Physics-Informed Machine Learning for Fluid Dynamics

Jose Florido

University of Leeds

School of Computing

Submitted in accordance with the requirements for transfer to the degree of

Doctor of Philosophy

July, 2023

Declaration of Academic Integrity

The candidate confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to the work of others.

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

The right of Jose Florido to be identified as Author of this work has been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

Signed: Jose Florido Date: 3rd July, 2023

Contents

1 Introduction and Literature Review **1**

1.1 Introduction 1

1.1.1 Numerical Methods, Machine Learning 1

1.1.2 Physics Informed Neural Networks 2

1.2 Literature Review 5

1.2.1 Machine Learning and Scientific Computation 5

1.2.2 Physics Informed Neural Networks 8

1.2.3 Relevant Research 10

2 Research Focus **16**

2.1 Research Aim 16

2.2 Objectives 16

3 Methods **18**

3.1 Python 18

3.2 TensorFlow2, PyTorch 18

3.3 DeepXDE 19

4 Progress **21**

4.1 Toy Problem - Initial NN Coding Training 21

4.2 Python and Tensorflow 2 Training 22

4.3 PINNs for Burger's Equation - Hyperparameter Investigation 24

4.3.1 Generating Ground Truth through FDM 25

4.3.2 Hyperparameter Investigation 26

4.4 PINNs for Burger's Equation - Bias Investigation 29

4.4.1 Simple Biasing 30
4.4.2 Refined Biasing 32
4.4.3 Variance 34
4.5 Baseline Results - Pseudo-Random sampling 37
4.6 Adaptive Sampling - Validation 38
4.7 Ongoing Work 42

5 Research Plan **44**
5.1 Until End of Calendar Year 44
5.2 Medium & Long Term 46
5.3 Publishing Goals and Conferences 47
5.4 Gantt Chart 49

References **i**

A Appendix 1 **vi**
A.1 Responsible Research & Innovation (RRI) Plan vi
A.2 Training Needs and Plan vi
A.3 Data Management Plan vii
A.4 Expenditure Plan viii

Abbreviations

ML	Machine Learning	NN	Neural Network
PIML	Physics Informed Machine Learning	PINN	Physics Informed Neural Network
FE	Finite Element	FDM	Finite Difference Method
CFD	Computational Fluid Dynamics	HPC	High Performance Computing
DNS	Direct Numerical Simulation	CoD	Curse of Dimensionality
RANS	Reynolds-Averaged Navier-Stokes	RAR	Residual-based Adaptive Refinement
RAD	Residual Based Distribution	RAR-D	RAR with Distribution
PDF	Probability Density Function	TF2	TensorFlow 2
PDE	Partial Differential Equation	ODE	Ordinary Differential Equation
LHS	Latin Hypercube Sampling		

Chapter 1 : Introduction and Literature Review

1.1 Introduction

1.1.1 Numerical Methods, Machine Learning

The study of fluid dynamics plays a role across many scientific and engineering disciplines, and provides insights into the behaviour of fluids in a wide range of scenarios and applications. Traditionally, the solution of fluid dynamics problems has relied on numerical methods ranging from Finite Element (FE) methods to more advanced Computational Fluid Dynamics (CFD) simulations via software such as Ansys or OpenFoam. While our understanding today of fluid flow phenomena is underpinned by research using these numerical methods, they are not without their limitations. This introduction will explore how Physics Informed Neural Networks (PINNs) have emerged from the field of Machine Learning (ML) as a promising alternative method for solving fluid dynamics problems.

Classic numerical methods have benefited from decades of research that have led to them being robust and trustworthy. However, they are not without limitations. As the complexity of the problems increases, the Curse of Dimensionality (CoD) poses a significant challenge. This curse of dimensionality refers to the exponential increase in computational cost as the number of dimensions or variables in a problem grows. With advancements in hardware capabilities and increasing availability of High Performance Computing (HPC), even expensive computational simulations such as Large Eddy Simulations (LES) and Direct Numerical Simulations (DNS) which fully resolve increasingly smaller scales have become feasible as research tools. However, the computational costs associated with these approaches remain prohibitively expensive for most practical applications and design processes (Molland and Turnock, 2007), and the curse of dimensionality remains a formidable obstacle for high-dimensional problems. Although the development of HPC technology is an ongoing research topic of its own (Al-Shafei et al., 2022), recently attention has been given to alternative approaches utilising machine learning methods.

In recent years, there has been a remarkable rise in the utilization of machine learning algorithms for a variety of applications. Deep neural networks (prefix 'deep' indicating multiple network layers) in particular have demonstrated remarkable capabilities in pattern recognition, data analysis, and decision-making tasks. The ability to

learn from data and extract meaningful patterns has led to significant advancements in image and speech recognition, natural language processing, and many other fields, some of which are discussed in section 1.2.1. Applications such as ChatGPT Brown et al. (2020) have billions of parameters, effectively being a very high-dimensional problem without being limited by the CoD. This characteristic is why it was seen to have potential and why efforts have been made to merge ML approaches with CFD (Section 1.2.1). Once trained, machine learning models can generate solutions very quickly compared to numerical methods. However, training these models requires large amounts of data of high-resolution in order to produce results of comparable accuracy. Often this requires having solution data obtained from numerical models to begin with, and the ability of the models to generalise to different problems is an important issue. Cueto and Chinesta (2023) argues "neural networks are not very popular among the scientific community in general" due to other well known problems with ML such as overfitting, or the large impact small perturbations in data can have on results.

1.1.2 Physics Informed Neural Networks

Physics Informed Neural Networks (PINNs) emerge as a specific type of machine learning algorithm that combines the advantages of neural networks whilst minimising the downsides by incorporating a priori knowledge of governing physics. By incorporating physical information into the neural network architecture, PINNs can offer efficient and accurate predictions while significantly reducing the computational burden and reducing the large data requirements. By attempting to satisfy the physics by construction, they are also innately more generalisable, not being as limited by the data introduced. This paradigm shift towards integrating machine learning techniques opens up new avenues for tackling high-dimensional fluid dynamics problems that are beyond the capabilities of traditional numerical methods alone.

Instead of minimising a loss that is defined as the difference between some data and the network's prediction, PINNs work by specifying a loss function that is a result of the governing Partial Differential Equation (PDE) and boundary condition residuals. These residuals are evaluated at specific points in the domain, called 'collocation points'. A good schematic of this by Lu et al. (2021) is shown in figure 1.1 below. Evaluating the PDE residuals is possible thanks to automatic differentiation, which is a computational technique that allows for the efficient and accurate computation

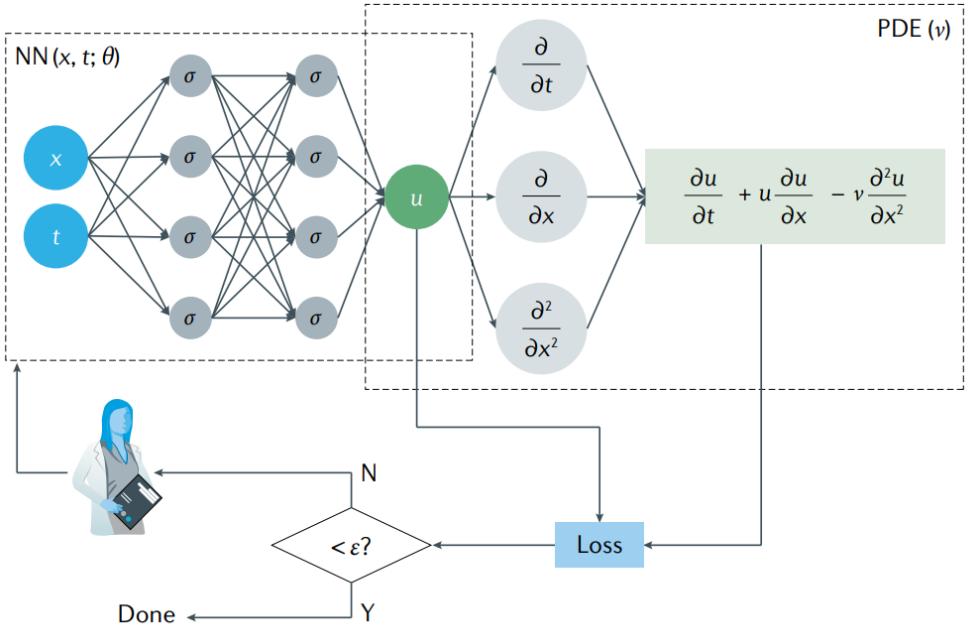


Figure 1.1: PINN diagram (Lu et al., 2021) for the example of Burgers' Equation. The Neural Network (NN) produces a velocity field for a domain x, t, and the loss function incorporates the PDE residual. The loss is minimised via a gradient descent optimiser, which trains the network until the loss is sufficiently small.

of derivatives by automatically evaluating the gradients of mathematical expressions with respect to their input variables. Unlike numerical differentiation methods, which approximate derivatives by finite differences, automatic differentiation provides exact derivatives by applying the chain rule, accurately computing the relevant derivatives. Formulations with only the PDE and boundary condition residuals in the loss function can thus solve for problems without the use of data (for example Sun et al. (2020)), or greatly reducing the dependency on large data.

The advantages of incorporating physics makes Physics-Informed Machine Learning (PIML) stand out as an independent method for solving forward problems, that is being able to predict solutions for a given problem satisfying underlying physics. To add to this, ideas from ML can be used to structure the problem so that the network can be run to solve for the variables in the equation. Setting up this type of reverse problem allows for solving inverse fluid dynamics problems where unknown properties or parameters of the system can be inferred from given data. This is an incredibly

powerful use-case that would be difficult to do with traditional numerical methods, and can be used for inferring parameters, reconstruct missing information, or enhance noisy images obtained from low-resolution sensors.

So, what are the key issues that need to be addressed regarding PINNs? One significant drawback is the lack of interpretability. Understanding how the network arrived at a particular solution is crucial in many applications, especially in those with higher safety margin requirements. Another challenge is the generalisation capability of the networks. For instance, multi-scale problems might be harder to generalise to, necessitating unique adaptions to the vanilla PINN such as that by Oppenheimer et al. (2023). This limitation restricts the range of applications where PINNs can be effectively employed and if retraining is necessary for each specific case, the cost efficiency of PINNs compared to traditional methods becomes less competitive.

While PINNs possess both strengths and weaknesses, it is clear that there are specific applications and niches where its advantages make them the preferred method. However, due to the novelty of PINNs, several aspects still need to be addressed in the literature. One of these areas is the lack of theoretical understanding and the need for implementation standards and benchmarks. Long term, a solid theoretical foundation and proofs that deep learning approaches can overcome CoD will be necessary. Furthermore, there is a need for interpretability of results and for and uncertainty quantification, particularly in applications that require stringent safety measures, such as civil engineering or medical applications. However, the large amount of interest in the subject means that many solutions are being researched to all the above problems - as just one example, recently a lot of research into implementing Bayesian methods has been carried out to address the need for more robust uncertainty estimates in PINN predictions.

The primary challenge for this PhD was identifying a specific area under the broad umbrella of PINNs where I could contribute meaningfully. There are many approaches to implementing physics in the literature, as will be seen shortly in the literature review. The distribution of the collocation points, the points where the physical information is gathered during the training of the network, was identified as one promising avenue of research for general improvement of PINNs. This work echoes research on adaptive meshing done in classical FE methods, and as a result there are many exciting opportunities to incorporate methods from other fields. An improved method of collocation

point distribution could be incorporated into various PINN architectures, as most rely on collocation points for training, and so could be very impactful. This chosen avenue overall holds significant potential for publication (with some work on the topic having been published already, reviewed in section 1.2.3), which means there is enough room for me to make original contributions for my PhD thesis.

1.2 Literature Review

This literature review is separated into three main sections. Section 1.2.1 will cover literature in the broader context of this PhD, covering the types of problems data driven methods have been used for generally, and applications to fluid dynamics. Section 1.2.2 will focus on PINNs; reviewing the more important papers on the topic and highlighting original contributions of most note. In the last section, 1.2.3, the papers most relevant to the planned work are discussed - covering literature on the methodology, adaptive collocation point methods, and other areas of work that might be tackled in the future.

1.2.1 Machine Learning and Scientific Computation

Machine Learning

The concept of machine learning is by no means a new one, with early publications going back as far as Nilsson (1965) describing the theory and possibilities of the machine learning methods. It took a few decades for machine learning applications to emerge for simple tasks such as handwriting recognition in Lecun et al. (1998). In the two decades since however, the field of machine learning has developed significantly, reaching a point where today fully developed ML algorithms (also referred to as 'neural networks') are used in commercial applications. One popular example in academia has been the development of ChatGPT, a deep network utilising a highly specialised model designed for task-agnostic natural language processing (NLP) applications. The abilities of the GPT3 network are explained in detail Brown et al. (2020), from where its significant that the model is noted to work with 175 billion parameters; an incredibly high-dimensional problem that is an extreme example of the ability for machine learning methods to defy the curse of dimensionality.

The potential of ML is not limited to NLP, image processing and other similar applications that have been explored in computer science. In Weinan et al. (2022), the

authors review numerical and theoretical advances on numerical algorithms for solving PDEs in very high dimension. This paper is great in its uniqueness at tackling the mathematics and theory behind neural networks. It comprehensively cover what analysis have been carried out in the literature in testing deep learning networks' ability to overcome the Curse of Dimensionality in the solution of PDEs. Citing a total of 165 papers, it's a great starting point for a more mathematically focused study of deep learning. Overall, the message from this 2021 paper is that the study of very high-dimensional problems using deep learning is very promising for scientific computing, a feeling which is certainly paralleled by the vast amount of papers showing the implementation of machine learning for scientific computing problems - including across many fluid dynamics applications.

ML in SciComp and Fluids

One branch of ML for fluids focuses on combining ML with existing numerical methods. The aim here is to take the advantages of ML and use them to accelerate the more expensive steps of numerical processes, without impacting the robustness and accuracy of traditional CFD. Vinuesa and Brunton (2022) reviews the different ways in which this paradigm has been researched in the literature as of 2021; namely either in accelerating DNS via ML-driven hyperresolution, to compensate for DNS' high costs; improving the accuracy of faster turbulence modelling methods (such as Reynolds-Averaged Navier-Stokes (RANS)); or using ML to develop reduced-order models, using deep networks and autoencoders to drive the dimensionality reduction, as these have shown to improve the efficiency of the reduced-order models. It's unfair to make sweeping statements on all these methods, as the way the advantages of ML are leveraged vary from paper to paper, but the outlook is that a lot of these areas have potential for improvement. The main drawbacks listed are the lack of established benchmark systems for fluid simulations using ML methods and the importance of having high quality available data. Vinuesa and Brunton (2022) mentions here the emergence of PINN, the focus of the next section, as an alternative family of methods to traditional numerical methods, due to how these can also be used for flow simulations without depending on traditional numerical methods.

