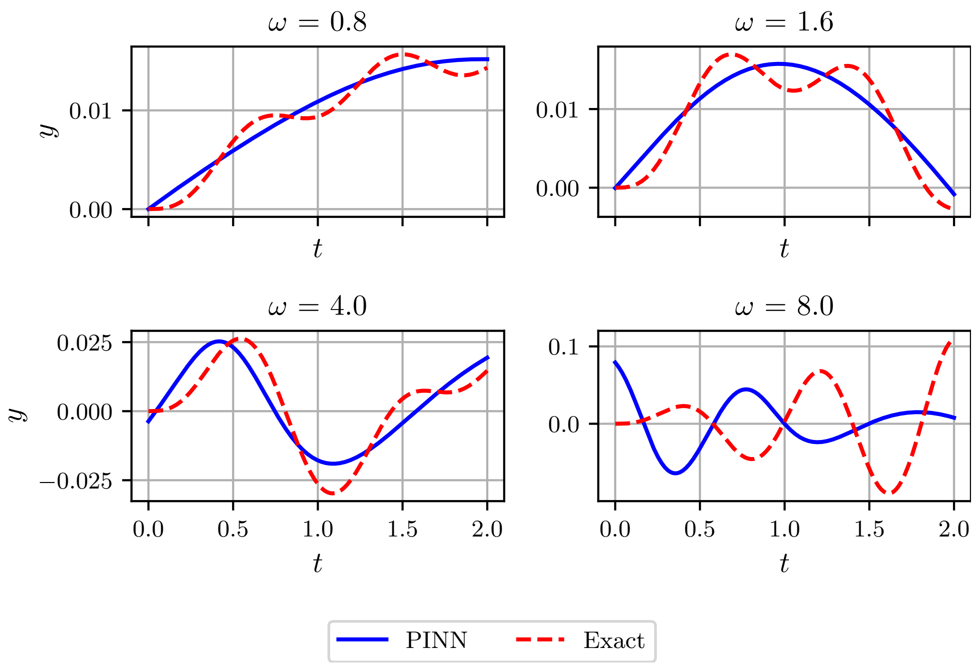


Figure 5.5: Case 3: Different angular frequency force values. Trained on 10000 iterations with Adam with learning rate=0.001.

Investigating Figure 5.5 is clear that by setting a lower frequency, the PINN is still not able to find the correct solution. However, it is clear that for omega ω equal to 0.8,1.6, and 4.0, the PINN can capture the low frequency. This can indicate that PINN struggles with capturing high frequencies or solutions with two different frequencies. Two different frequencies characterize a stiff problem. Traditional explicit numerical methods, such as forward Euler, require very small timesteps for stiff ODEs.

5.3 Improving the model

Different techniques will be investigated to improve the PINN. To limit the search space, the same 50 collocation points will be used for all the tests, and the neural network architecture is described in Table 3.3. The focus will be on improving the training and optimization. However, it should be noted that changing the architecture also influences the training of the neural network.

5.3.1 Curriculum Learning

Krishnapriyan et al. Krishnapriyan et al. 2021 proposed using curriculum training to enhance the training of PINNs. The idea behind curriculum training is to start with an easier problem and progressively tackle more challenging problems. After each training session, the weights derived from the more straightforward problem are applied to the more complex problem. All the following training weights are set to one, and the learning rate is set to 0.001 with 10,000 iterations using the Adam optimizer. Although it has been determined that assigning a lower weight to the residual loss typically yields better solutions, this approach has been selected to isolate the effects of curriculum learning.

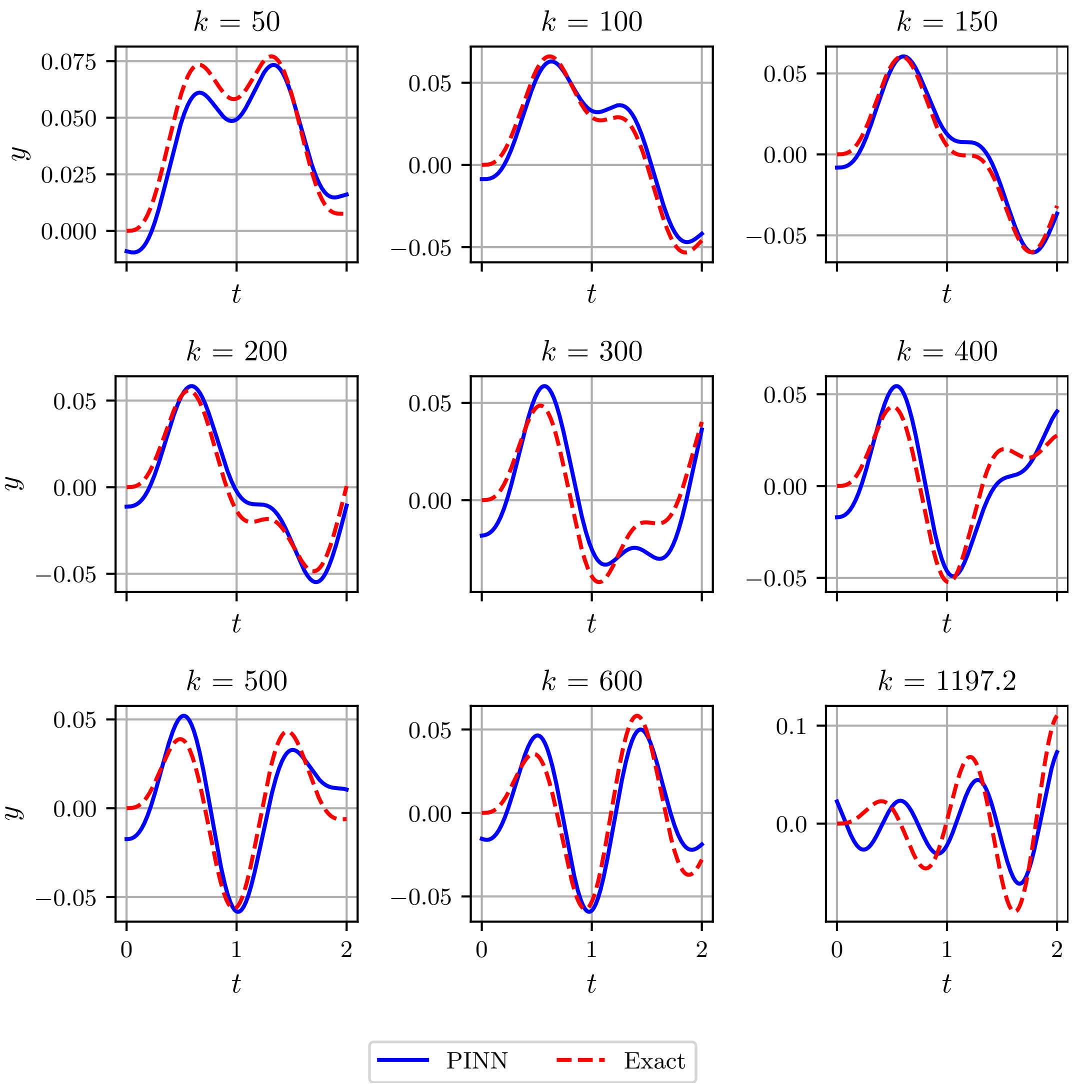


Figure 5.6: Case 3: Curriculum training. 20000 iterations with Adam optimizer learning rate of 0.001.

Comparing the results from Figure 5.6 and Figure 5.4, it is evident that the training is improved with the use of curriculum training.