Wang et al. (2017) uses what they called a physics informed ML approach to solve industrial flows. Their approach models the flows first using RANS models, and then

applies ML to the results to reconstruct Reynolds stresses. Here, their goal is to improve RANS prediction capabilities and at a glance they are successful in doing this even for scenarios with slightly different geometries. The difference in geometry is quite limited, and the data required to train the ML network also needs to be of sufficiently similar cases so their random-forests regression technique is successful. The bigger catch - the ML is trained on DNS data, relying on a database of DNS simulations being available so that the training data is of sufficiently high resolution. There are other examples of ML being used for solving scientific problems outside of in a hybrid approach with traditional CFD. Han et al. (2018) utilises a deep learning approach ('deep' again meaning multi-layered neural network models) to attempt to solve high-dimensional PDEs, creating an algorithm with a lot of potential for problems in areas of economics and finance that have to deal with high numbers of parameters. The potential to solve high dimensional problems is a recurring feature of deep learning based methods .Weinan and Yu (2018)'s Deep Ritz Method is another example of such a network, including an example solution of the Poisson Equation in high dimensions.

For more specific fluids applications, the review on hemodynamics research by Taebi (2022) does a comprehensive job of covering methods using deep learning for fluids; mentioning how dimensionality reduction capabilities of deep learning can be used for creating effective reduced-order models, and for reconstructing noisy or missing information from given data through super-resolution methods. The characteristics of ML algorithms make them excel at tasks like image recognition in regular computing problems. Applied to fluids, those characteristics mean deep learning can be used for super-resolution of noisy or low quality data.

A significant part of Taebi (2022) also discusses a specific deep learning approach that has extensively been researched by different researchers looking at hemodynamics, namely physics-informed neural networks. These are touted as a promising direction for research, and many papers referenced by Taebi (2022) researching the application of PINNs to hemodynamics. So why are PINNs being chosen over ML approaches so far discussed?

1.2.2 Physics Informed Neural Networks

Vanilla PINNs and Variations

Physics Informed Neural Networks are first introduced by Raissi et al. (2019). In this first implementation, they are already showed to be data-efficient, and able to solve both forward problems and reverse problems. The paper containing a variety of example solutions for both problems of both types, showing the method's high potential to work in a variety of classical problems from fluid mechanics to quantum mechanics. Despite the potential, the authors do echo some of the concerns mentioned earlier about the lack the of maturity compared to classic numerical methods, highlighting many of the questions about how different networks parameters would influence performance, as well as questions about how the algorithms might struggle with deeper architecture or for higher order differential operators. In this sense, this paper does a great job of highlighting the achievements of PINNs whilst remaining critical and realistic about the obstacles in the way of the technology.

Raissi et al. (2019) garnered an incredible amount of attention, with over 2000 citations to date, with 1022 in 2022. PINNs themselves have become an independent subject within Deep Learning academic literature, being covered in books on computational science such as Kollmannsberger et al. (2021). As a result of the large amount of literature surrounding the topic, it would be a near impossible task to comprehensively cover all the subsequent work. Review papers like Karniadakis et al. (2021) take a high-level approach. Karniadakis et al. (2021) does a thorough introduction to the principles behind PINNs, and discusses limitations, merits, applications, trends and the outlook into the future.

Cuomo et al. (2022) is another good review that helps distill the more significant outputs, performing a comprehensive study of the research citing an astounding 202 references. Cuomo et al. (2022) notes that most work has focused on investigating the impact of different hyper-parameters and design choices on the efficacy of the PINN architecture, as well as many studies on the different applications of PINNs. However, the main goal of the article was to characterize the many variations of PINNs that have since appeared in the literature. For this reason, it is one of the best starting points for understanding the work that has been done so far in PINNs.

'Physics-Constrained' networks presented by Geneva and Zabaras (2020) are one such variation on the vanilla PINN. They are built with a Bayesian framework that uses

information on the governing PDE in the loss function, whilst being able to provide uncertainty qualification for aleatoric and epistemic uncertainty (error due to noise and due to model uncertainties, respectively).

In Liu and Wang (2021), a physics informed approach is implemented in a model-based reinforcement learning framework in order to alleviate some of its downsides. Deep neural operators, first proposed by Lu et al. (2021), focus on learning representations present in the data. As Goswami et al. (2022) explores in their review of these networks, information of governing equations can also be incorporated to help the generalisation abilities and address the prohibitively expensive reality of acquiring data for many science and engineering problems.

These publications on more specialised PINNs architectures have a flaw that they don't tend to address more foundational questions of how to optimise the PINNs network. In exchange however, they offer benefits like better quantifiable error or slightly better generalisation for specific applications, which can still be helpful in developing a broader understanding of PINNs.

PINNs in fluids

Some of the literature already covered in this review has either been tested on problems related to fluid dynamics or acknowledged the high potential of PINNs for solving fluid dynamics problems. This section will highlight some publications that had a focus on applying PINNs to fluid dynamics applications.

The first publication of relevance is Mao et al. (2020)'s work investigating the effectiveness of PINNs for high-speed flows. Interestingly, they investigate the effect of clustering collocation points manually near discontinuities for their problems and note the benefits of this (using a posteriori knowledge of the solution). Their conclusion was that PINNs are not competitive with traditional numerical methods for the forward solving of problems. However, they are capable of solving inverse problems utilising PINNs, a task "that cannot even be solved with standard techniques." It's a good demonstration of the potential of PINNs even without the benefit of optimised architectures and experience that will come as PINNs mature as a field.

Work by Cai et al. (2021) covers applications of PINNs to heat transfer problems, covering many convective flow problems. The ability of PINNs to solve ill-posed inverse problems is also noted, which in that specific field is noted as being an "omnipresent

problem [due to] unknown thermal boundary conditions".

Zhu et al. (2021) looks at a more specific problem, making a first attempt at implementing a PINN to predict temperature and fluid dynamics of melt pools encountered in the the specific field of metal additive manufacturing. The paper poses a very complex problem, and finds some initial success that the authors claim demonstrates PINN's potential for the complex modelling necessary for this field. However, the PINN is not tested on the full problem, and the authors propose further work to be done incorporating additional PDEs to fully model the entire process.

Sun et al. (2020) examines a different specific application of PINNs in the field of clinical diagnosis. PINNs characteristics make it greatly suited for biomedical engineering, where there is a need for cost-effective models suitable for real-time applications. Data available is often insufficient for training traditional machine learning in these applications, a problem which Sun et al. (2020) tackles with a physics constrained PINN network that is shown to have excellent agreement with a traditional numerical simulation of aneurysmal flows - without simulation data to train on.

A dedicated review of PIML in fluid mechanics was recently published bySharma et al. (2023). This review serves as a great introduction to the topic and even provides a case study to demonstrate some of the advantages of incorporating physics via showing the results. Whilst it has the benefit compared to reviews already mentioned of being published very recently, the paper largely focuses on explaining the context and underpinning methods behind PIML. As a result there isn't much discussion of individual publications in the field of fluid dynamics, as it is not really the goal of it to review individual papers but the topic broadly. That said, it is still definitely a good source to turn to for a more in-depth discussion of the potential of PIML for fluid dynamics.

1.2.3 Relevant Research

Methods

Before covering the literature on adaptive collocation point sampling, the current focus of the PhD, mention must be made on the underlying software that enables the research. PyTorch Paszke et al. (2019) and Tensorflow Abadi et al. (2016) are two of the primary Deep Learning Frameworks for python, which offer tools to enable the building and training of neural networks for a variety of uses. Using these as the back-end framework, there are a multitude of software libraries that are further used for enabling the

user to easily define NN architectures. There are many to choose from, mostly based on Python. However, there are still some such as Rackauckas and Nie (2017), which enables coding for Julia. These software libraries are usually built with different implementations in mind - SimNet (Hennigh et al., 2020) by NVIDIA for example is built to best take advantage of NVIDIA GPUs for performance, but its black-box nature makes it less suitable for research by complicating the debugging process and limiting customisability. Many other examples of software libraries have been developed over time, and review papers like Karniadakis et al. (2021) do a good job of outlining these.

The most relevant software library for this project is DeepXDE, introduced by Lu et al. (2021), as it has been used by researchers in the field of adaptive sampling already (Wu et al., 2023). Lu et al. (2021)'s DeepXDE is designed to be a research tool that aids in machine learning development, and is especially useful for the field of PINNs research by having been designed around this in mind. The paper shows its ability to run PINNs effectively on different benchmarks and showcasing how the tools it provides can simplify the more tedious processes of domain construction and problem definition.

Adaptive Sampling

As mentioned in the introduction, the main research question tackled by this PhD concerns the distribution of collocation points in PINNs. In the literature, the default approach was to randomly place these points throughout the domain. This can be seen in the original paper by Raissi et al. (2019), but is also present in more recent literature like in figure 1.2, from Krishnapriyan et al. (2021). However, it is known from other fields such as the stochastic PDE community that uniform random sampling is certainly not the optimal solution. This has led to pseudo or quasi-random designs such as Sobol sequences or Latin Hypercube Sampling being developed for specific applications in areas like statistical analysis. Simply, these distribution methods ensure the points cover the entire domain in an efficient manner, following rules that prevent redundant points being placed in the same location, or areas of the domain not having any points. Whilst we can instinctively conjecture that these pseudo-random sampling methods must be better than random sampling, the default so far in the literature on PINNs has been to simply use random sampling.

The default collocation point algorithm in Raissi et al. (2019) never changes the positions of these points throughout the training of the network. One alternative approach

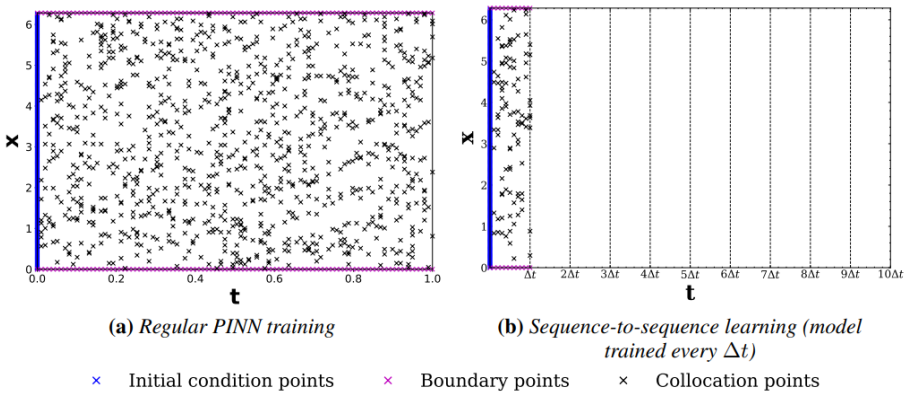


Figure 1.2: Figure from Krishnapriyan et al. (2021), where the collocation points throughout the domain can be seen to have been placed randomly.

is to randomly or pseudo-randomly distribute the points multiple times throughout the training process, which is referred to as re-sampling. If this re-distribution is done randomly, the associated costs are very low but can potentially improve performance. Lastly, adaptive sampling aims to re-distribute the points like re-sampling methods, but utilising information obtained from the network during the training process to guide where to place these points. The first adaptive sampling method in the literature appears in Lu et al. (2021), which introduces Residual-Based Adaptive Refinement (RAR). In RAR, the idea is to replicate FE refinement methods such as the work by Ainsworth and Oden (2000), repeatedly adding new points in locations where the PDE residual (as calculated by Monte Carlo integration) is greatest. The effects are tested on the 2D Burgers' Equation. The main focus of the paper however is on the DeepXDE framework described in the previous section, and the investigation on the impact of RAR is very limited. A slight improvement to error is shown compared to using a random distribution; which as described earlier has a few issues that could make it worse than inexpensive pseudo-random methods or random-resampling methods.

Follow up work by the same research group on 'gradient-enhanced' PINNs, or gPINNs, (Yu et al., 2022) better illustrates the advantages of RAR. Again, the focus of the paper is not on the adaptive sampling method, this time on a novel architecture aiming to improve on the original PINN via inserting gradient information into the loss function. Comparisons they make show gPINNs to improve accuracy of PINNs, performing even better with RAR. Significantly though, the inclusion of just RAR on

PINN is shown to have a much greater effect on accuracy than utilising gPINNs, showing that simple improvements to the utilisation of collocation points can still yield good performance improvements for the solution of Burgers and the Allen-Cahn equations.

The study by Wu et al. (2023) introduces two new methods. Residual-based Adaptive Distribution (RAD) is a smart resampling technique that utilizes a Probability Density Function (PDF). It outperforms alternative methods by incorporating an additional term, denoted as "c", which ensures that the entire domain is not ignored. This is in contrast to Nabian et al. (2021) paper, where the focus is solely on areas with high residuals. Another method proposed by Wu et al. is Residual-based Adaptive Refinement with Distribution (RAR-D), which is a refinement method that also employs the same PDF, but this time not requiring the global distribution hyperparameter c. The paper is highly valuable as it not only presents their own approaches but also conducts a systematic review of existing methods in the field.

However, a limitation of Wu et al. (2023)'s methods is the lack of information regarding computational time. Although their conclusions suggest that RAR-G may be more suitable for cases with "computational limitations," the exact computational cost difference between RAD and other techniques remains unclear. This omission makes it difficult to determine the specific applications where RAD might be prohibitively expensive or beneficial for quick and rough estimations.

The inclusion of two hyperparameters into the unique PDF dictating the sampling of RAD and RAR-D makes the PDF a generalised version of other work in the literature. The authors do a thorough comparison with other simple implementation such as Nabian et al. (2021), employing importance sampling, which they show is an example of the RAD method with c = 0. However, there are still novel elements to other previous research. In Subramanian et al. (2022), the authors proposes a cosine annealing strategy that involves sampling a fraction of points uniformly and another fraction non-uniformly. The specific details of the non-uniform sampling are not mentioned, but the entire approach is still novel by considering not only residuals but also loss gradients.

Although the advantages of smart sampling for improving accuracy are evident from the studies mentioned above, the computational costs associated with these methods remain unclear. Reference to steps taken is made in Wu et al. (2023) which is not useful as the time taken by a step can vary a lot depending on other factors. Although clock time is highly dependent on hardware, having this would allow to at least evaluate the

relative cost of adaptive vs non-adaptive methods. There are other several areas that require further exploration, including the development of more sophisticated sampling approaches. These sophisticated approaches could involve investigating gradients of flow or incorporating useful techniques from other numerical methods employed in relevant fields.