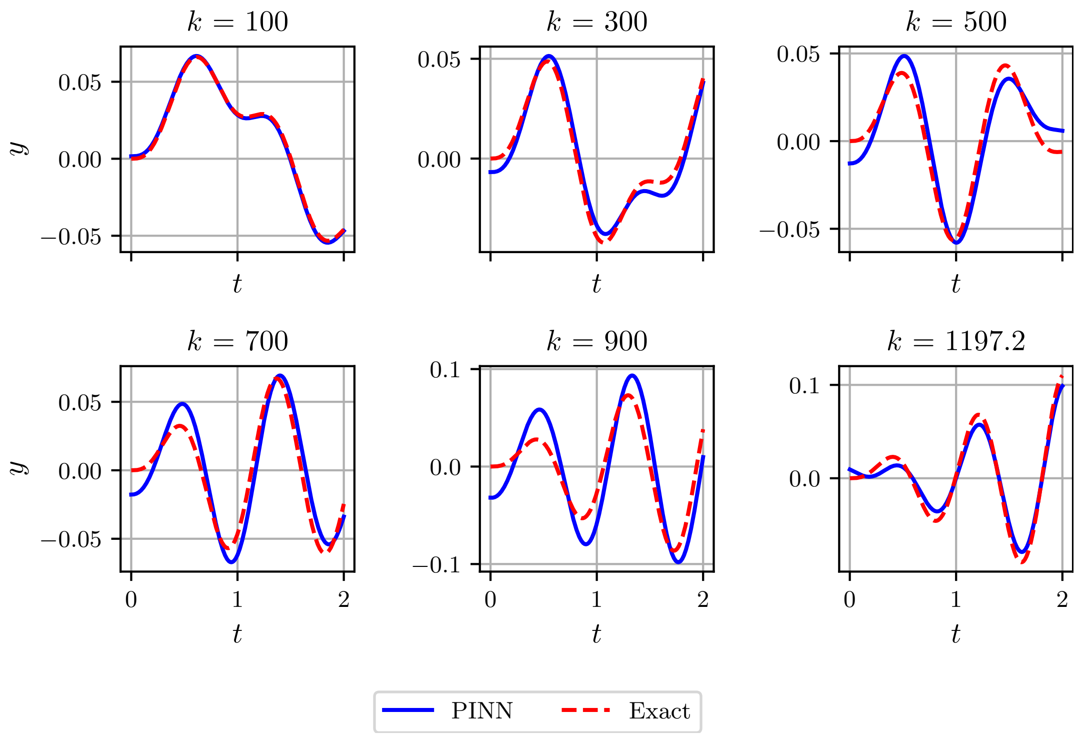


Figure 5.7: Case 3: Curriculum training. Weighted residual loss ω_{res} = 0.1. 20000 iterations with Adam optimizer learning rate of 0.001.

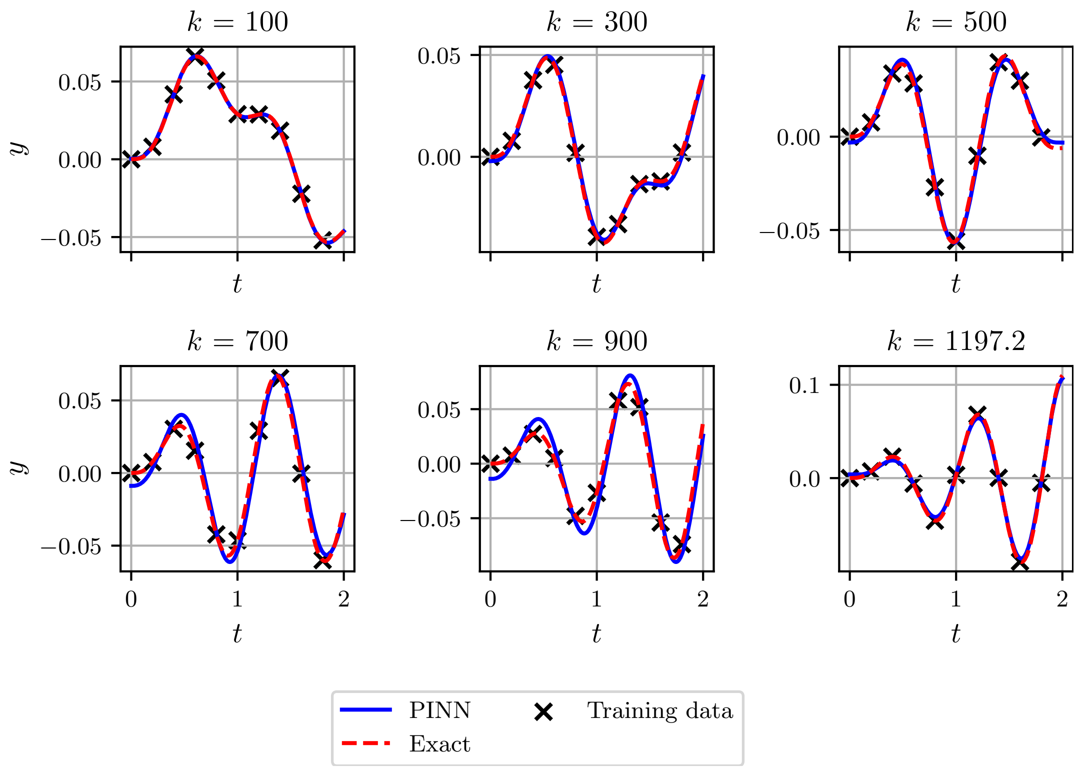


Figure 5.8: Case 3: Curriculum training with training data. Weighted residual loss ω_{res} = 0.1. 20000 iterations ω_{res} = 0.1

Figure 5.7 shows that by setting a lower residual loss weight, the curriculum approach improves more, even when larger steps are used between the k values. Including training data adds another term to the loss function. Figure 5.8 shows that the prediction becomes more accurate by incorporating training data. However, at k=700 and k=900, the initial condition is not satisfied. The results indicate that there are still some problems with having multiple different loss terms. The method also increases the training time substantially. Thus other methods will be investigated.

5.3.2 Sequence-to-sequence

Krishnapriyan et al. 2021 proposed applying a sequence-to-sequence approach to improving the training of PINNs. This method discretizes time into different timesteps, each containing its own set of collocation points. The model's training is then conducted incrementally: Initially, the model is trained solely on the first timestep. Subsequently, it is trained on both the first and second timesteps together, and this sequential process continues. This sequence-to-sequence framework has several potential advantages. First, breaking down the complex problem into more manageable tasks allows the model to incrementally learn and adapt its understanding of the problem at each timestep. It is also possible to add more layers if needed, limiting the need for a very large neural network initially. The model was trained using ten steps, with five collocation points for each step.

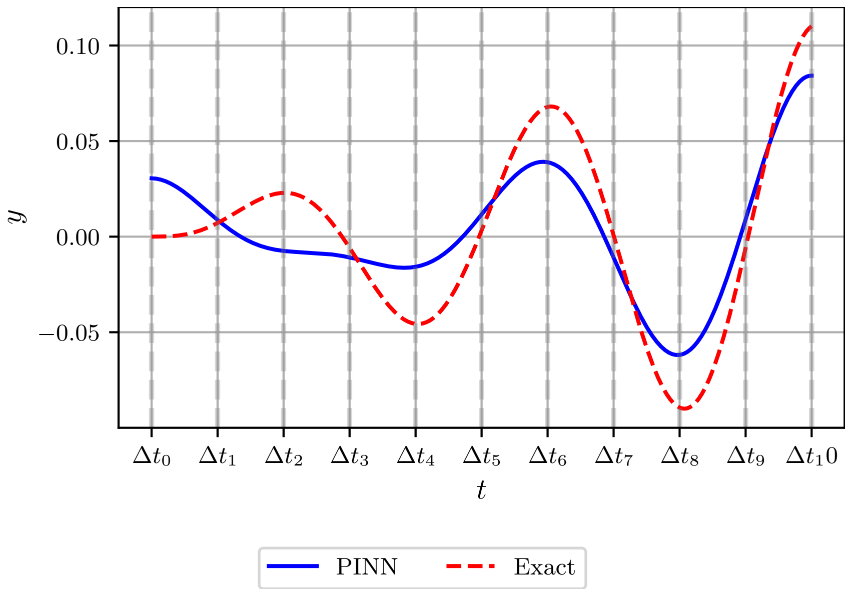


Figure 5.9: Case 3: sequence to sequence training. 10 steps with 5 collocation points for each step. Residual weight 0.01. Learning rate 0.001. Optimized using Adam for 20000 iterations

In Figure 5.9, the $t \in [0, 2]$ is divided into 10 steps, with 5 collocation points in each step δt. Using sequence-to-sequence training, the PINN methodology is still not capable of learning the system's underlying dynamics completely, as shown in Figure 5.9. This approach comes with a significant time cost, as the process involves training the neural network ten times. Another similar approach to consider is to train a new NN for each timestep and combine all these individual models into one comprehensive model at the end. This method could help alleviate issues associated with the increasing time domain, which will need more expressibility of the neural network. However, sequence-to-sequence training will not be investigated further due to the longer training time.

5.3.3 Multi-objective optimization

In this section, methods that make multi-objective optimization work more seamlessly will be investigated.

Hard constraints

The best fix is sometimes just to remove the problem. Therefore enforcing the neural network to satisfy the initial condition by construction can lead to easier optimization. Since the problem is a time-dependent problem, ensuring that the neural network output matches the desired initial condition can be achieved by multiplying the output with the input. This is also what is done in the paper using NN to solve ODE (Lee and Kang 1990). The initial condition was enforced through the structure of the neural network. This approach offers the advantage of simplifying the loss term by eliminating one constraint. The suggested formulation for the neural network then becomes,

$$u(x, t; \theta) = t \cdot NN(x, t; \theta) + x_0 \tag{5.4}$$

In this equation, u denotes the new model with the enforced initial condition, t is the input, and x_0 is the position at $t = 0$. This formulation ensures that the initial condition is always satisfied, effectively converting it into a hard constraint. However, since case 3 is a second-order differential equation, there is also an velocity initial condition that needs to be satisfied to find the unique solution. Since there is no easy way of enforcing the velocity initial condition, it is included in the loss function as a soft constraint.