For example, grid adaptive strategies have been an important part of improving computational efficiency and accuracy in traditional numerical simulations. And whilst the PINNs approach is point based and therefore mesh-less, lessons could be learned from the research previously done on adaptive sampling in FEM. One interesting opportunity is goal-oriented meshing techniques (Venditti and Darmofal, 2000), that can yield more accurate predictions of outputs of interest by adapting the grid accordingly. This error estimation strategy was very influential and follow up work by other authors such as Hart et al. (2011) for example demonstrate the benefits of applying this method to an elastohydrodynamic lubrication problem. A similar paradigm could be explored with collocation points, exploring whether PINNs could be trained cost-effectively solve for functional forward problems by designing a goal-based adaptive sampling method.

There are other questions such as: how can we know enough steps have been taken that we can trust the residual information and use it to refine the point location? How many points in the first place are needed for the same reason? As well as other more elemental questions of how to choose the refinements amount to maximise the benefits of adaptive sampling. At the very least, it's clear that benefits exist and investigating whether geometric information can further extend these benefits is an interesting research question with many applications.

Future work - Other areas of PINNs

Due to the relative novelty of PINNs, there are many areas that could be targeted for improvement, or where better understanding of PINNs is needed. At the moment the plan is to focus on adaptive sampling due to the potential of this area, first looking at utilising geometric information to guide adaptive sampling. Long term, goal-based sampling and looking at expanding the sampling algorithm for higher dimensional problems will be looked at. These fit the overall aim to improve PINNs broadly. However, many other avenues of research exist that also fit that overall aim; and as such, it would be worthwhile to keep an eye on other potential areas of work as time goes on.

One area that has garnered attention has been implementing efficient domain-decomposition methods to PINNs. These have been shown to alleviate spectral bias issues, where vanilla PINNs would struggle to learn solutions with multiple-frequencies and only learn high-frequency trends, in papers such as Kharazmi et al. (2021). This is yet another example of bringing over a technique from other fields; domain decomposition is not a concept unique to machine learning, but just like other research in PINNs, it has seen a large interest and fast rate of publications in this field. With multiple papers on domain discretisation methods and frameworks existing (Jagtap and Karniadakis, 2020, Jagtap et al., 2020). More recently, work by Oppenheimer et al. (2023) highlighted another avenue of research on PINNs, adapting the architecture to better solve multi-scale problems, bringing in knowledge from other fields.

What areas of research might still be relevant a year or two from now is impossible to tell - what is certain is that current PINNs technology is still far from optimised, despite the potential of the technology and all the applications it has shown to be effecitve for. This highlights the need to stay on top of new published literature as the PhD progresses.

Chapter 2 : Research Focus

While physics informed neural networks by design circumvent some of the limitations of ML in scientific applications (as described in section 1.1.2), they still face a variety of issues due to their novelty. As a consequence there is ample opportunity to improve their implementation. One specific area within PINNs was identified for potential improvement, regarding the distribution of collocation points. Having identified this focus, the research question became: Can a sophisticated approach for collocation point sampling be developed that brings benefits to PINN performance?

2.1 Research Aim

The overarching aim in this PhD is to immerse myself in the active research field of PINNs and identify an area where I can contribute to the development of PINN technology.

In the niche of collocation point adaptive sampling, the initial aim is extend the benefits that have been seen in literature of using a simple residual-based adaptive approach by utilising additional geometric information available from the PINN network.

In the medium term, I would want to consider developing an analogue to goal-based error estimation from finite elements; where the error of a specific 'goal' output is minimised by adapting the mesh accordingly. Goal-based collocation point sampling could also prove to be beneficial for PINNs, where minimising error in the entire solution field rarely outperforms the accuracy of traditional numerical methods. The aim would be to test this approach in fluid dynamics benchmarks with clear objectives (for example, minimising the error in pressure drop across a blood vessel).

Long term, the aim is to test how the benefits of adaptive sampling that have been observed thus far extend to other fluid dynamics benchmarks and eventually high-dimensional problems, as the ability of PINNs to solve the latter without falling to the CoD is one of its strongest advantages.

2.2 Objectives

To achieve the outlined aims with regards to collocation point research, the below objectives were set. These break down the aims into a set of more easily quantifiable

goals - with detailed information on how these will be achieved in the research plan in Chapter 5.

- Recreate results of Wu et al. (2023) on adaptive sampling using residual information only; and create set of baseline results on accuracy and cost of random and regular distribution methods to compare future advances against.
- Implement algorithm into vanilla PINNs that utilises solution gradient (and possibly Hessian) information to re-position collocation points to achieve an improvement over random methods on first benchmark problem (Burgers' Equation).
- Attempt to merge residual and gradient information to inform point relocation, and expand the methods investigated to a second fluid dynamics benchmark with different characteristics (Current candidate being Korteweg-De Vries Equation).
- Pursue publication in a relevant journal, or presenting at a relevant conference with published conference proceedings (See section 5.3 for details).
- Carry out additional literature review on goal-based error estimation and adaptivity from classical FE to better understand how to carry over those ideas to PINNs.
- Investigate impact of collocation point location on minimising error of specific output solutions - a goal-based adaptive approach.
- Pursue a second publication on goal-based adaptive PINNs.
- Adapt the adaptive algorithm where necessary so it can be used in the solving of higher dimensional problems using PINNs.
- Pursue publication on PINN adaptive sampling for higher dimension applications.

Chapter 3 : Methods

The underlying principles of PINN were explained in section 1.1, and the details on the adaptive algorithms will be explained in turn in chapter 4. The purpose of this chapter is to further expand on the choice of software and pythonic frameworks used in run the PINN networks and code the adaptive sampling algorithm. The technical details of these were explained in section 1.2.3 of the literature review, along with their general advantages and disadvantages. Here I will detail my experience with these and justify the decisions to use the chosen framework.

3.1 Python

Whilst software like MATLAB and R (high-level programming languages with specialised applications for engineering and statistical computing respectively) have packages that extend the tools to include some ML capabilities, the majority of scientific computing research on machine learning is done on python. Python is very popular broadly across research for a variety reasons. One is that it's less technical than other lower-level programming languages like Java and C++, whilst still being very flexible and useful for a variety of purposes, meaning the barrier to entry is quite low.

One advantage that is particularly relevant to research is there is a large number of open-source libraries and frameworks that have already been developed for python by other researchers. These libraries cover a vast array of functionalities, making it quick and easy to implement already optimised functions without needing to do the coding oneself. For machine learning in particular, there are also a few main 'frameworks': these are pre-built sets of tools and libraries that contain a lot of functionality that simplify the process of coding up machine learning algorithms even further. The two main frameworks that are seen in the literature are Tensorflow 2, and Pytorch.

3.2 TensorFlow2, PyTorch

Both TensorFlow2 (TF2) and PyTorch are open source frameworks with similar features but developed independently by different companies. Whilst subtle differences exist in their design philosophies and due to their history, choosing a platform is more greatly a matter of preference than a case of one being better suited than another. Historically,

previous versions of TensorFlow were more complex and as a result less user-friendly. This would've been an important concern to take into account, as it was important for this PhD that I be able to quickly learn how to use these platforms so I could proceed with my work. However, the updated second version, TF2, addressed these issues successfully, eliminating this key difference.

Ultimately, TF2 was more appropriate as it was the framework utilised by in a lot of literature on PINNs, like Raissi (2018), Raissi et al. (2019) and Sun et al. (2020). As looking through the open-source code published in this literature was an important way of learning how the methods were implemented in practice, learning how to modify PINNs through TensorFlow2 rather than PyTorch was deemed the better approach during the initial stages of research.

Additionally, the availability of an external course on TF2 code and modification meant I could learn this framework in a structured way. I could undertake this straight away in the first semester in order to quickly learn how to modify PINNs code myself. The course was split into 3 parts covering TF2 coding at various levels. Completing the first 2 of these before December meant I had a good foundational understanding and some experience coding more traditional fully connected networks and convolutional neural networks (see section 4.2). The remaining course was the most specialised focusing on customisation of individual modules. This however was put on hold in order to complete the internal "Deep Learning" module with the University of Leeds from January to June, which utilised PyTorch.

Thanks to the module, I also gained some experience coding in PyTorch. However, the module focused on theory and the practical aspects didn't include much teaching, meaning I have not learned how to code up NN with pytorch as comprehensively as with TF2. However, whilst the way of implementing things is different with the different frameworks, the way most things work is very similar in concept. The result is even having practice with only TF2, I could still interpret code written in PyTorch even if I could not write code using PyTorch.

3.3 DeepXDE

Proposed in Lu et al. (2021), DeepXDE is a python library that aims to work "as a research tool for solving problems in computational science and engineering". Built with TensorFlow2 as the back-end, it adds commands that can simplify the aspect of setting

up the problem by compartmentalising the code into the different aspects: geometry, boundary and initial conditions definition, PDE and the NN model. However, being built on TF2 it means the library remains customisable in detail whilst having a user-friendly code for the more straightforward aspects (For example, allowing to build in simple geometries via constructive solid geometry).

Being used by Wu et al. (2023), this means in working to validate their results I've already had experience modifying PINNs via DeepXDE. Here, my previous experience thanks to the TF2 course has aided me in understanding the how the code has been built, which will hopefully allow me to more easily modify it with an adapted collocation point resampling algorithm moving on. For all the above reasons, the DeepXDE framework has been used to obtain the most recent simulation results in sections 4.5 and 4.6 and is being used to code the novel re-sampling algorithm.

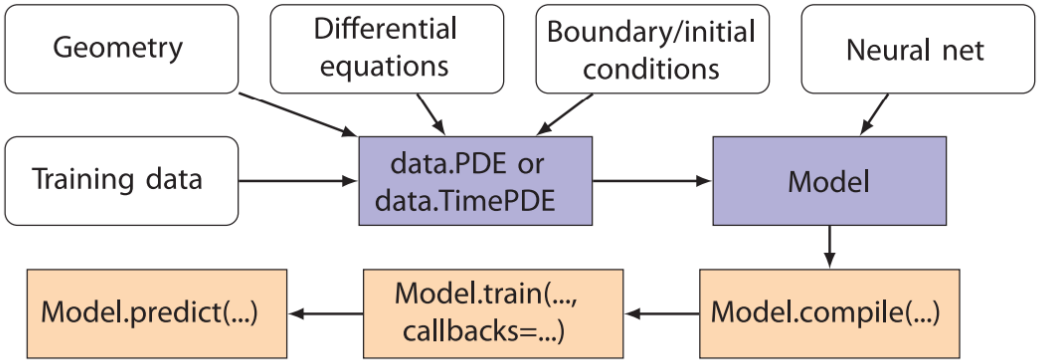


Figure 3.1: From Lu et al. (2021). Flowchart illustrating typical DeepXDE usage. The problem and parameters are defined in the white boxes, blue boxes combine the information and the orange boxes are the solving steps.

Chapter 4 : Progress

In this chapter the main progress done to date will be covered in chronological order. The main purpose at the early stages of the PhD was to catch up to the state of the art of physics informed machine learning in fluid dynamics. To accomplish this, in addition to conducting a literature review , a hands-on approach was taken for learning to code up neural-networks. Section 4.1 covers how the latter was done by building a neural network that could attempt to solve a trivial problem.

A more structured approach to learning was taken by also signing up to an online course, information on this is in section 4.2. In order to make my learning more relevant to fluid dynamics, publicly available PINNs code present in the literature based on Raissi (2018) was looked at[1]. This code was used for an investigation of how different parameters of a PINN affect the solution, mainly as a learning exercise, which is discussed in section 4.3.

Having learned some of the basics of coding ML algorithms, an area of interest was identified regarding the distribution method of collocation points in PINNs. In section 4.4, this question was tackled by investigating how biasing the distribution of points towards areas with high gradients could affect convergence and accuracy.

With evidence that the method of distribution of collocation points could in fact improve the performance of a PINN, the main focus of the PhD became work towards producing a sophisticated adaptive algorithm that could re-distribute or add points throughout the training process. In section 4.5, results from using a pseudo-random distribution method were produced to have a baseline against which to compare moving on.

The adaptive sampling algorithms of Wu et al. (2023) based on residual information were recreated, and results on computational cost and accuracy of these were obtained and are detailed in 4.6. Lastly, the work currently being carried out is then explained in section 4.7, as well as some of the future work.

4.1 Toy Problem - Initial NN Coding Training

At the start of the project, the wide breadth of the topic meant it was hard to tell exactly what kind of problems I would be focusing on further into the PhD. It was

[1](available from https://github.com/okada39/pinn_burgers)

therefore thought optimal to begin by looking at developing some general practice on coding neural networks. To achieve this practice, I first looked at PyTorch, a framework for coding machine learning methods in python. The PyTorch official website contains some tutorials with example problems that allowed me to learn both how to use PyTorch to code a simple neural network, and how these neural networks are trained to solve problems.

The first sample problem involved simply fitting a polynomial to the line $y = sin(x)$. The network was first implemented by using the popular science python library NumPy to perform the same actions a more complex network would carry out. First, the output of the polynomial was calculated with the current weights, and the loss was defined simply as the difference between the output and the ground truth. From this, the gradients of the weights with respect to the loss were obtained and used to update the weights in the direction of quickest minimisation of the loss. Tgus process was repeated, minimising the loss function until the network was trained to predict $sin(x)$. This network was enhanced then by adding PyTorch functionality to the different aspects in turn. First, instead of manually computing the gradient of the loss, PyTorch modules were imported that utilise automatic differentiation to allow for the gradient to be computed even as the network is made more complex with more nodes and/or layers. Tensors (PyTorch's version of NumPy arrays) replaced the arrays, allowing for GPU computations. Lastly, PyTorch was used to define the optimisation method and loss functions to be used. The final result was a fully connected network capable of approximating $y = sin(x)$.

Whilst this was a good starting point to get familiarised with the basic usage of PyTorch, I didn't really have the tools to understand or apply the more complex Physics-Informed-Neural Networks; like for example the work of Raissi et al. (2019). However, by the end of this work I had set up a coding environment and version control for my coding projects via Visual Studio Code and GitHub that would help with data management; and more importantly, had a more clear idea of how to best continue learning the necessary coding skills for the project.

4.2 Python and Tensorflow 2 Training

At the same time, I had been attending some more informal weekly python programming tutorials run for the CDT by Patricia Ternes. In these we tackled some 'program-

ming challenges' that helped us learn or revise useful techniques for general scientific computing. These ranged from tips to improve readability and efficient computation (with techniques like list comprehension) to practice using scientifc python libraries like NumPy for data processing and implementing randomness. But as mentioned at the end of the previous section, I had also found a more appropriate way of learning the essentials of machine learning for my project. I observed a lot of code on PINNs specifically used TensorFlow2, and I was recommended the coursera specialisation on TensorFlow2 for Deep Learning. Prerequisite knowledge for this included "knowledge of general machine learning concepts (such as overfitting/underfitting, supervised learning tasks, validation, regularisation and model selection), and a working knowledge of the field of deep learning, including typical model architectures (MLP/feedforward and convolutional neural networks), activation functions, output layers, and optimisation". A foundational course offered by Google "Machine Learning Crash Course with TensorFlow APIs" was completed in order to catch up to the prerequisite knowledge required.

Then, the first of 3 courses forming the Tensorflow2 for Deep Learning specialisation was undertaken, "an introduction to Tensorflow2" with the following main learning objective:

- "To learn a complete end-to-end workflow for developing deep learning models with Tensorflow, from building, training, evaluating and predicting with models using the Sequential API, validating your models and including regularisation, implementing callbacks, and saving and loading models."

This course involved learning through a mix of pre-recorded instructional videos, tests and coding tutorials; with a "Capstone Project" assignment due at the end. This project involved developing both a feedforward MLP network and a CNN model for classifying images from the Street View House Numbers (SVHN) dataset. These two were coded and used to accurately classify unseen images from the dataset - see figure 4.1.

The second course was also begun - "Customising Tensorflow2 modules", which looked at expanding knowledge by teaching how to use lower level APIs, which will be necessary for creating and adjusting custom PINNs networks. The official learning objective is described as:

- To "deepen your knowledge and skills with TensorFlow, in order to develop fully

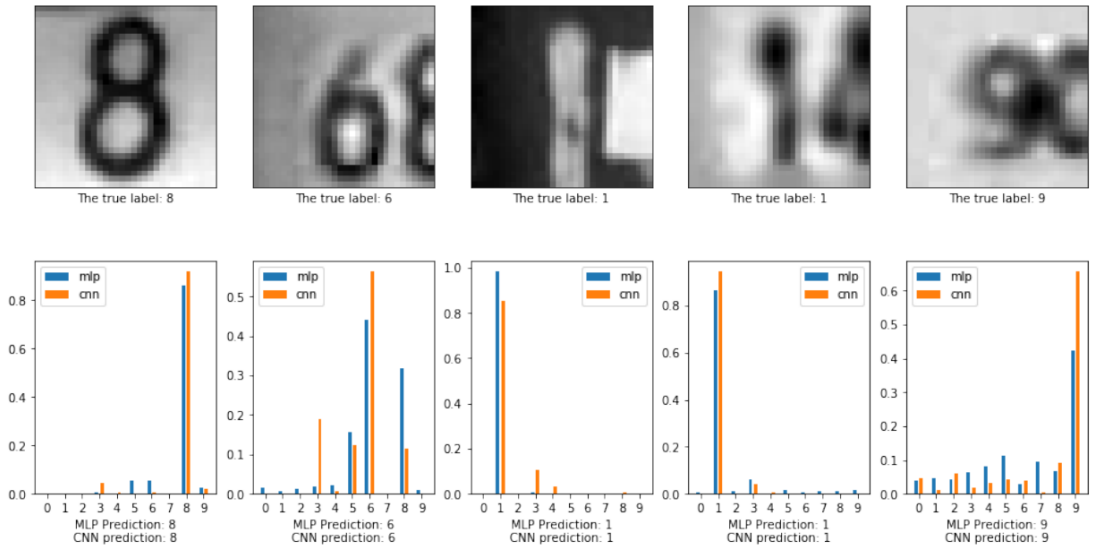


Figure 4.1: Above: Image input to classifiers; Below, bar graphs showing the perceived likelihood of the image matching each number, with the networks' guess below.

customised deep learning models and workflows for any application. To use lower level APIs in TensorFlow to develop complex model architectures, fully customised layers, and a flexible data workflow. To expand your knowledge of the TensorFlow APIs to include sequence models."

4.3 PINNs for Burger's Equation - Hyperparameter Investigation

To get hands on experience with PINNs, github code solving Burger's equation based on Raissi et al. (2019)'s work was pulled and modified. The idea was to play with hyperparameters of the network to get hands on experience of effects of different parameters of network and a feel of how big these effects were relative to each other. To undertake a proper quantification of effectiveness of network, first some ground truth data had to be created to compare the results of the network to. As burger's problem is relatively simple, this was done via a Finite Difference Method (FDM) approximation.

4.3.1 Generating Ground Truth through FDM

The objective was to have an accurate, fine grid to compare the NN solution to. To compute the answer to Burger's equation, it was re-written in Ordinary Differential Equation (ODE) form using a central difference approximation. An ODE integrator python library from SciPy was then used by looping through space and integrating the ODE in time to generate a square matrix of $u(x,t)$ at every point in space and time in the domain ($-1 < x < 1$ and $0 < t < 1$). To quantify the discretisation error of the FDM grid (whether it was fine enough), carried out a convergence study by looking at pointwise discrepancy (Quantified by the L_2 error, see eq 4.1) between grids increasing in resolution (see figures 4.3a, 4.3b, note 6400 means L_2 error between 3200 and 6400 grid). These were compared by restricting the higher resolution grid. Interpolating the lower resolution grids into higher resolution was also considered but would've been more time-consuming and the grid was determined to be adequately fine enough at 6400 points from the mesh convergence study. As the solution data was obtained through the study there was no reason not use the high resolution grid obtained.

$$L_2 = \frac{1}{n}\sum_{i}^{n}(u_{i,c} - u_i)^2 \tag{4.1}$$

Where:

n = Total number of points,

$u_{i,c}$ = Velocity at point i in restricted grid,

u_i = Velocity at point i in lower resolution grid,

For comparing the performance of the PINN, it was necessary to look at some quantitative form of error as visual inspection was not good unless results are very wrong - most network parameters capture the burger's equation shock general shape but it's hard to spot small deviations or errors in magnitude without taking a detailed look at slices throughout the domain. It is also very hard to compare the more complex / accurate networks as the differences between their solutions and the ground truth solution become too subtle (See figure 4.2 for an example visual solution). Therefore, the main way of comparing the accuracy of networks was to again look at the pointwise error. For the network, no restriction was necessary, as once trained the network could be used to predict the solution onto a grid of any size. This allowed to simply subtract the matrix of ground truth solutions from the NN prediction and take the mean error over the domain.

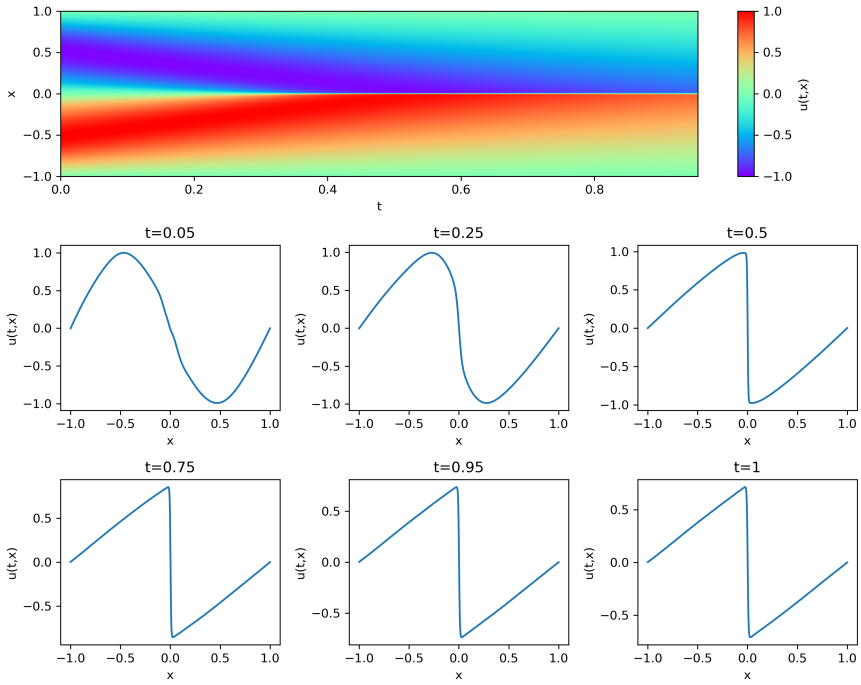
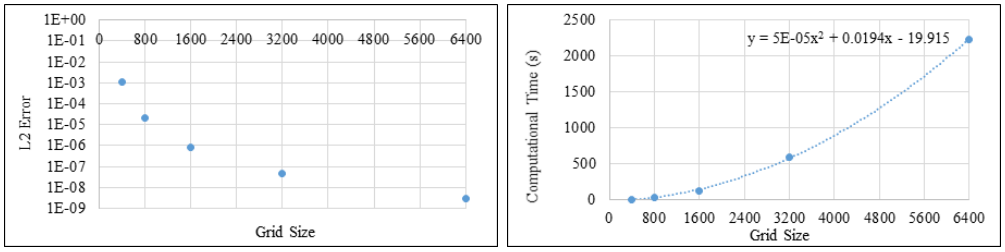


Figure 4.2: Burger's Equation solution using 8000 training points and a 3 layer network shape $32 \times 16 \times 32$. Above: Contour plot of u(t,x); below, 2D slices of the domain at different t.

One of the investigations looked at was how changing the viscosity of the problem would affect the network's ability. For this case, a separate solution had to be computed with the different viscosity. Instead of repeating the grid size sensitivity study, the same grid size of 6400x6400 was deemed sufficient and used. For lower viscosity ($\nu = \frac{1}{400\pi}$), run time increased a substantial amount (from 37 minutes to 54 minutes for the finest case, a 46% increase). Increasing to a much higher viscosity ($\nu = \frac{1}{\pi}$) led to a case too viscous for a shock to form, so the case was discarded.

4.3.2 Hyperparameter Investigation

Three main hyperparameters were investigated: Training point number, shape of neural network (mainly node number) and convergence criteria (factr). Then viscosity was also changed, changing the equation to be solved. The training point investigation was then repeated for the new viscosity, comparing against the other ground truth solution. The max number of steps was limited as a compromise due to time; which is



(a) FDM Solution: L_2 Error vs Grid Size, noting grid size indicates length of square array ($N \times N$). Note log scale on y-axis.

(b) FDM Solution: Computational time taken vs grid size. Dotted line is quadratic best fit approximation (function shown).

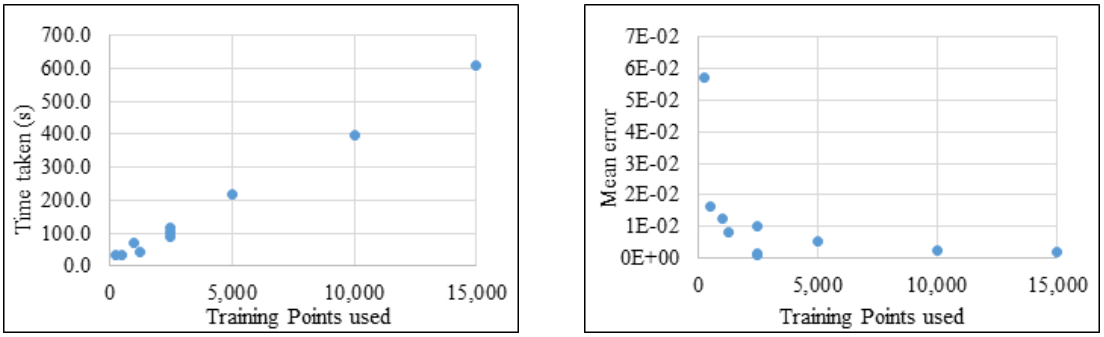
Figure 4.3: Grid independence study of FDM solution.

worth noting as it could have limited the maximum possible accuracy for training with slower convergence. Regarding the PINNs default optimisation methods, the default L-BFGS-B algorithm was kept as the optimiser for the network; and the default 'tanh' activation function and learning rate were also left unmodified. With these settings, it was possible to look at parameter trade studies and their effects on mean error and time taken.

Training points: At this stage didn't have a way of automating runs to efficiently do re-runs of experiments. There was an anomaly run 3 times at $N = 2500$ where, despite having less training points than other runs, was much more accurate than more dense, slower simulations. These results can be seen in figure 4.4. Generally, increasing number of training points linearly increased time taken for network to train. Increasing the points decreased the error as expected, with the increase in accuracy requiring more training points.

Network shape: Default shape was 3 dense layers of 32, 16 and 32 nodes respectively. This investigation was carried out by halving the number of nodes and looking at the effect on the quality of the solution. Raissi et al. (2019) paper mentions they utilised 6 layers of 20 nodes, so this was also looked at. In order to quantify this, the number of parameters (total connections between nodes) was plotted against the mean error. For reference, the 6x20 case had 3021 parameters compared to the default case's 1201 parameters. Results can be seen in figure 4.5.

For figure 4.5b, time taken seems to approach a finite number with increasing parameter number, but might be that the higher layer '20s' case is more efficient computationally, giving it the wrong shape. For a relatively small increase to time taken, the 6



(a) Training Points vs error

(b) Training points vs time

Figure 4.4: Training point number investigation, for viscosity of ($\nu = \frac{0.01}{400\pi}$) and the default NN shape ($32 \times 16 \times 32$).

layer case has lower error than the other cases (note the log scale means this is lower than may appear).

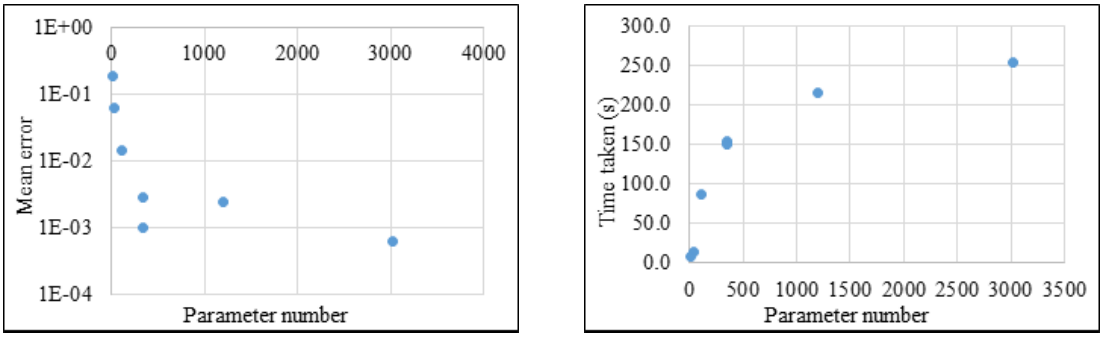
NN Convergence Criteria: Defined as "factr" within the code, it was a unique variable to this implementation of PINN that I was trying to understand. It seems to dictate at what residual the solutions would be considered to be converged. The results of adjusting this parameter can be seen in figure 4.6.