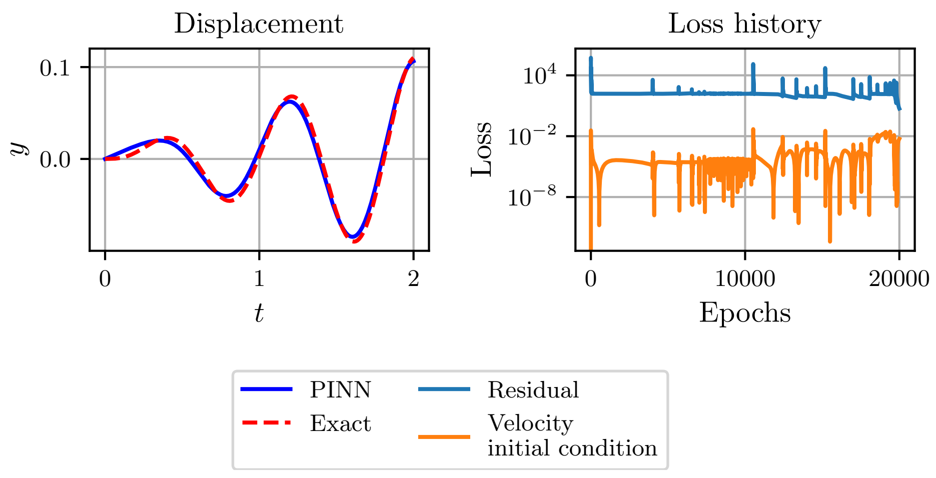


Figure 5.10: Case 3. Hard constraints. Weight residual 1.0, learning rate 0.01, 20000 iterations Adam

Figure 5.10 demonstrates that by enforcing the initial condition, the model successfully captures the correct solution. The results were highly encouraging, prompting further attempts to enhance the model's performance. For this purpose, the model was trained using 40,000 iterations using the Adam optimizer and subsequently switching to the L-BFGS optimizer. The learning rate was set to 0.001.

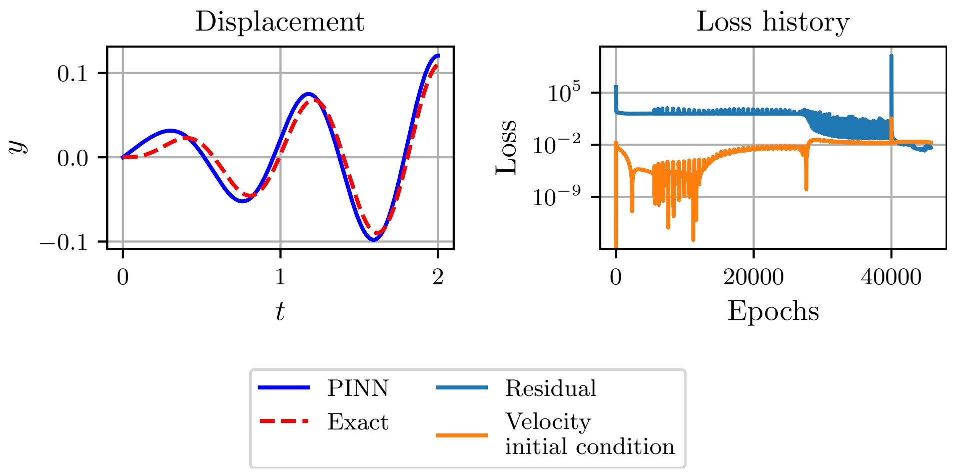


Figure 5.11: Case 3, Hard constraints. Weight residual=1. 40000 Adam iteration then L-BFGS.-

Figure 5.11 demonstrates that increasing the training duration leads to worse results. Previous experience has indicated that setting the residual weight, denoted as ω_{res}, to 0.1 has yielded better outcomes. To validate this observation, the same training setup as in Figure 5.11 with weighted residual utilized.

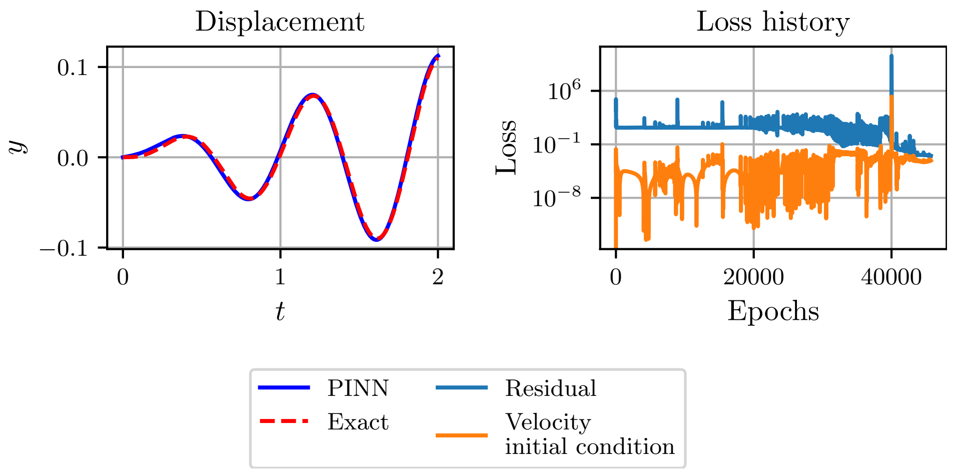


Figure 5.12: Case 3, Hard constraints. Weight residual=1. 40000 Adam iteration, then L-BFGS optimizer

Figure 5.12 presents the results, clearly illustrating that setting the residual weight to 0.1 leads to significantly improved outcomes.

By incorporating the initial displacement directly into the neural network, the optimization problem becomes more manageable, resulting in better results. Additionally, it has been observed that the choice of the residual weight plays a crucial role in achieving convergence of the model. Thus motivating the investigation of automatic loss weighting.

Learning rate annealing

An innovative learning rate annealing algorithm was proposed in the study addressing gradient pathologies (S. Wang, Teng et al. 2020). This algorithm is designed to adaptively scale the loss function using gradient statistics, thereby eliminating the need for manual tuning to find the optimal weights.

The algorithm considers the loss function $L(\theta)$, which is a function of the neural network parameter θ. The function $L(\theta)$ is defined as follows:

$$L(\theta) := L_{res}(\theta) + \sum_{i=1}^{M} \lambda_i L_{data}(\theta) \tag{5.5}$$

In this equation, $L_{res}(\theta)$ denotes the residual loss, while $L_{data}(\theta)$ represents data loss, which includes measurements, initial conditions, and boundary conditions. The data loss is weighted by the free parameter λ_{data}.

The algorithm then computes $\hat{\lambda}_i$ as follows:

$$\hat{\lambda}_i = \frac{\max_\theta |\nabla_\theta L_{res}(\theta_n)|}{\text{mean}_\theta |\nabla_\theta L_{data}(\theta_n)|}, \quad i = 1, ..., M, \tag{5.6}$$

Next, the weights λ_i are updated using a moving average of the form:

$$\lambda_i = (1-\alpha)\lambda_i + \alpha\hat{\lambda}_i, \quad i = 1, ..., M, \tag{5.7}$$

Finally, the parameters θ are updated via gradient descent:

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta L_{res}(\theta_n) - \eta \sum_{i=1}^{M} \lambda_i \nabla_\theta L_{data}(\theta_n) \tag{5.8}$$

The authors of the study recommend hyper-parameter values of $\eta = 10^{-3}$ and $\alpha = 0.9$. The parameter can be updated either on each iteration or at a specified interval. This novel approach allows for automatic tuning of the loss weights, removing the need for manually adjusting the weights. By adaptively scaling the loss function using gradient statistics, this algorithm offers a robust solution to one of the challenges posed by the PINN method (S. Wang, Teng et al. 2020). The algorithm will be implemented for two tests, one with an adaptive parameter interval equal to 10 and one at each iteration. To test the implementation, the method will be verified on the Case 1 problem.

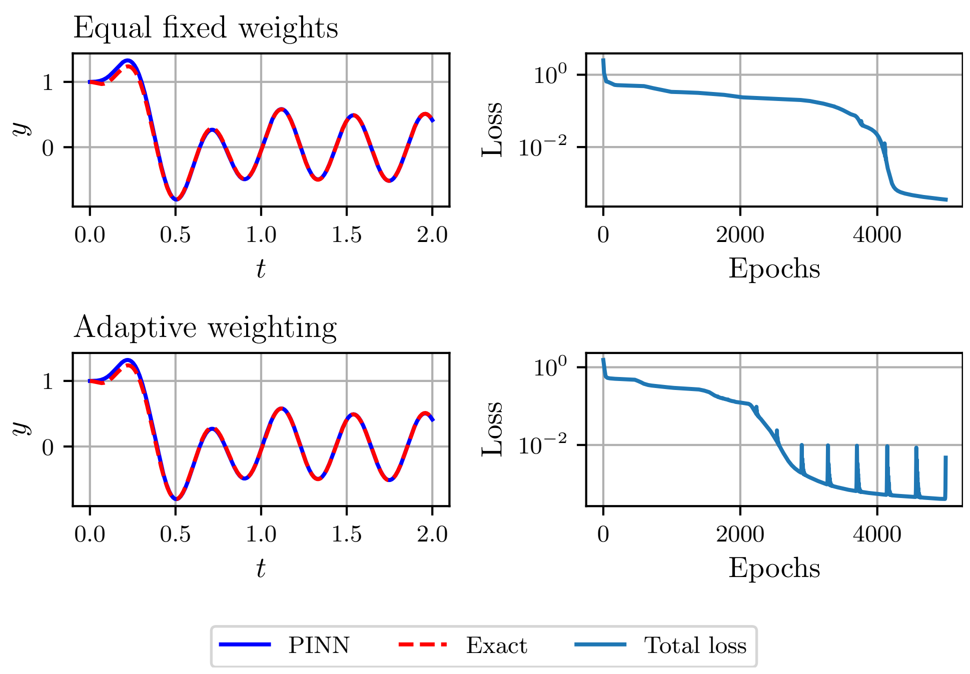


Figure 5.13: Case1: Top: All weights equal to 1. Bottom: Adaptive weights according to S. Wang, Teng et al. 2020

Figure 5.13 shows that the method works better than equally fixed weights as the model total loss is lower than 10^{-2} after 2500 epochs, while for the equal fixed weights, the loss is lower than 10^{-2} after 4000 epochs. However, the method also comes with a small increase in computational work as the adaptive weight needs to be calculated.

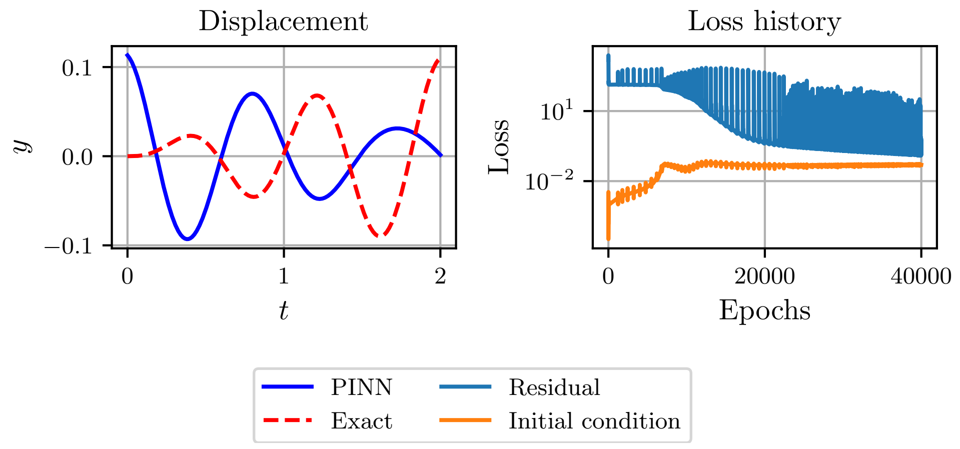


Figure 5.14: Initial condition, lr=0.001, adaptive each iteration

Figure 5.14 illustrates the adaptive weight is not able to find the correct solution. Since the formulation included data points in the data loss term, 5 equally sampled data points will be included in the next simulation. The data points are sampled from the exact numerical solution.

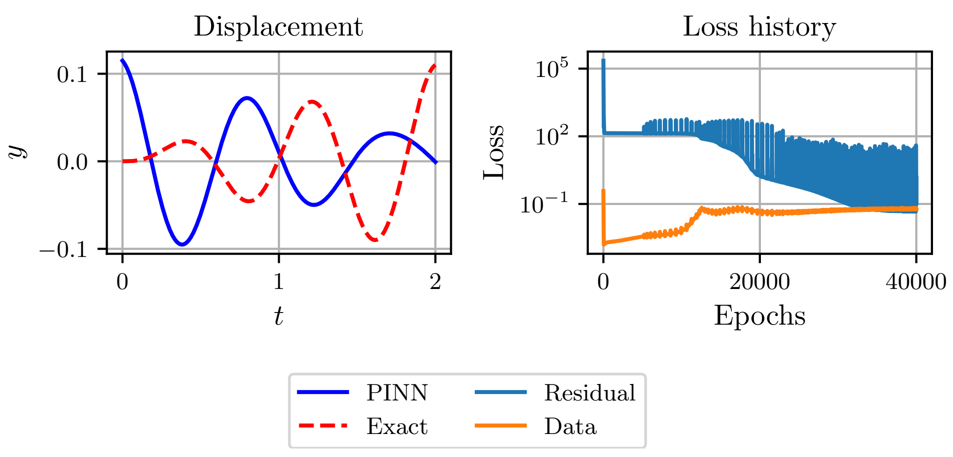


Figure 5.15: lr=0.001, updated weight each iteration

Figure 5.15 illustrates that by including data points, the model is still not able to learn the function. Changing the update frequency of the self-adapting weighting to 10epochs. The following results.

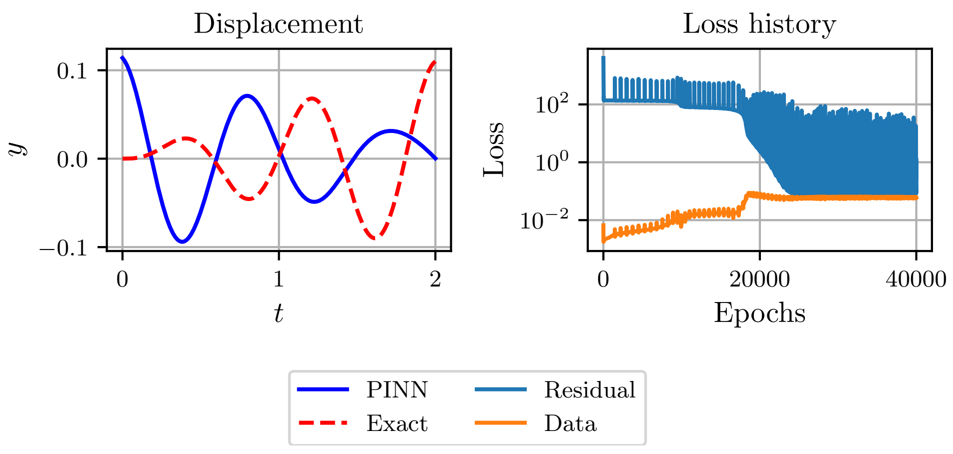


Figure 5.16: lr=0.001. Adaptive parameter updated every 10 iterations, her

Figure 5.16 illustrates that the method is still not able to solve the problems. This leads to the belief that the method is still insufficient, and another multi-loss method will be considered.

Self-adaptive weights

The original authors of the framework (Raissi et al. 2019) recommend initially running the training loop and then calibrating the weights for the various loss components to ensure they are of the same order. This results in weights being presented as a scalar, which remains constant throughout the training process. In contrast, Levi McClenny and Ulisses Braga-Neto propose an adaptive scheme for determining the appropriate weights McClenny and Braga-Neto 2022. This strategy calculates weights for each training point, thereby giving greater weight to points located in challenging areas. The self-adapting weights are updated in the loss function concurrently with the network weights through gradient descent. Implementing the method according to the paper results in the following results.