The effect of factr was looked at for both a high and a low number of training points, to better understand when it plays a factor and whether its dependent on others. Looks like a linear relationship in time, but its harder to tell for the mean error. However, it's clear that with more points gradient of time taken higher, as a result sacrificing factr yields much greater time saves. At the same time, more training points means error still lower for modest factr. Moving on, the 'modest' factr value of 1E7 was used, as it produced suitable accuracy for the hyperparameter studies without compromising the computational cost too much.

Viscosity: To look at viscosity, again the default network shape was used. The mean error and time were compared between cases as the number of training points used was varied.

It was observed that the PINN network took a similar amount of time to compute answers for both cases, but had a lower accuracy for the lower viscosity case ($\nu = \frac{0.01}{400\pi}$).

Overall the network seemed quite robust and didn't fail to produce a solution unless it had very very few points - even the simplest NN shapes with 3 layers of 8, 4, and 8 neurons were able to predict the burger's equation shockwave. By calculating the mean



(a) Number of parameters vs error

(b) Number of parameters vs time

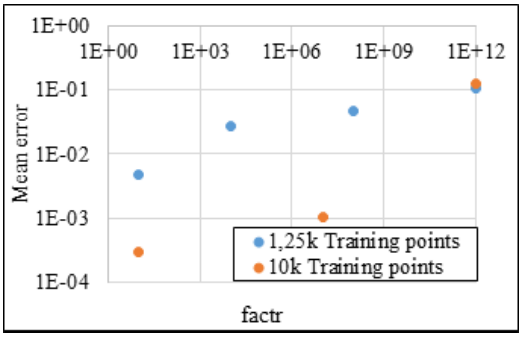
Figure 4.5: Network shape investigation results. Shapes followed 2N:N:2N shape with the exception of the 3021 point which had 6 layers of 20 nodes each. Note log scale on error graph.

error across the domain, it was possible to compare how the accuracy was affected by the various hyperparameters. Plotting the results of this, could easily see the trade-offs between computational cost and accuracy for the different parameters and confirm the expected behaviour: generally, more complex and bigger networks provide increase to accuracy at an increased computational cost - with increases to accuracy requiring exponentially more resources below a certain threshold.

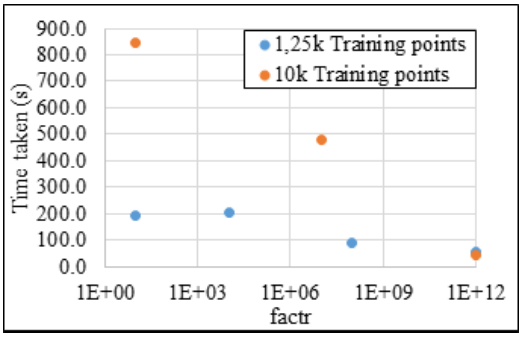
Overall, these sets of investigations were really beneficial for getting a grasp on the basic concepts learned at the same time through first coursera course, and how they apply to PINNs specifically. Having to modify the code directly also meant I learned exactly how these hyperparameters were implemented in python and how the machine learning method was coded in practice. This would be incredibly useful later on for knowing how to modify the code in order to investigate biasing/adaption.

4.4 PINNs for Burger's Equation - Bias Investigation

Having a better idea of how PINN network worked, I wanted to look at changing the fundamental method employed by this PINN in ways that might bring about benefit. Namely, to investigate whether the way collocation points are distributed influenced the cost and accuracy of the solution. The first step for this was to look at how biasing the points location would affect the results. A very simple bias towards the center of the domain was tested, and results seemed to indicate a potential benefit could be

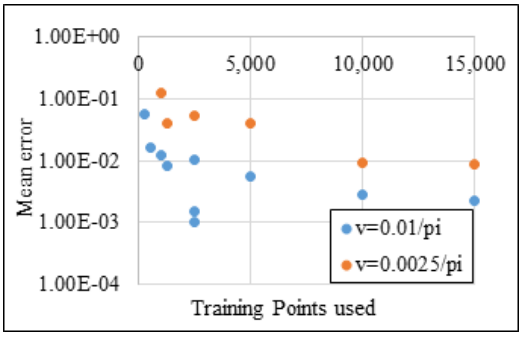


(a) factr vs error

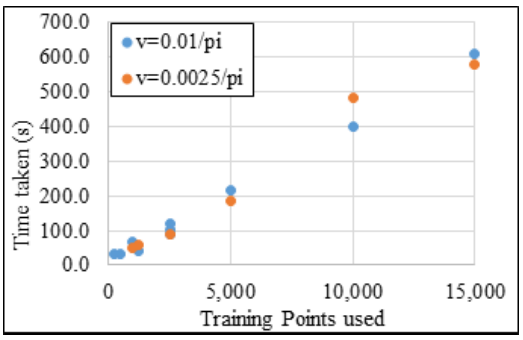


(b) factr vs time

Figure 4.6: Convergence criteria (factr) investigation (lower number = higher accuracy). 1E1 = highest accuracy, 1E7 = modest, 1E12 = low accuracy



(a) Error vs Number of Points



(b) Time vs Number of Points

Figure 4.7: Results for different viscosity cases

obtained from this. A more complex shape was given to the bias to see if the effect could be recreated. Lastly, it was noted that both the sampling of points and the ML methods have a random nature, and I wanted to quantify the effect of the randomness of the sampling of points on the variance between runs. To do this, I looked at the variance in error through a series of bias studies.

4.4.1 Simple Biasing

For the first look at biasing, the $32 \times 32 \times 32$ NN architecture was chosen. As the number of points used was predicted to also influence how much bias would have an impact, various numbers of points were tested too. A preliminary simple approach was carried out, biasing points towards the middle of the domain (which can be seen in

figure 4.8) to see if it had an effect. The biasing was done by dividing the domain in 3 parts, and adjusting the number of points distributed within each area according to the desired bias. This random sampling was relatively trivial; a random number generator was used to generate a $[2 \times N]$ vector of floats between 0 and 1, where N is the number of points. By multiplying this vector by the domain range, an (x, t) vector of random coordinates is produced within that range. When biasing into 3 areas, the number of points in each area N is adjusted as is the range.

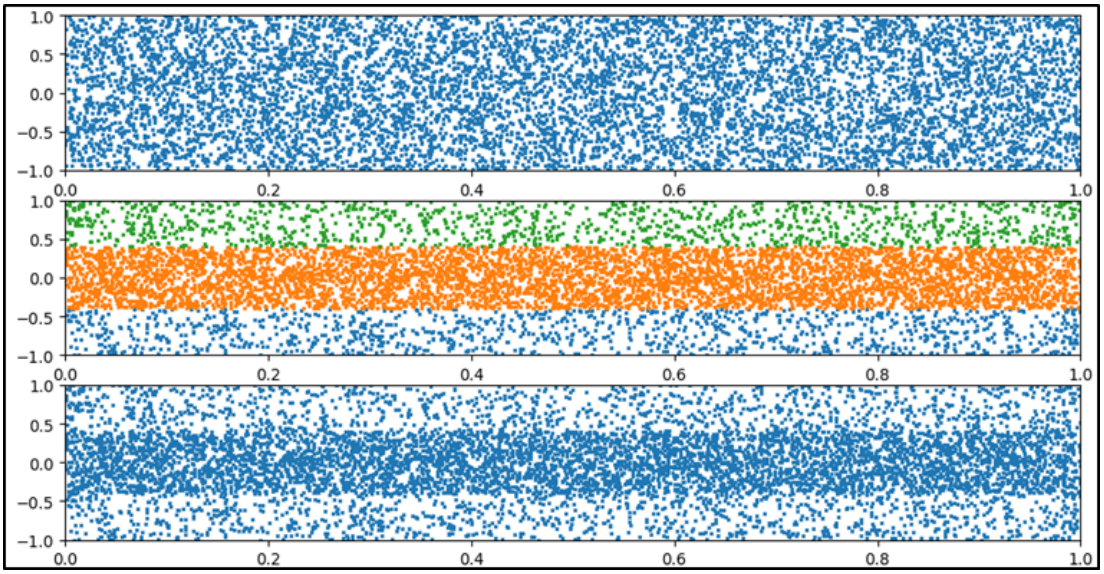


Figure 4.8: Random distribution of points throughout the domain at the top versus the distribution using a simple bias towards the center.

To quantify this bias, it was defined as :

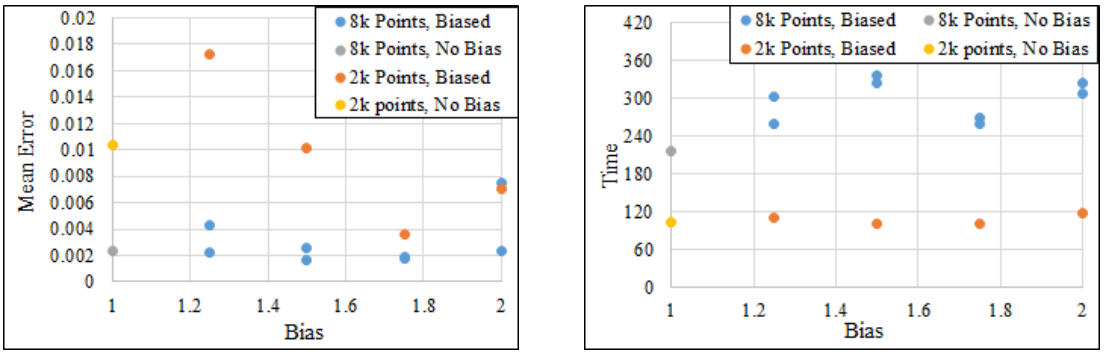
$$\text{Bias ratio} = \frac{N_{Bias}/N_{Total}}{A_{Bias}/A_{Total}}, \tag{4.2}$$

Where:

$$N_{Bias} = \text{Number of points in biased section},$$
$$N_{Total} = \text{Total number of points},$$
$$A_{bias} = \text{Area of biased section},$$
$$A_{Total} = \text{Total Area}$$

Following this, if 40% of the points were in 40% of the area, bias = 1 = no bias. If 80% of the points were in the same area, the bias = 2.

To test the effect of this, repeated runs were made for every bias. For each run, independently of the number of points used, the network was used to predict the velocity field on a grid the size of the FDM solution grid. This way the difference between the predicted solution and the ground truth FDM solution could be calculated at every point. The average error in u was then taken. This average error and the time taken for the computations was then stored.



(a) Mean error vs Bias.

(b) Time vs Bias.

Figure 4.9: Results of the simple biasing.

Observing the results in figure 4.9, it can be quickly observed that increasing the number of points significantly increases the accuracy of the solutions, as observed in the hyperparameter investigations, despite the different distribution of collocation points. A small improvement can be seen with regards to accuracy for the 8k points case, but more significantly, for the 2k case a more significant improvement can be seen when the bias is between 1.5 and 2. These preliminary results were interesting enough that it was thought worth repeating with a less simple biasing method, for which the number of runs could be repeated more often to ensure there results were statistically significant.

4.4.2 Refined Biasing

Next step was refining the bias area. This was less trivial as it was not longer possible to populate the non-rectangular bias areas with the methods described earlier. The bias areas were split into triangles and then a slightly more complex function was used to generate points in the triangle given the vertices of this. Functionally though, the biasing worked in the same manner; with the bias being described by assigning the appropriate number of points in each bias area and distributing them randomly. The

more complex shape can be seen in figure 4.10. Additionally, at this point the PINNs code was modified so it would save the prediction results and allow for looping of different parameters such as number of training points. This allowed repeated runs to be run automatically for different investigations, increasing the reliability of the results. The key results of these bias investigations are shown in figure 4.11.

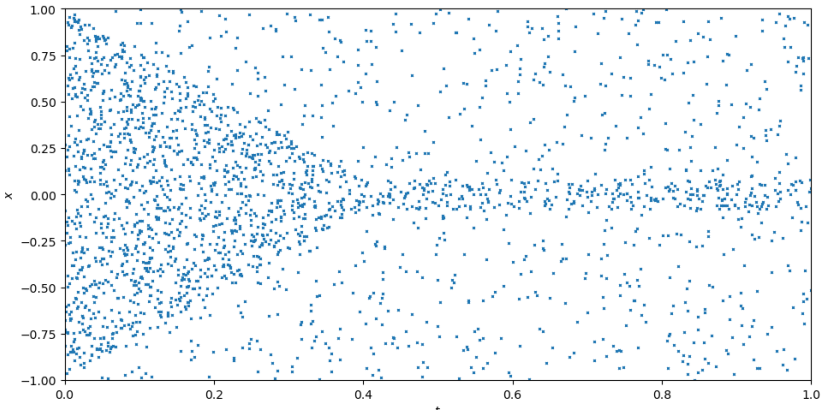
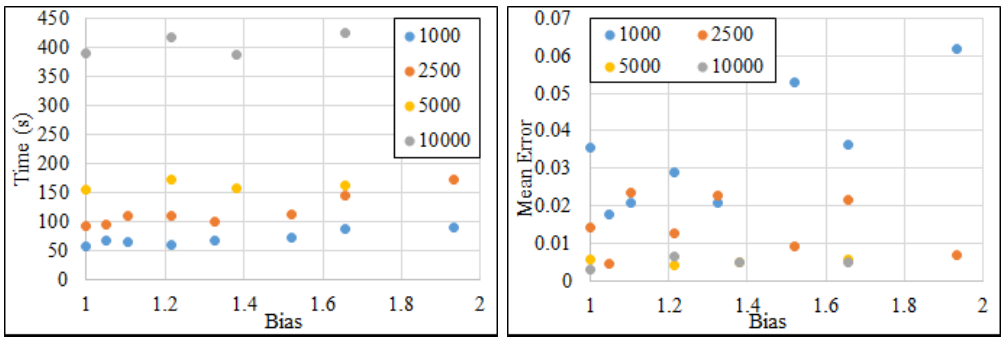


Figure 4.10: Collocation points distribution into more complex bias shape



(a) Time vs Bias

(b) Error vs Bias

Figure 4.11: Line graph showing results of biasing as shown in figure 4.10, with a central width in x of 0.3 and trapezium height of 0.5 in t.

For this bias shape it is harder to observe clear improvements to the accuracy from the introduction of biasing. The usual expected patterns observed in section 4.3 can still be seen: increasing number of points increases accuracy up to a point at the cost of increasing computational time. Introducing bias didn't seem to increase computational time very much, trivially compared to the effect of changing number of training points.