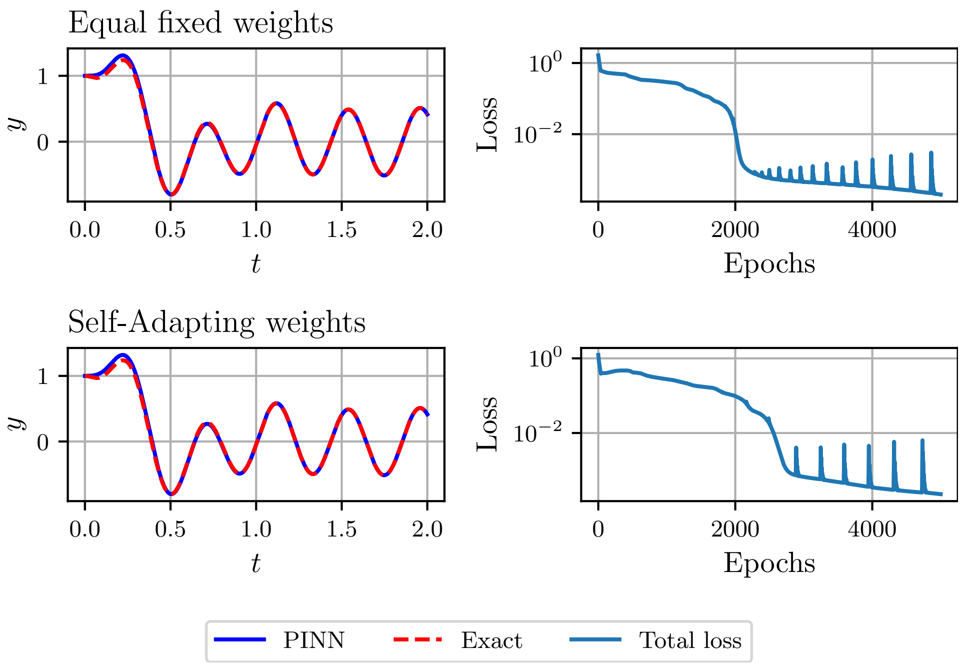


Figure 5.17: Case 1: self-adapting weights Case 1

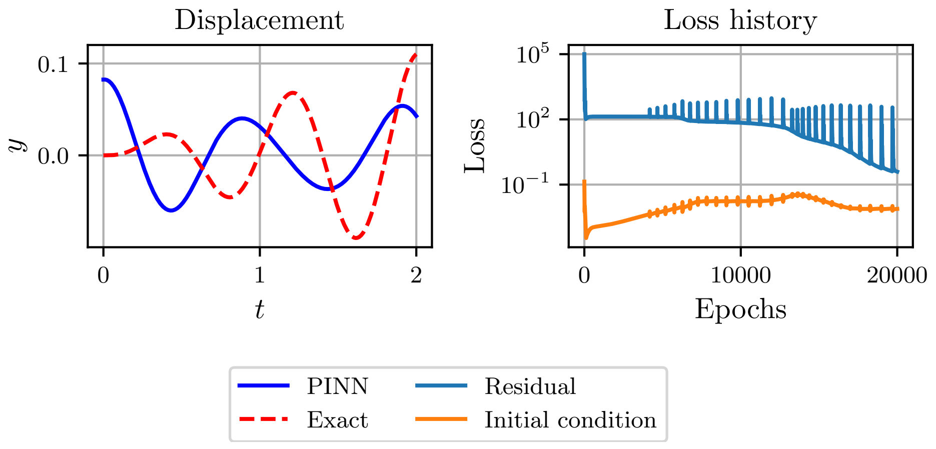


Figure 5.18: Case 2: learning rate=0.001

No performance was gained. However, it should be noted that more variations of the algorithm could be tested, for example, different masking functions. The algorithm was also implemented very rushed based on the published code.

5.3.4 Summary

A few of the published extensions to PINNs have been investigated. Curriculum training offers several benefits. It enables the model to converge to the correct solution by gradually increasing the difficulty of training examples. This approach allows the model to learn the easier problem before tackling more challenging tasks, effectively avoiding suboptimal solutions or local minima. However, it does come at the cost of increased training time. The concept of sequence-to-sequence training employs a similar thought. This method divides the training process into ten timesteps, gradually spanning the entire solution.

This technique fails to achieve convergence and requires significant time to train. As a result, further investigation into the sequence-to-sequence training was not pursued due to the prolonged training duration.

Furthermore, research has been conducted on enforcing the initial always to be satisfied. By using hard constraints, the model was able to converge to the correct solution. The weighting of residual loss remains important for the results. Two techniques that automatically weigh the loss terms were explored. However, using learning rate annealing or self-adapting loss did not yield noticeable improvements in the results.

Due to the hard constraints significantly affecting the training, this motivates further research into different NN architectures more optimized for PINN. The problem remains intricate, with many components that must work together for good results. Much more work is needed to find a suitable model that is easy to train.

5.4 Improved model: Case 2

Based on the takeaways from the investigations, the initial condition will be enforced by the neural network's architecture. The neural network will also be enlarged with two additional layers with 20 neurons in each layer. Adding two extra layers comes from experimenting with different architectures.

5.4.1 Forward problem

In solving the forward problem, the learning rate is set to 0.001, and the weighted of the residual is set to 0.0005.

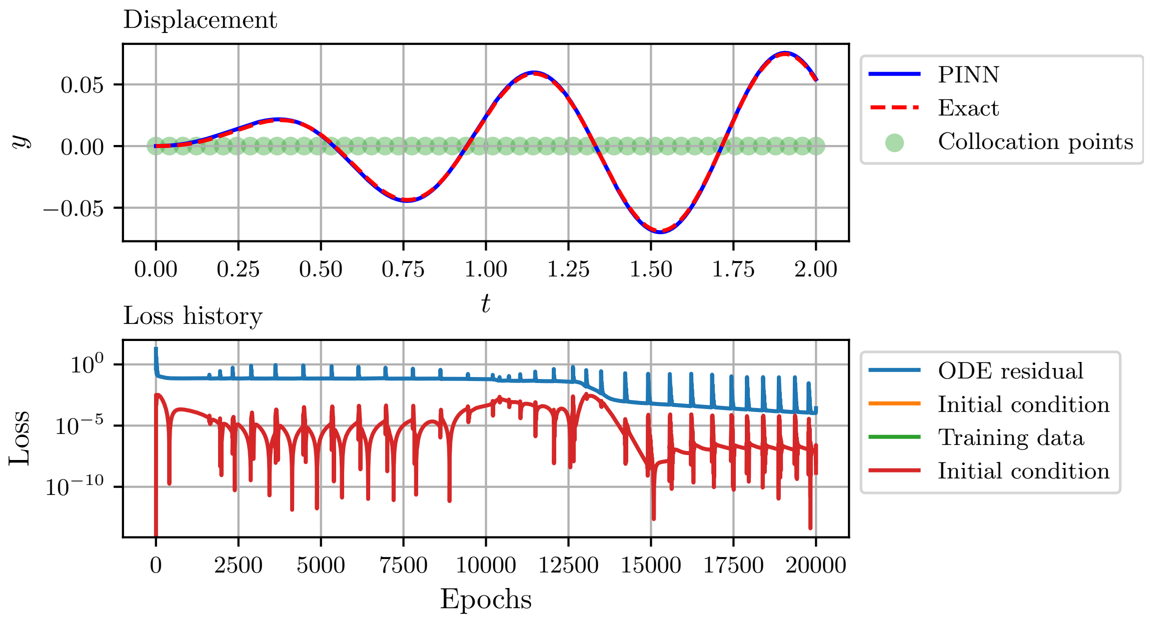


Figure 5.19: Case2, lr=0.001, weight residual = 0.0005

Figure 5.19 shows that PINN is able to find the forward solution matching the solution from the numerical solver. Investigating what will happen if training data is included in the loss function.

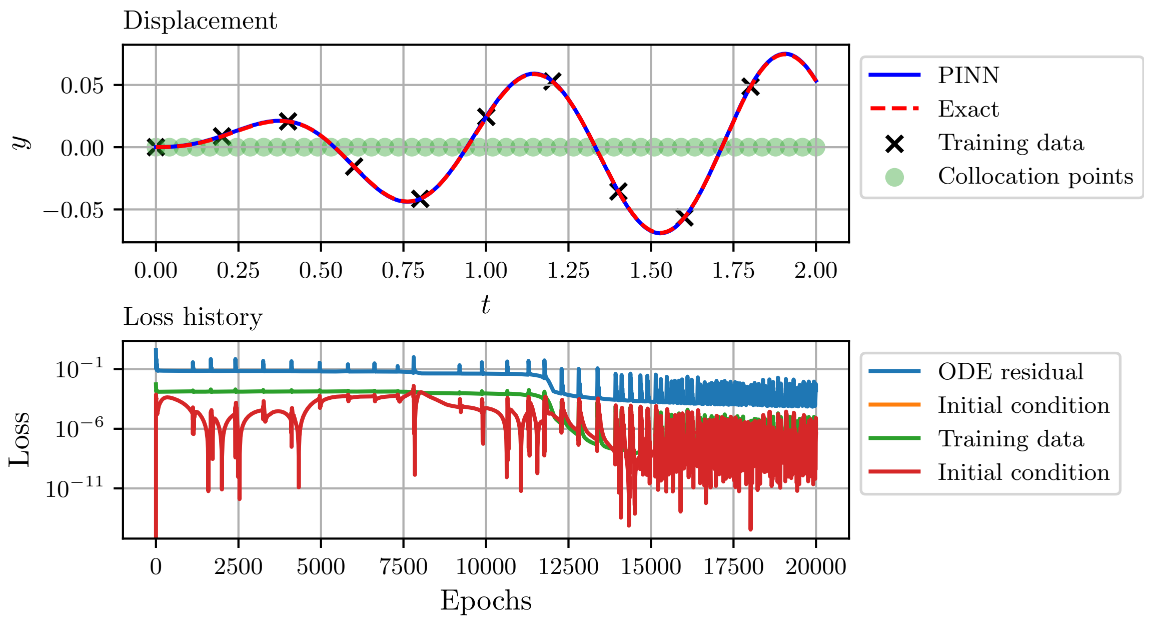


Figure 5.20: Case2, with data: lr=0.001, weight residual = 0.0005

Figure 5.20 shows that by including data points, the method is still able to find the solution. Therefore the inverse-solving capabilities of the neural network will be further investigated.

5.4.2 Inverse problem

Estimating the spring stiffness using simulated data from VIVANA-TD.

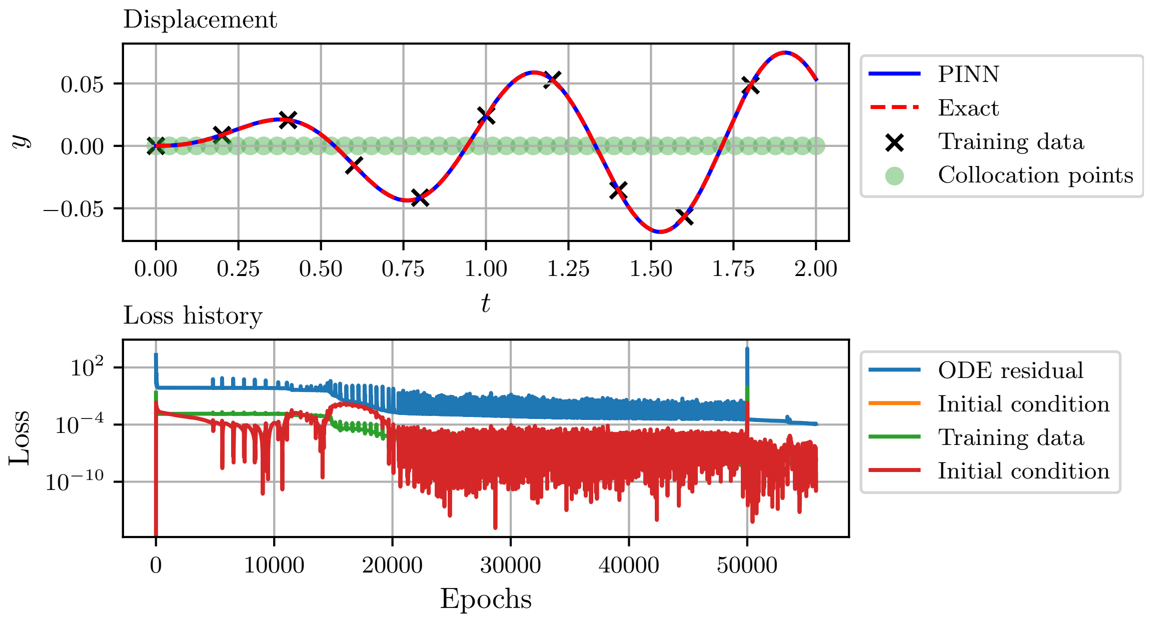


Figure 5.21: Unknown k. Initial guess k = 1200

Figure 5.21 shows that PINN is able to find the solution with one when provided 10 measurement points. The estimated k is the same as the real k. However, the initial guess is very close to the true value. Much more can be investigated, like the effect of initial guess, training time, and more. However, a more robust NN, optimizer, and weighting should be prioritized. Therefore no more investigation is conducted.

5.5 Summary

Different variants of the equation have been explored to investigate the challenges associated with training in case 2. It has been observed that setting a lower value for the parameter k allows the model to converge to the correct solution, indicating that the problem becomes simpler with a smaller k. This finding aligns with the results reported by Krishnapriyan et al. (2021). However, despite exploring various extensions, the method still needs to work on achieving convergence to the correct solution. Furthermore, attempts to improve the optimization process by incorporating automatic loss terms have yet to resolve the convergence issues. Different neural network architectures present another potential possibility for improving the training process. Exploring alternative architectures can improve the overall performance and effectiveness of the training.

One promising approach simplifies the problem by incorporating the initial condition as a hard constraint in the NN. The inclusion of these constraints improves convergence in PINNs. However, the effectiveness of this method is still influenced by the weighting assigned to the loss terms. Employing a larger neural network with hard constraints makes optimizing the PINN for case 2 feasible. The same approach also makes it possible to solve the inverse problem starting with an initial guess close to the actual value. These observations suggest that factors, such as the choice of neural network architecture or the optimizer used, play a significant role. Still, the method is interesting. However, more ML expertise and development are required. Some things that need to be made simpler are the design of the neural network. The neural network architecture is found empirical, requiring considerable time and knowledge about the problem. Therefore, further research is necessary to explore effective architectures, optimizers, and loss weighting schemes specialized for PINNs.

Chapter 6

Conclusion and future work

This thesis aimed to investigate hybrid analytic methods to improve VIV prediction. The theory underlying the VIV model VIVANA-TD is presented. Due to limitations in the model, particularly in the choice of empirical coefficients, a PINN approach has been suggested to improve the prediction. The method aims to improve the VIVANA-TD model with two concepts: adjusting the empirical coefficient based on response measurements and shaping the prediction to match the measurement data while maintaining the structure of the physical model.

PINNs offer a seamless way to combine data with prior knowledge, allowing the utilization of measurement data to improve existing models. Case study 1 shows that the PINN method can combine a simplified model with measurement data to improve predictions. The best result is when one of the parameters in the differential equation is unknown, and the PINN method uses the measurement points to find the parameter that matches the measurement points. When using the simplified model without trainable parameters, the model can satisfy the measurement points and keep the structure and characteristics of the simplified model. However, the result is very dependent on the choice of loss weighting. The method also shows that it can solve an ODE and find multiple parameters with sparse and noisy data.

Applying the same method in case studies 2 and 3 requires much more fine-tuning and modifications to get the model convergence to the correct solution. The training process becomes more challenging, making it a highly intricate task for the model to converge to the correct solution. For PINN to become a reliable tool in engineering applications, further research is needed to understand why PINN sometimes fails to train. Identifying possible failure modes and developing strategies to prevent them is crucial. It would greatly benefit the method to have a published guide that outlines these failure modes and provides recommendations for avoiding them. By improving the robustness of PINN, it can become a successful method in hybrid analytics, bridging the gap between real life and simulations. The method can easily be extended to include spatial dimension, custom NN architectures, and multi-fidelity data. Additionally, PINN can be a great model for digital twins due to its fast computational speed when trained.

Further work is needed to improve the robustness of the method. The method should also be expanded to include spatial dimensions (modeling of the beam). The expanded method should first be tested on simulated data and then measurements for model experiments. There is also the possibility of designing an algorithm that can weigh the measurement data and the physical model when there are differences between them. While PINN is an interesting hybrid analytics approach, other promising methods can improve physical models. Instead of focusing solely on improving the solution, one could also improve the differential equation. This is possible by using a Neural ODE, also called a universal differential equation (Christopher Rackauckas et al. 2021). Focusing on improving the differential equation will make it possible to use numerical solvers. In this approach, a NN is included directly in the differential equation. Thus, by using data, it can find a different term in the differential equation that compensates for the unmodeled physics. It could also be interesting to test multi-fidelity NN to improve VIV modeling.

Bibliography

Aloísio, Pina et al. (June 2013). 'ANN-based surrogate models for the analysis of mooring lines and risers'. In: *Applied Ocean Research* 41, pp. 76–86. DOI: 10.1016/j.apor.2013.03.003.

AlQuraishi, Mohammed (May 2019). 'AlphaFold at CASP13'. In: *Bioinformatics* 35.22, pp. 4862–4865. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz422. eprint: https://academic.oup.com/bioinformatics/article-pdf/35/22/4862/31501703/btz422.pdf. URL: https://doi.org/10.1093/bioinformatics/btz422.

Andersen, Martin Lieberkind et al. (2022). *Estimation of VIV-parameters based on Response Measurements and Bayesian Machine Learning Algorithms.* eng. URL: https://hdl.handle.net/11250/3058496.

Basir, Shamsulhaq and Inanc Senocak (Jan. 2022). 'Critical Investigation of Failure Modes in Physics-informed Neural Networks'. In: DOI: 10.2514/6.2022-2353. URL: https://doi.org/10.25142F6.2022-2353.

Cai, Shengze et al. (Apr. 2021). 'Physics-Informed Neural Networks for Heat Transfer Problems'. In: *Journal of Heat Transfer* 143.6. 060801. ISSN: 0022-1481. DOI: 10.1115/1.4050542. eprint: https://asmedigitalcollection.asme.org/heattransfer/article-pdf/143/6/060801/6688635/ht_143_06_060801.pdf. URL: https://doi.org/10.1115/1.4050542.

Chen, Cheng et al. (2021). 'Deep learning based on PINN for solving 2 DOF vortex induced vibration of cylinder'. In: *Ocean Engineering* 240, p. 109932. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2021.109932. URL: https://www.sciencedirect.com/science/article/pii/S0029801821012774.

Chen, Li-Wei and Nils Thuerey (2021). *Towards high-accuracy deep learning inference of compressible turbulent flows over aerofoils.* DOI: 10.48550/ARXIV.2109.02183. URL: https://arxiv.org/abs/2109.02183.