The effect of biasing at high number of training points was negligible as expected - as with enough points even non-refined areas will have more points that non-biased cases with less points. For the cases with lower number of training points (1000 and 2500), some small improvements were observed but not in a consistent way - the 1000 case saw small improvements with very small biasing between 1 and 1.4, whereas for the 2500 case improvements to accuracy seemed to occur randomly with bias. It is worth nothing that choosing the area of refinement was done by manually observing where the highest velocity gradients existed in the solution. What is most likely is that only points on the shock, where the gradients are highest, help increase the accuracy of the network. As a result, expanding the refinement area to capture the time before the shock only diluted the benefits of refining near the shock. Looking at Lu et al. (2021), where they identify in their figure 8 that the points with highest residuals for Burgers' Equation lie exactly on the shock, it seems likely that only refinement on the shock would provide benefit.

4.4.3 Variance

Before continuing on to further work, one observed issue was the variance between runs being so big it made it hard to spot the patterns (See figure 4.12). The focus in December became examining the impact of the random distribution of points compared to the random nature of the ML method. How much the error varied when keeping the points in the same place (See figure 4.13) was examined. From this particular example, it seemed like with constant settings, the random distribution of the points seems to increase the variation quite significantly, with individual runs seeming to be distributed much closer to the mean.

To do a more thorough investigation of this, a slightly more sophisticated way of checking this was planned by looking at the variance. The variance was computed by evaluating the standard deviation of the errors, and how this changed as the number of collocation points was increased was observed for different biases. This was done for 2 cases: in the first case, the collocation points were sampled randomly according to the bias specified every re-run; in the second case, the collocation points were sampled randomly once and then the same distribution was re-used by setting a seed for the distribution of points. The results of this can be seen in figure 4.14.

Expected fixed points to reduce variation (solid line below dotted line) and variance

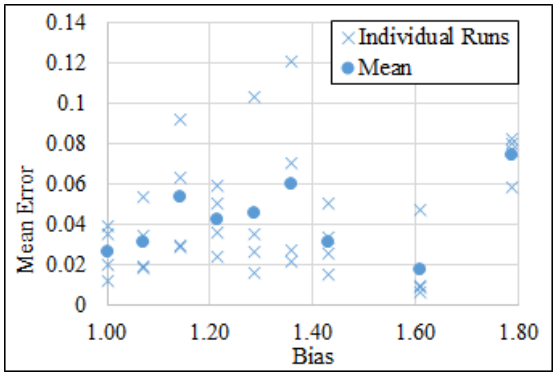


Figure 4.12: Error vs bias runs showing the spread of individual run errors. For the 500 Points case seen in figure 4.11

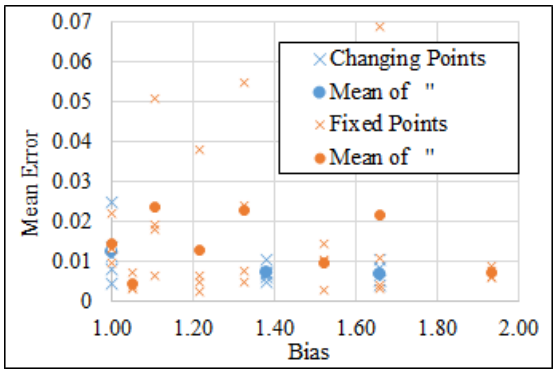
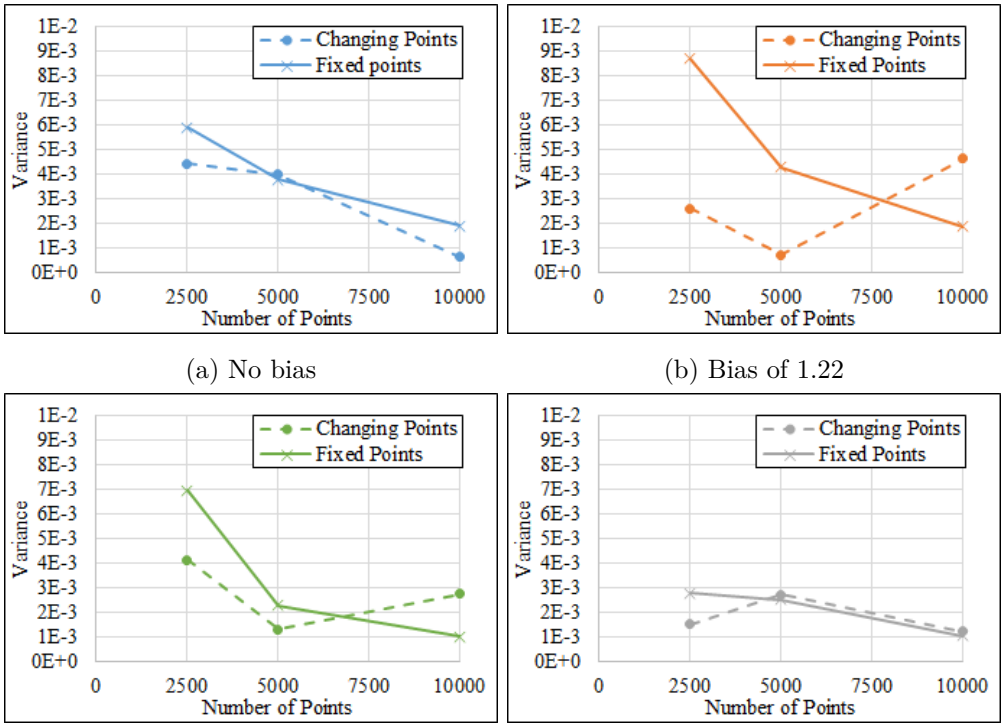


Figure 4.13: Graph showing the comparison between fixing the points and re-sampling every run, for the 2500 points seen in figure 4.11

to reduce as number of points used increase. The first expectation was generally observed to be more true as the number of points was increased, but was not true across all cases. Possibly, the variance was not calculated from enough points. Additionally, at the outset it was assumed that the variation due to the method would be consistent, and that therefore removing the randomness of the points sampling between runs would allow it to be observed. However, it is possible that if the distribution of points that was fixed happened to be relatively bad this could impact not just the accuracy of the network but also the variation produced by the network between runs. This should be less of an issue as the number of points is increased as the effect of distribution and bias decreases, as observed previously.

Overall, high variance due to the use of random sampling every run was observed.



(a) No bias

(b) Bias of 1.22

(c) Bias of 1.38

(d) Bias of 1.66

Figure 4.14: Effect of re-sampling points every run through variance vs training points for increasing biases.

This called attention to the need for using uniform methods such as Latin Hypercube Sampling (LHS) instead of simply random sampling, even before looking at adaptive sampling. This would help make it easier to spot patterns by reducing the variance between runs. Thanks to the fact a benchmark case was being analysed a ground truth solution is available so it is possible to see that some of the solutions output by PINN are good even with random sampling. However, this would not be the case for the vast majority of applications. Because of this its hard to justify the use of completely random sampling as multiple re-runs would be needed due to the variation in accuracy from case to case. Additionally, LHS and other similar sampling methods would potentially be easy to implement and not add significantly to the computational cost, as there is a high likelihood that example implementations may already exist within the Python libraries used.

4.5 Baseline Results - Pseudo-Random sampling

Following the successful results of section 4.4 it was known that appropriate distribution of collocation points near areas of high gradients could increase accuracy. In the case of the investigations on Burger's equation, the position of refinement was guided by a priori knowledge of the position of the shock. To be applicable to problems with unknown solution, a collocation point distribution algorithm would be needed that could harness information garnered throughout the solution to choose how to best distribute points - an adaptive algorithm.

In order to measure the potential benefit of such an algorithm, it would be necessary to first have a baseline way of distributing points that the adaptive approach could be compared to. The standard so far was a random algorithm, which is not an optimal distribution method as there is a risk of points being positioned redundantly or not evenly throughout the domain. As seen in the previous section 4.4.3, the results of this is a high variance to the accuracy associated with random distributions. Solutions to this via pseudo-random formulations have been explored in other fields in scientific computing. For this reason, the next logical step was to consider utilising a pseudo-random approach that would certainly improve the distribution of collocation points and therefore be a better baseline against which to compare future adaptive work.

About this time, the work of Wu et al. (2023) was published proposing two adaptive sampling approaches. More relevant to this section, they also compared the effectiveness of 4 pseudo-random methods with random and uniform sampling. Out of these, the Hammersley sequence (Hammersley and Handscomb (1964)) had the better accuracy over most of the problems looked at by the authors. For this reason, it was decided to use the Hammersley distribution algorithm as the baseline against which to compare future algorithms.

To create a Hammersley sequence, a function was available through the DeepXDE python library, based on Wong et al. (1997). Therefore, all that was required was to insert the domain size and number of points to be generated to the function. For this baseline simulation, the PINN was then trained utilising the points from the Hammersley sequence from start to convergence.

To account for the natural variation of ML algorithms, multiple re-runs were done with for each number of training points investigated. Unlike the literature, the time cost in minutes was also recorded. It must be noted this is not an accurate measure of

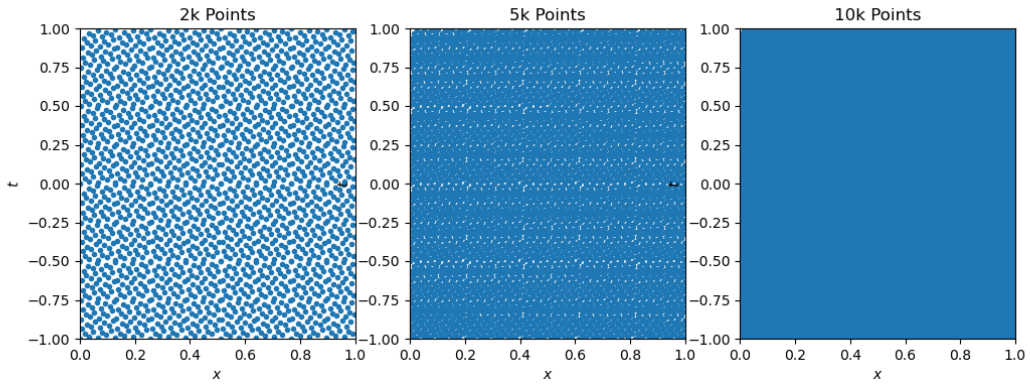


Figure 4.15: Collocation point distribution following Hammersley method for 3 varying numbers of points.

computational cost, but for now merely serves as a reference to compare the relative cost.

Visually the results are shown in figures 4.16 and 4.17. The error for 2000 points is the main result that Wu et al. (2023) explicitly states, as this is the number of points used to compare all the approaches it tests. At 2000 points, the percentage difference between average L^2 error computed is $\pm 2.98\%$, but the overall error curve can be seen and the other points roughly fit.

Close agreement was observed with Wu et al. (2023)'s results. The full curve cannot be recreated due to resource limitations, but simulations at discrete points throughout the range produced results of similar order of magnitude. Additionally, the decrease in standard deviation of runs was also noticed as the number of residual points was increased. Looking at figure 4.17, the time taken was observed to increase proportionally to the number of points as expected.

4.6 Adaptive Sampling - Validation

In Wu et al. (2023), the two proposed adaptive approaches utilise residual information to dictate the distribution of collocation points near areas of importance. Utilising residual information was the logical first step for an adaptive approach as this information is obtained naturally during training. A sensible next step would be to explore utilising other markers to inform where to place collocation points. From the preliminary bias

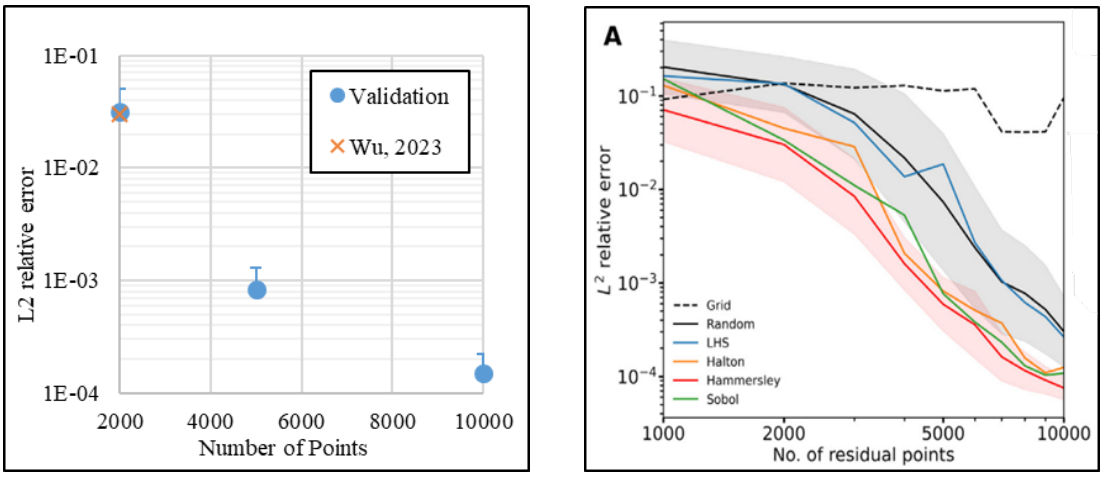


Figure 4.16: Error vs number of training points. Right: Same result, Wu et al. (2023)

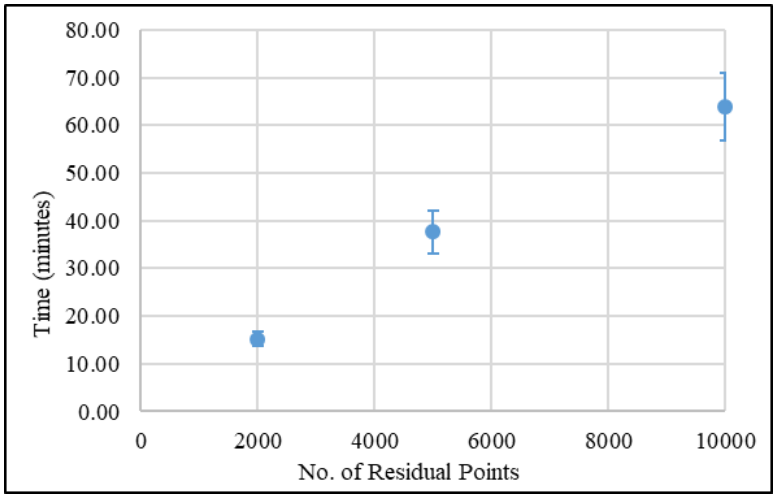


Figure 4.17: Time vs number of training points, utilising Hammersley regular sampling

study on the Burgers equation, the benefit was observed to come from placing points near the shock, where the velocity gradient of the flow was highest. The question to investigate was thus how good geometric information (such as the first order gradient of u and higher derivatives of it) was as a guide of where to best place points.

Before undertaking that, however, I first wanted to attempt recreate the two adaptive approaches proposed by Wu et al. (2023). By doing this, I could gather information on time taken by those compared to the baseline results obtained in the previous section.