Chuang, PY (Nov. 2019). *Optimize TensorFlow & Keras models with L-BFGS from TensorFlow Probability — import pyChao.* https://pychao.com/2019/11/02/optimize-tensorflow-keras-models-with-l-bfgs-from-tensorflow-probability/. (Accessed on 04/25/2023).

Cuomo, Salvatore et al. (2022). *Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next.* arXiv: 2201.05624 [cs.LG].

Delizisis, Panagiotis et al. (June 2022). 'Numerical Investigation of the Effect of Vortex Induced Vibrations (VIV) Parameters on the Behavior of Submarine Power Cables: A Comparison With Scaled Down Experimental Results'. In: International Conference on Offshore Mechanics and Arctic Engineering Volume 7: CFD and FSI. V007T08A032. DOI: 10.1115/OMAE2022-79509. eprint: https://asmedigitalcollection.asme.org/OMAE/proceedings-pdf/OMAE2022/85925/V007T08A032/6929385/v007t08a032-omae2022-79509.pdf. URL: https://doi.org/10.1115/OMAE2022-79509.

DNV (n.d.). Wind energy - going offshore. https://www.dnv.com/to2030/technology/wind-energy-going-offshore.html [Accessed: 13.11.2022].

DNV-GL (2016a). Guideline on analysis of vortex-induced vibrations in risers and umbilicals. DNV.

DNV-GL (2016b). How digital tools and solutions can improve Subsea Integrity Management. DNV.

Fergestad, Dag, Svein Are Løtveit and Svein Sævik (2017). *Handbook on design and operation of flexible pipes.* https://www.sintef.no/en/latest-news/2017/updated-handbook-on-design-and-operation-of-flexible-pipes/. (Accessed on 12/12/2022).

Frank, Michael, Dimitris Drikakis and Vassilis Charissis (2020). 'Machine-Learning Methods for Computational Science and Engineering'. In: *Computation* 8.1. ISSN: 2079-3197. DOI: 10.3390/computation8010015. URL: https://www.mdpi.com/2079-3197/8/1/15.

Géron, A. (2017). *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media. ISBN: 9781491962299. URL: https://books.google.no/books?id=I6qkDAEACAAJ.

Glorot, Xavier and Y. Bengio (Jan. 2010). 'Understanding the difficulty of training deep feedforward neural networks'. In: *Journal of Machine Learning Research - Proceedings Track* 9, pp. 249–256.

Goodfellow, Ian, Yoshua Bengio and Aaron Courville (2016). *Deep Learning.* http://www.deeplearningbook.org. MIT Press.

Greco, Marilena (2022). *TMR 4215: Sea Loads Lecture Notes.*

Haghighat, Ehsan and Ruben Juanes (2021). 'SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks'. In: *Computer Methods in Applied Mechanics and Engineering* 373, p. 113552.

Hornik, Kurt, Maxwell Stinchcombe and Halbert White (1989). 'Multilayer feedforward networks are universal approximators'. In: *Neural Networks* 2.5, pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

IBM (2022). What is a digtal twin? https://www.ibm.com/topics/what-is-a-digital-twin [Accessed: 13.11.2022].

Jacquier, Pierre (2019). *pierremtb/PINNs-TF2.0: TensorFlow 2.0 implementation of Maziar Raissi's Physics Informed Neural Networks (PINNs).* https://github.com/pierremtb/PINNs-TF2.0. (Accessed on 05/09/2023).

Ji, Weiqi et al. (July 2021). 'Stiff-PINN: Physics-Informed Neural Network for Stiff Chemical Kinetics'. In: *The Journal of Physical Chemistry A* 125.36, pp. 8098–8106. DOI: 10.1021/acs.jpca.1c05102. URL: https://doi.org/10.10212Facs.jpca.1c05102.

Karniadakis, George (May 2023). *The Crunch Group – The collaborative research work of George Em Karniadakis.* https://sites.brown.edu/crunch-group/. (Accessed on 05/23/2023).

Karniadakis, George Em et al. (2021). 'Physics-informed machine learning'. In: *Nature Reviews Physics* 3.6, pp. 422–440. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00314-5. URL: https://doi.org/10.1038/s42254-021-00314-5.

Kharazmi, Ehsan, Dixia Fan et al. (2021). 'Inferring vortex induced vibrations of flexible cylinders using physics-informed neural networks'. In: *Journal of Fluids and Structures* 107, p. 103367. ISSN: 0889-9746. DOI: 10.1016/j.jfluidstructs.2021.103367. URL: https://www.sciencedirect.com/science/article/pii/S088997462100150X.

Kharazmi, Ehsan, Zhicheng Wang et al. (Aug. 2021). 'From Data to Assessment Models, Demonstrated through a Digital Twin of Marine Risers'. In: OTC Offshore Technology Conference Day 3 Wed, August 18, 2021. D031S035R003. DOI: 10.4043/30985-MS. eprint: https://onepetro.org/OTCONF/proceedings-pdf/21OTC/3-21OTC/D031S035R003/2523337/otc-30985-ms.pdf. URL: https://doi.org/10.4043/30985-MS.

Kim, Sang Woo et al. (2021). 'Prediction of deepwater riser VIV with an improved time domain model including non-linear structural behavior'. In: *Ocean Engineering* 236, p. 109508. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2021.109508. URL: https://www.sciencedirect.com/science/article/pii/S0029801821009069.

Kim, Sung Wook et al. (2021). 'Knowledge Integration into deep learning in dynamical systems: an overview and taxonomy'. In: *Journal of Mechanical Science and Technology* 35, pp. 1331–1342.

Krishnapriyan, Aditi S. et al. (2021). *Characterizing possible failure modes in physics-informed neural networks.* arXiv: 2109.01050 [cs.LG].

Lagaris, I.E., A. Likas and D.I. Fotiadis (1998). 'Artificial neural networks for solving ordinary and partial differential equations'. In: *IEEE Transactions on Neural Networks* 9.5, pp. 987–1000. DOI: 10.1109/72.712178. URL: https://doi.org/10.11092F72.712178.

Langen, Ivar and Ragnar Sigbjørnsson (1986). *Dynamisk analyse av konstruksjoner.*

Lee, Hyuk and In Seok Kang (1990). 'Neural algorithm for solving differential equations'. In: *Journal of Computational Physics* 91.1, pp. 110–131. ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(90)90007-N. URL: https://www.sciencedirect.com/science/article/pii/002199919090007N.

Lenail, Alex (2023). NN generator. https://alexlenail.me/NN-SVG/[Accessed: 1.12.2022].

Liang, Liang et al. (Jan. 2018). 'A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis'. In: *Journal of The Royal Society Interface* 15. DOI: 10.1098/rsif.2017.0844.

Lu, Lu, Xuhui Meng et al. (2021). 'DeepXDE: A Deep Learning Library for Solving Differential Equations'. In: *SIAM Review* 63.1, pp. 208–228. DOI: 10.1137/19M1274067. eprint: https://doi.org/10.1137/19M1274067. URL: https://doi.org/10.1137/19M1274067.

Lu, Lu, Raphael Pestourie et al. (2021). *Physics informed neural networks with hard constraints for inverse design.* arXiv: 2102.04626 [physics.comp-ph].

Lu, Zhou et al. (2017). *The Expressive Power of Neural Networks: A View from the Width.* DOI: 10.48550/ARXIV.1709.02540. URL: https://arxiv.org/abs/1709.02540.

McClenny, Levi and Ulisses Braga-Neto (2022). *Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism.* arXiv: 2009.04544 [cs.LG].

Meng, Xuhui and George Em Karniadakis (Jan. 2020). 'A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems'. In: *Journal of Computational Physics* 401, p. 109020. DOI: 10.1016/j.jcp.2019.109020. URL: https://doi.org/10.10162Fj.jcp.2019.109020.

Moradi, Sarvin et al. (2023). 'Novel Physics-Informed Artificial Neural Network Architectures for System and Input Identification of Structural Dynamics PDEs'. In: *Buildings* 13.3. ISSN: 2075-5309. DOI: 10.3390/buildings13030650. URL: https://www.mdpi.com/2075-5309/13/3/650.

Myers, Greg (2017). *Digital twin for marine drilling risers.* https://www.oedigital.com/news/446684-digital-twin-for-marine-drilling-risers. Accessed on 12/07/2022.

Myrhaug, Dag and Bjørnar Pettersen (2021). *Læreboken Havromsteknologi - NTNU.* https://www.ntnu.no/sf-marin/l-rebok. (Accessed on 05/28/2023).

Nasution, Fachri P., Svein Sævik and Janne K.Ø. Gjøsteen (2014). 'Finite element analysis of the fatigue strength of copper power conductors exposed to tension and bending loads'. In: *International Journal of Fatigue* 59, pp. 114–128. ISSN: 0142-1123. DOI: https://doi.org/10.1016/j.ijfatigue.2013.09.009. URL: https://www.sciencedirect.com/science/article/pii/S0142112313002703.

Ngom, Marieme and Oana Marin (2021). 'Fourier neural networks as function approximators and differential equation solvers'. In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 14.6, pp. 647–661. DOI: https://doi.org/10.1002/sam.11531. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sam.11531. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11531.

Nilsen-Aas, Christoffer et al. (May 2019). 'Flexible Riser Fatigue Counter Developed from Field Measurements and Machine Learning Techniques'. In: OTC Offshore Technology

Conference Day 1 Mon, May 06, 2019. D011S003R004. DOI: 10.4043/29531-MS. eprint: https://onepetro.org/OTCONF/proceedings-pdf/19OTC/1-19OTC/D011S003R004/1136151/otc-29531-ms.pdf. URL: https://doi.org/10.4043/29531-MS.

Nvidia (2023). *Modulus - A Neural Network Framework — NVIDIA Developer.* https://developer.nvidia.com/modulus. (Accessed on 05/09/2023).

Pawar, Suraj et al. (Jan. 2021). 'Physics guided machine learning using simplified theories'. In: *Physics of Fluids* 33.1, p. 011701. DOI: 10.1063/5.0038929. URL: https://doi.org/10.1063%5C2F5.0038929.

Rackauckas, Chris (Sept. 2020). *Introduction to Scientific Machine Learning through Physics-Informed Neural Networks - MIT Parallel Computing and Scientific Machine Learning (SciML).* https://book.sciml.ai/notes/03-Introduction_to_Scientific_Machine_Learning_through_Physics-Informed_Neural_Networks/. (Accessed on 05/01/2023).

Rackauckas, Christopher (July 2019). *The Essential Tools of Scientific Machine Learning (Scientific ML) - Stochastic Lifestyle.* https://www.stochasticlifestyle.com/the-essential-tools-of-scientific-machine-learning-scientific-ml/. (Accessed on 05/25/2023).

Rackauckas, Christopher et al. (2021). *Universal Differential Equations for Scientific Machine Learning.* arXiv: 2001.04385 [cs.LG].

Rai, Rahul and Chandan K. Sahu (2020). 'Driven by Data or Derived Through Physics? A Review of Hybrid Physics Guided Machine Learning Techniques With Cyber-Physical System (CPS) Focus'. In: *IEEE Access* 8, pp. 71050–71073. DOI: 10.1109/ACCESS.2020.2987324.

Raissi, M., P. Perdikaris and G.E. Karniadakis (2019). 'Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations'. In: *Journal of Computational Physics* 378, pp. 686–707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Raynaud, Gaétan, Sébastien Houde and Frédérick P. Gosselin (Aug. 2022). 'ModalPINN: An extension of physics-informed Neural Networks with enforced truncated Fourier decomposition for periodic flow reconstruction using a limited number of imperfect sensors'. In: *Journal of Computational Physics* 464, p. 111271. DOI: 10.1016/j.jcp.2022.111271. URL: https://doi.org/10.10162Fj.jcp.2022.111271.

Riemer-Sørensen, Signe (2023). *Hybrid analytics - AI.* https://www.sintef.no/ekspertise/digital/hybrid-analytics/. (Accessed on 05/25/2023).

Rueden, Laura von et al. (2021). 'Informed Machine Learning - A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems'. In: *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1. DOI: 10.1109/tkde.2021.3079836. URL: https://doi.org/10.1109/tkde.2021.3079836.

Sadeghi Eshkevari, Soheil et al. (2021). 'Physics-based neural architecture design for nonlinear structural response modeling and prediction'. In: *Engineering Structures* 229, p. 111582. ISSN: 0141-0296. DOI: https://doi.org/10.1016/j.engstruct.2020.111582. URL: https://www.sciencedirect.com/science/article/pii/S0141029620341833.

SciPy (n.d.). *scipy.integrate.odeint — SciPy v1.9.3 Manual.* https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html. (Accessed on 12/15/2022).

Sintef (n.d.). *SIMA.* https://www.sima.sintef.no/. (Accessed on 04/19/2023).

Stadtman, Florian et al. (2023). *Digital twin and asset management for wind energy.* ttk29, Adil Rasheed.

Staff, GUunnar (Mar. 2021). *Physics and AI hybrid delivers working AI for industry.* https://www.cognite.com/en/blog/physics-and-ai-hybrid-deliver-working-ai-for-industry. (Accessed on 05/25/2023).

Sumer, B Mutlu and Jørgen Fredsøe (2006). *Hydrodynamics Around Cylindrical Structures.* WORLD SCIENTIFIC. DOI: 10.1142/6248.

Sundararaman, S., R. Thethi and M. Hill (Apr. 2018). ‘Data Driven Virtual Sensors for Riser Prognostic Integrity Management’. In: OTC Offshore Technology Conference Day 1 Mon, April 30, 2018. D011S014R006. DOI: 10.4043/28732-MS. eprint: https://onepetro.org/OTCONF/proceedings-pdf/18OTC/1-18OTC/D011S014R006/1193809/otc-28732-ms.pdf. URL: https://doi.org/10.4043/28732-MS.

Szegedy, Christian et al. (2014). *Going Deeper with Convolutions*. arXiv: 1409.4842 [cs.CV].

TechnipFMC (2023). *Deep Purple™ Pilot - TechnipFMC plc*. https://www.technipfmc.com/en/what-we-do/new-energy-ventures/hydrogen/deep-purple-pilot/. (Accessed on 12/09/2022).

Tensorflow (2023). *Making new Layers and Models via subclassing TensorFlow Core*. https://www.tensorflow.org/guide/keras/custom_layers_and_models. (Accessed on 04/25/2023).

Triantafyllou, Michael S. et al. (2016). ‘Vortex-Induced Vibrations’. In: *Springer Handbook of Ocean Engineering*. Ed. by Manhar R. Dhanak and Nikolaos I. Xiros. Cham: Springer International Publishing, pp. 819–850. ISBN: 978-3-319-16649-0. DOI: 10.1007/978-3-319-16649-0_36. URL: https://doi.org/10.1007/978-3-319-16649-0_36.

Wang, Sifan, Shyam Sankaran and Paris Perdikaris (2022). *Respecting causality is all you need for training physics-informed neural networks*. arXiv: 2203.07404 [cs.LG].

Wang, Sifan, Yujun Teng and Paris Perdikaris (2020). *Understanding and mitigating gradient pathologies in physics-informed neural networks*. arXiv: 2001.04536 [cs.LG].

Wikipedia (2023). *GPT-3 - Wikipedia*. https://en.wikipedia.org/wiki/GPT-3. (Accessed on 06/09/2023).

Wong, Jian Cheng et al. (2022). ‘Learning in Sinusoidal Spaces with Physics-Informed Neural Networks’. In: *IEEE Transactions on Artificial Intelligence*, pp. 1–15. DOI: 10.1109/tai.2022.3192362. URL: https://doi.org/10.11092Ftai.2022.3192362.

Wu, Jie (2022). Vortex-Induced Vibrations (VIV), Power Point.

Wu, Jie et al. (Aug. 2020). ‘Time Domain VIV Analysis Tool VIVANA-TD: Validations and Improvements’. In: International Conference on Offshore Mechanics and Arctic Engineering Volume 8: CFD and FSI. V008T08A031. DOI: 10.1115/OMAE2020-18759. eprint: https://asmedigitalcollection.asme.org/OMAE/proceedings-pdf/OMAE2020/84409/V008T08A031/6607028/v008t08a031-omae2020-18759.pdf. URL: https://doi.org/10.1115/OMAE2020-18759.

Yuan, Fuh-Gwo et al. (Apr. 2020). ‘Machine learning for structural health monitoring: challenges and opportunities’. In: p. 2. DOI: 10.1117/12.2561610.

Zubov, Kirill et al. (2021). *NeuralPDE: Automating Physics-Informed Neural Networks (PINNs) with Error Approximations*. DOI: 10.48550/ARXIV.2107.09443. URL: https://arxiv.org/abs/2107.09443.

Appendix

A Different neural network configurations

Model	Layers	Neuron per layer	Training time	MSE training	MSE validation
1	3	10	75.0075	0.0434822	0.0241561
2	3	20	76.7692	0.000177694	0.00390117
3	3	40	65.8685	0.000166874	0.00362876
4	3	100	88.1377	0.000217707	0.00372386
5	4	10	59.7509	0.000114735	0.00260235
6	4	20	59.8184	5.49479e-05	0.00175854
7	4	40	69.4466	0.000150192	0.00344858
8	4	100	139.146	1.25112	0.0085477
9	6	10	67.7607	0.000829788	0.00613894
10	6	20	75.6862	0.000457474	0.004464
11	6	40	104.244	0.00365923	0.00562641
12	6	100	317.298	1.25112	0.0106423

B Python code

The code can be found at the following link:
https://github.com/oleholme/Mthesis.git

NTNU

Norwegian University of
Science and Technology