To compute the location of new points, both sampling methods depend on the following probability density function:

$$p(\mathbf{x}) \propto \frac{\varepsilon^k(\mathbf{x})}{\mathbb{E}[\varepsilon^k(\mathbf{x})]} + c, \tag{4.3}$$

where:

$$\begin{aligned} p(x) &= \text{probability function}, \\ \varepsilon &= \text{PDE residual}, \\ k, c &= \text{constant hyperparameters} \geqslant 0 \end{aligned}$$

The hyper-parameters k and c control how $p(\mathbf{x})$ distributes the points; balancing how much the points are uniformly distributed throughout the domain vs distributed at the areas of highest residuals. The first adaptive method proposed was Residual-based Adaptive Distribution (RAD). This was run with the same settings as in the paper. Following the approach from Wu et al. (2023) described in algorithm 1.

Algorithm 1 RAD

1: Sample initial residual points using a random or uniform sampling method;
2: Begin training for a fixed number of steps;
3: **repeat**
4: Replace points with new set of points obtained using Eq. 4.3;
5: Train for a fixed number of steps;
6: **until** Pre-determined number of re-sampling loops is reached;

The first set of 1000 points were randomly distributed throughout the domain, and the training was started with 15 000 steps using the Adam optimiser. Then, after a maximum of 1000 steps with the L-BFGS optimiser the re-sampling loop began. Every

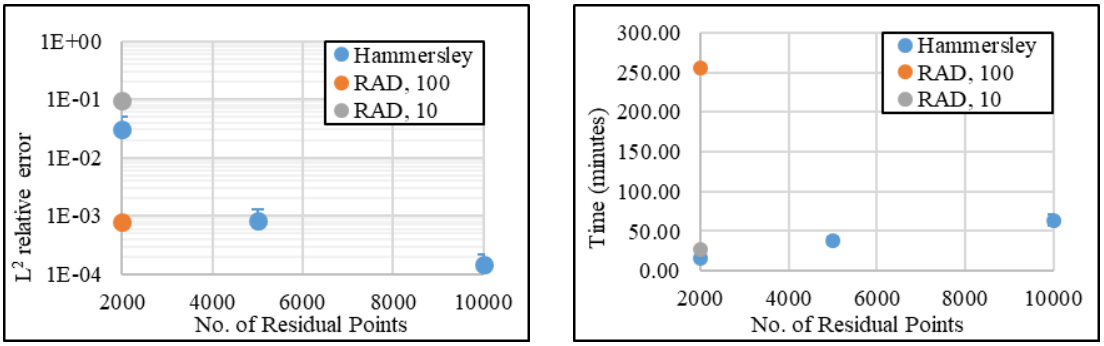


Figure 4.18: Error and Time of RAD approach, with 10 and 100 re-samples

loop, points were added such that after 100 loops there would be 2000 collocation points. Every loops the Adam and L-BFGS optimisers were used in turn to ensure the loss reduced. For RAD, both in the original paper and my investigations the hyper-parameters were chosen such that some of the points are uniformly distributed.

Some studies were made adjusting certain parameters to see if cost could be reduced. The number of initial steps before re-sampling begun was adjusted, as well as the steps of the different optimisers between re-sampling. The main impact seen was from changing the number of re-sampling periods, as each period added an additional 2000 solver steps. However, none of the changes above found success reducing computational cost whilst achieving similar accuracy.

The error obtained utilising the RAD algorithm with 2000 final collocation points was pf 0.078% vs the 0.02% obtained by Wu et al. (2023), a percentage difference of 118.5%. This was when keeping the same settings, which meant doing 100 rounds of re-sampling (with an associated big increase in steps and therefore cost). The error went up to 9.347% with only 10 rounds of re-sampling, however this still saw an increase to computational cost without an increase to accuracy relative to the baseline using Hammersley. These results can be seen in figure 4.18.

Whilst utilising RAD increased the accuracy for a fixed number of points, at 100 re-samples the simulation ran for ∼200 000 steps compared to ∼30 000 for the baseline, massively increasing the cost. The actual re-sampling takes a fraction of the computational cost related to the solver steps. Recalling the results from 4.4, it was observed that fixed random sampling in areas of importance can increase accuracy without affecting cost, so therefore it is reasonable to believe that this increase to accuracy could be arrived at without having to increase the cost as substantially as the RAD method

does, by extending the number of steps the solution can run for.

In addition, the second adaptive method proposed (Residual-based Adaptive Resampling with Distribution, RAR-D), seen below in algorithm 2, could be more efficient at arriving at a solution, and so it is important to also recreate the algorithm and analyse its cost. There are many other questions raised by this work so far: how much training is necessary before the solution information can be trusted to decide where to re-sample collocation points? How many re-sampling loops are needed? Is the PDF used the most optimal across cases, and what hyperparameter values should be used? This highlights the many ways in which this work could be taken and are questions to keep in mind as I continue working on better understanding PINNs.

Algorithm 2 RAR-D

1: Sample initial residual points using a random or uniform sampling method;
2: Begin training for a fixed number of steps;
3: **repeat**
4: Obtain new points using Eq. 4.3;
5: Add new points to current set of points;
6: Train for a fixed number of steps;
7: **until** Pre-determined number of sampling loops is reached

4.7 Ongoing Work

Currently, work was being done on debugging the RAR-D algorithm, attempting to recreate the error accuracy of Wu et al. (2023) and making a note of the associated cost. RAR-D works very similarly to RAD, but with the key difference that collocation points aren't removed after each loop. Instead, at every sampling loop, new points are added, again following the probability distribution function. However, to account for the fact that the initial random sampling covers the broader area, the hyper-parameters of the PDF (eq. 4.3) are adjusted so that new points are added only in areas with high residuals. This different approach has the potential of being more computationally efficient as it allows for faster training at the beginning thanks to there being fewer initial points. In addition, only adding points after the network has trained for some time makes it is more likely that areas with high residuals are good candidates for collocation pints

About the novel adaptive approach, the initial step is to obtain first derivatives of the solution, which could potentially be queried via functions built in to the DeepXDE framework. The next step would be to modify the current code to utilise only that information to correctly move points towards areas of interest, without aid of residuals. The differences to accuracy would then have to be compared to previous simulations

Potentially, one could then look to combine information from the different sources, residuals as well as solution information. There are many nuances that could be experimented with, another is looking at utilising higher order solution information. This could be done by examining the hessian of the solution at different points, which should be possible thanks to functions within the DeepXDE library.

After those initial steps are done, what follows is more prospective. If the novel adaptive approach is successful, it would be necessary to work on testing its efficacy on different benchmarks. After that, the next objectives described in section 2 would be pursued. The first of these would involve studying goal-based error estimation, and how that was used in FE to direct grid adaptivity approaches. Creating an analogue to PINNs point distribution could also bring a benefit for specific applications in a similar way, but for applications utilising PINN methods.

Another further question that emerges is what is the associated cost of running simulations for more steps versus simply adding more training points? Wu et al. (2023) only explores low dimensional problems, and certain aspects of the current implementation will make it inefficient for high-dimensional problems. One of the main advantages of PINNs are their potential to solve high-dimensional problems that are impossible for traditional numerical methods. For this reason it is important that any adaptive sampling method is scalable to high dimensions so the benefits aren't exclusive to highly specific problems. This would be the last item of work to investigate on the research plan.

Chapter 5 : Research Plan

In this chapter, the current plan for the PhD project from now until completion is presented. A detailed plan from the transfer report deadline until the end of the calendar year 2023 is presented first, followed by a less detailed plan for the remainder of the PhD. The plan of publications and conferences is detailed in section 5.3, while training, data management and expenditure plan are detailed in Appendix A. All of this is summarised in the Gantt chart in section 5.4.

5.1 Until End of Calendar Year

The main focus in July is to work on the first objective, continuing implementation of a novel method of adaptive collocation point sampling that's not only based on residual information, but utilises gradient and other geometric information to decide where to best distribute collocation points. The current code will have to be extended significantly to do this. A way to retrieve solution gradient information from the neural network needs to be written, as well as the possible mechanism by which this information informs the point re-distribution algorithm.

In order to appropriately assess whether this solution gradient information is being useful, the first step will be testing an implementation where only the gradient information is used (without loss residuals) to distribute the points in the Burger's equation case. Success will be determined by whether an improvement over the baseline (defined in section 4.5) is seen. If successful, the next step will be coming up with a way to combine the information from residual points and gradients, as well as possibly higher derivatives of the solution information. An empirical investigation would need to be made on how to weight the two sources of information in the point redistribution to maximise accuracy. The expected outcome of this process would be a reduced error for the solution of Burgers' equation, using the same number of points and network parameters as Wu et al. (2023). A curve of error against number of points (See figure 5.1) could also be produced for a more in-depth comparison.

Now that the module on Deep Learning has been completed, a secondary thing to work on after the transfer report is finishing the training I'd begun on TensorFlow 2 that I had begun last year. The remaining course is the most in-depth, covering details on low-level customisation which could prove helpful for when I have to create custom

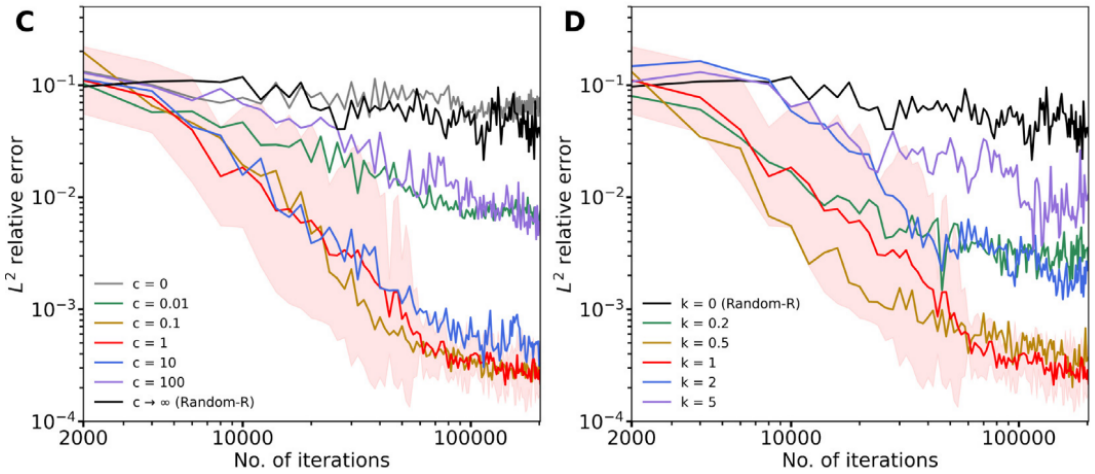


Figure 5.1: L^2 relative errors vs Residual Points for RAD method, with k=1 and c=1 respectively. From Wu et al. (2023).

code that hasn't been done before.

Burgers' equation is a good problem suited for use of adaptive methods, where a very prominent feature in the middle of the domain makes it likely that biased sampling of points will improve the solution. However, this also means that in order to show that use of adaptive methods is useful more generally, other benchmarks should be looked at too. The next step in the plan is therefore to investigate the effectiveness of the adaptive sampling for improving the accuracy if a PINNs solving a second benchmark problem. As this was also done by Wu et al. (2023), there are a few sample benchmarks to chose from. Solving an inverse problem (where the parameters of the equation are unknown and one of the objectives) is one way of doing this, and a potential problem is the Korteweg-de Vries equation, as it is also a general benchmark case with a known exact solution. This time, the outcome would be a relative error of both the velocity and the unknown parameters.

Due to the novelty of the method in an area that has shown a lot of potential for publication so far (as covered in chapter 1.2), it is therefore reasonable to pursue publication in a journal at this stage for the work done so far. A reasonable amount of time is allocated in the plan at this stage for the initial write-up of the paper on this work, aiming to submit for review by the end of 2023. Alternatively, if the work carried out is not considered mature enough for a full journal publication at that stage

the aim would be to instead prepare a conference paper proposal for conferences in the new year, specifically ICCS - more information in section 5.3.

5.2 Medium & Long Term

Due to the fast-paced nature of PINNs research, the longer term plan for the PhD is more speculative. In 2024 I would aim to tackle the objectives regarding goal-based adaptive sampling for PINNs. Initially some time must be allocated for initial study of relevant literature. With a better understanding of goal-based error estimation and adaption in FE, the plan is to then work on creating an analogue adaptive sampling method for PINNs and investigate its potential. This would be the main focus of my PhD for the majority of 2024.

At the moment, while adaptive collocation sampling methods have been shown to bring some accuracy benefits over the more common random implementation, research so far has been mostly on simpler benchmarks in 2 dimensions (Wu et al., 2023). However, there is a lot of interest in the area of using deep learning to solve high PDE equations, due to them being potentially free of the curse of dimensionality Weinan et al. (2022). For this reason, attempting to implement this method and bring its associated benefits to the solving of higher dimensional problems could be a good plan for subsequent work in the long term. Example problems are the solving of the Schrödinger equation in the many-body problem, or the nonlinear Black-Scholes equation; where non-physics-informed deep learning approaches have been tested (Han et al., 2018).

It is also planned to continue monitoring the literature on PINNs implementation more generally in order to stay on top of potential areas for research that address the research focus of this PhD. Topics such as domain decomposition also show potential to improve PINNs generally in their application to engineering problems, and could be an alternative area of study in the case that adaptive sampling does not live up to expectations.

Lastly, attendance to conferences in order to present work is also expected from 2024 onwards. For this reason the time it may take to prepare for and attend these should be taken into account. The decision of which would be best to attend should be taken further on depending on the nature of the work to present, but some possibilities will be discussed in the next section, 5.3.

At the beginning of the last year of the PhD, the plan is to construct the final

structure for the thesis. This should allow me to evaluate what work needs to be completed in the next 6 months in order to fulfill all the pass requirements. This allows the first half of the last year to be dedicated to carrying out and publishing any final work, and the second half of the last year to be dedicated mostly to writing up the thesis and preparing for the viva. This plan thus allows for a couple of months to take into account annual leave and unplanned setbacks.

5.3 Publishing Goals and Conferences

Active interest in PINNs research means a lot is published regarding both applications of PINNs and advances in PINNs of many forms. Work in adaptive sampling for PINNs in particular has been published in a number of different journals broadly interested in scientific computing. The following list covers the journals I would attempt to publish in.

- Journal of Computational Physics (JCP): Accepts contributions dealing with "computational aspects of physical problems" and encourages original scientific contributions in advanced mathematical and numerical modeling. The landmark paper on PINN Raissi et al. (2019) was published in JCP. It publishes many papers on PINNs and data-driven methods.
- Computer Methods in Applied Mechanics and Engineering (CMAME): Covers use of computational methods for the simulation of complex physical problems, specifically mentioning fluid mechanics as an example problem field. It has specifically accepted contributions on the field of adaptive collocation, publishing the Wu et al. (2023) I am basing my work on.
- Journal of Scientific Computing (JSC): Looks for "developments in scientific computing and its applications in science and engineering". Has covered work such as extending PINN method to help solve PDEs Wu et al. (2022), and also publishes work on data-driven methods.

As mentioned above, there are at least 3 planned publications at the moment based on work on adaptive sampling of collocation points. Due to the nature of the work being very focused on a PINN based method, all of the above journals would be relevant, good quality candidates.

With regards to conferences, I will be attending the UK Fluids conference once a year with the CDT. This year, I will be presenting a poster on my project there, and the following years I will be giving a talk. As well as those talks, I should also look for the opportunity to present at other conferences in scientific computing starting next year.

One particularly relevant conference is the ICCS (International Conference in Computational Science) hosted annually. The next conference in summer 2024 could be a great opportunity to submit my work done to date and obtain a fully refereed conference paper. This would be a great opportunity to prove the quality of my work to date and to gain experience presenting my work in a highly relevant conference.

Whilst there are many conferences with a focus on AI and machine learning, a lot are relevant to purely data-driven methods and commercial applications, not being as relevant to scientific and engineering problems as those I am interested in with PINNs. However, there are still some like the Brown University International Conference on Mathematical and Scientific Machine learning, that are specific to scientific ML and could also be beneficial conferences to attend to present depending on the state of my work.

5.4 Gantt Chart

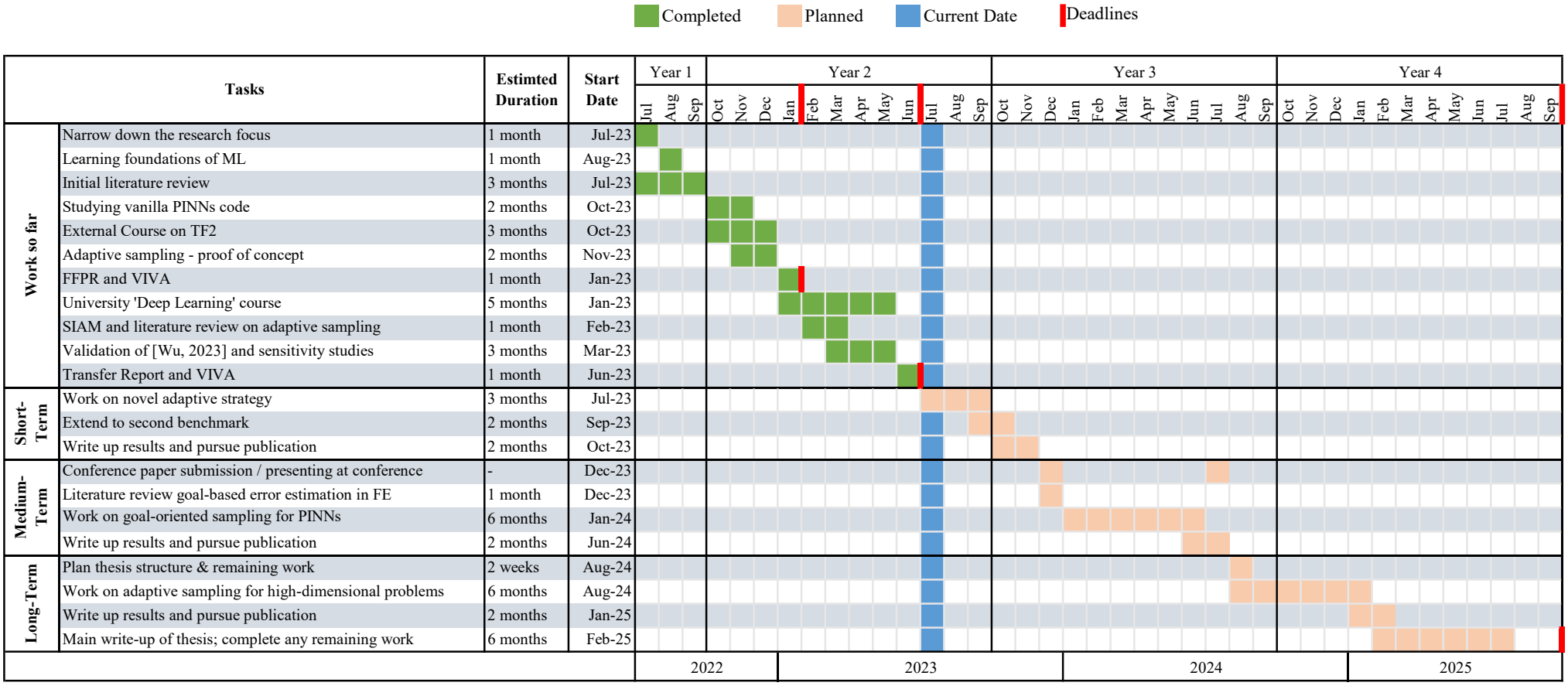


Figure 5.2: Gantt Chart summary of the research plan described in this chapter, from start to end of the PhD.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, USA. USENIX Association.

Ainsworth, M. and Oden, J. T. (2000). A posteriori error estimation in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*.

Al-Shafei, A., Zareipour, H., and Cao, Y. (2022). A review of high-performance computing and parallel techniques applied to power systems optimization. *ArXiv*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Cai, S., Wang, Z., Wang, S., Perdikaris, P., and Karniadakis, G. E. M. (2021). Physics-informed neural networks for heat transfer problems. *JOURNAL OF HEAT TRANSFER-TRANSACTIONS OF THE ASME*, 143(6).

Cueto, E. and Chinesta, F. (2023). Thermodynamics of learning physical phenomena. *Archives of Computational Methods in Engineering*.

Cuomo, S., Cola, V. S. D., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. (2022). Scientific machine learning through physics-informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92:88.

Geneva, N. and Zabaras, N. (2020). Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056.

Goswami, S., Bora, A., Yu, Y., and Karniadakis, G. E. (2022). Physics-informed deep neural operator networks. *ArXiv*, abs/2207.05748.

Hammersley, J. and Handscomb, D. (1964). Monte carlo methods, methuen & co. *Ltd., London*, 40:32.

Han, J., Jentzen, A., and E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510.

Hart, D. E., Berzins, M., Goodyer, C. E., and Jimack, P. K. (2011). Using adjoint error estimation techniques for elastohydrodynamic lubrication line contact problems. *International Journal for Numerical Methods in Fluids*, 67:1559–1570.

Hennigh, O., Narasimhan, S., Nabian, M. A., Subramaniam, A., Tangsali, K. M., Rietmann, M., del Águila Ferrandis, J., Byeon, W., Fang, Z., and Choudhry, S. (2020). Nvidia simnet{$\hat{}$TM}: an ai-accelerated multi-physics simulation framework. In *International Conference on Conceptual Structures*.

Jagtap, A. and Karniadakis, G. (2020). Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28:2002–2041.

Jagtap, A. D., Kharazmi, E., and Karniadakis, G. E. (2020). Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, 3:422–440.

Kharazmi, E., Zhang, Z., and Karniadakis, G. (2021). hp-vpinns: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547.

Kollmannsberger, S., D'Angella, D., Jokeit, M., and Herrmann, L. (2021). *Physics-Informed Neural Networks*. Springer Cham.

Krishnapriyan, A. S., Gholami, A., Zhe, S., Kirby, R. M., and Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 26548–26560. Curran Associates, Inc.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Liu, X.-Y. and Wang, J.-X. (2021). Physics-informed dyna-style model-based deep reinforcement learning for dynamic control. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2255):20210618.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. (2021). Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63:208–228.

Mao, Z., Jagtap, A. D., and Karniadakis, G. E. (2020). Physics-informed neural networks for high-speed flows. *COMPUTER METHODS IN APPLIED MECHANICS AND ENGINEERING*, 360.

Molland, A. F. and Turnock, S. R. (2007). 6 - theoretical and numerical methods. In Molland, A. F. and Turnock, S. R., editors, *Marine Rudders and Control Surfaces*, pages 233–311. Butterworth-Heinemann, Oxford.

Nabian, M. A., Gladstone, R. J., and Meidani, H. (2021). Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 36:962–977.

Nilsson, N. J. (1965). *Learning machines : foundations of trainable pattern-classifying systems*. McGraw-Hill, New York.

Oppenheimer, M. W., Doman, D. B., and Merrick, J. D. (2023). Multi-scale physics-informed machine learning using the buckingham pi theorem. *JOURNAL OF COMPUTATIONAL PHYSICS*, 474.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA.

Rackauckas, C. and Nie, Q. (2017). Differentialequations.jl - a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5.

Raissi, M. (2018). Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research 19*.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707.

Sharma, P., Chung, W. T., Akoush, B., and Ihme, M. (2023). A review of physics-informed machine learning in fluid mechanics. *Energies*, 16(5).

Subramanian, S., Kirby, R. M., Mahoney, M. W., and Gholami, A. (2022). Adaptive self-supervision algorithms for physics-informed neural networks. *arXiv e-prints*.

Sun, L., Gao, H., Pan, S., and Wang, J. X. (2020). Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361.

Taebi, A. (2022). Deep learning for computational hemodynamics: A brief review of recent advances. *Fluids*, 7:197.

Venditti, D. A. and Darmofal, D. L. (2000). Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow. *Journal of Computational Physics*, 164:204–227.

Vinuesa, R. and Brunton, S. L. (2022). Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358–366.

Wang, J. X., Wu, J. L., and Xiao, H. (2017). Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data. *Physical Review Fluids*, 2.

Weinan, E., Han, J., and Jentzen, A. (2022). Algorithms for solving high dimensional pdes: from nonlinear monte carlo to machine learning. *Nonlinearity*, 35:278–310.

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

Wong, T.-T., Luk, W.-S., and Heng, P.-A. (1997). Sampling with hammersley and halton points. *Journal of Graphics Tools*, 2(2):9–24.

Wu, C., Zhu, M., Tan, Q., Kartha, Y., and Lu, L. (2023). A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403.

Wu, W., Feng, X., and Xu, H. (2022). Improved deep neural networks with domain decomposition in solving partial differential equations. *Journal of Scientific Computing*, 93.

Yu, J., Lu, L., Meng, X., and Karniadakis, G. E. (2022). Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Computer Methods in Applied Mechanics and Engineering*, 393.

Zhu, Q., Liu, Z., and Yan, J. (2021). Machine learning for metal additive manufacturing: predicting temperature and melt pool fluid dynamics using physics-informed neural networks. *Computational Mechanics*, 67:619–635.

Appendix A : Appendix 1

A.1 Responsible Research & Innovation (RRI) Plan

The work planned as part of this PhD project does not fall within any of the UN sustainable development goals. The technology readiness level (TRL) of PINNs broadly is around 3 - meaning proof of concept has been made and research is focused on developing the technology further. Most research involving PINNs follows that definition, and involves looking for new implementations of PINNs or testing the feasibility of PINNs for different applications. Considering the specific field of adaptive collocation point sampling, at the moment the TRL would be even lower at 1 (the benefits of adaptive collocation point have been observed but not yet applied or tested experimentally on real cases), with an aim of reaching a TRL of 5 at the end of the PhD. This would mean benefits from adaptive sampling have been shown and validated in a relevant application of PINNs in industry.

Due to the low research impact potential of 1 and the lack of UN goals covered by the project, the RRI intensity level is 2 for my PhD, which is low. This means the level of action required with RRI in mind is going to be very low, falling into the category of "unaware", where it would be appropriate for the research group to not incorporate explicit RRI techniques in the processes. However, there is no detriment to implementing RRI. Some improvements to RRI could still be made, as an example ensuring my work is open access is an easy way of promoting follow up research by others.

A.2 Training Needs and Plan

A lot of training has been completed until this point, both for soft and hard skills. For soft skills related to research, through the CDT I've been able to both present my own work in presentation format (at the School of Computing colloquium and the CDT internal seminar) and poster format (at the Leeds Fluids Symposium), giving me practice presenting which will be useful in advance of presenting at larger conferences. Professional development sessions focused on publishing as well as my own experience having work from my Master's Project published means I have had a good insight into what the publishing process is like, from the first submission to how to properly

go through the revision process. All of this should hopefully mean that publishing my own work should be easier in future. Other general development was also done with the professional development sessions on things such as ethics and outreach. Considering further opportunities will also be available to me via the CDT, it isn't necessary to further plan training for other soft skills.

As for hard skills, external courses I've completed on machine learning and TensorFlow 2 specifically have allowed me to quickly catch up to all the programming knowledge I need to work with the code presented in the literature and be able to modify and create my own. The Deep Learning module undertaken as part of my second year has also given me more broad knowledge of machine learning methods and some limited practice using the PyTorch library. However, as mentioned in the research plan, continuing onto the last TensorFlow 2 external course by Imperial College London is a priority after the transfer report. Other necessary skills such as learning how to use LaTeX and HPC were also learned in the integrated masters year. However, as I will want to transition to using HPC to speed up simulations, I have also signed up to a free dedicated workshop "HPC1: HPC Carpentry" with the university's IT deparment. This is taking place over 2 sessions at the end of June and beginning of July and will help me recap through additional training.

Additional training needs however should be revised in future years of the PhD, especially if other areas of interest within the field of PINNs are explored. However, it is unlikely that other programming languages or software will need to be learned. Free opportunities for soft skill training are also sometimes available within the university, so it is always worth occasionally checking within the university newsletters and the skills@library team. For example, I'd be interested in signing up for advanced academic writing workshops if any were offered by the university - but there is currently no need to actively plan or pursue this.

A.3 Data Management Plan

Currently, code is written in python using Visual Studio Code software, and linked to Github. This incorporates version control to all the code I write and I periodically sync the code/analysed data/reports to github, effectively backing up the more important information online. The raw results usually occupy more storage space and so are not backed up online, but if it's necessary to keep these it would be worth making local

copies on either external hard disk drives or on other computers. Aside from the code, most documents related to the PhD (Such as write-ups, presentations and posters) are backed up to OneDrive. This includes my work diary on OneNote which I use to keep track of meetings and work for GRAD, which automatically syncs up to the cloud.

A.4 Expenditure Plan

As it is unlikely there will be any experimental aspects to this project, it is safe to assume the majority of costs incurred will be associated with attending of conferences and seminars. The purpose of these is to become involved with the scientific machine learning community. This should be especially important in this PhD due to the quickly evolving nature of the research, but it would also be a great opportunity for disseminating my work in order to receive constructive criticism and feedback. International conferences however can be particularly expensive, and so it is important to weigh up the benefit to cost ratio when deciding which are best to attend.