Finding a Convergence Theory for PINNs

Aengus Roberts

Department of Mathematical Sciences
University of Bath

May 9, 2025

1 Introduction

Differential Equations are a keystone of scientific research with applications including modelling drug dispersion in pharmacology, predicting markets in economics, predicting the weather, and modelling atomic interactions within quantum physics. Whilst it is often easy to formulate differential equations, finding solutions to them can prove to be much harder. Tthrough this difficulty, the field of numerical analysis has grown. Often analytic solutions can be hard or impossible to find and instead one must find an approximate solution to the differential equation. We note that differential equations can broadly be split into two categories, Ordinary Differential Equations (ODEs), problems formulated using only total derivatives of functions, and Partial Differential Equations (PDEs), problems formulated using partial derivatives of functions.

Further categorisations exist (elliptic, non-linear, etc) to describe smaller classes of differential equations, often with accompanying theorems related to the existence and uniqueness of solutions. Numerical methods often leverage behaviours of these subclasses of problems however there exist methods that aim to be generic solvers, applicable large classes of problems with little to no problem specific implementations (i.e. leverage of symmetries, conservation laws or meta-knowledge of the problem). Since the advent of the computer in the 20th century numerical methods have drastically evolved from by-hand calculated interpolations to solvers of multidimensional problems.

Whilst one could write entire books (and this has been done) on the importance of finding solutions to differential equations, it should hopefully be apparent to the reader of this fact and we shall therefore not motivate further this importance. We therefore turn our interest to modern methods of approximating solutions to differential equations and in particular Physics Informed Neural Networks (PINNs) [1] in comparison to other modern computational approaches such as Finite Difference [2] and Element [2] Methods (FDM/FEM) and collocation methods [2]. These methods will be explored in depth in their respective chapters: FDM in section 2.1, FEM in section 2.2, Collocation methods in 2.3 and PINNs in section 3 but we shall provide a brief overview of their main characteristics and differences here.

We first consider a general boundary value problem (BVP)

$$\begin{aligned} Du(x) &= f(x) \qquad x \in \Omega \\ u(x) &= g(x) \qquad x \in \partial\Omega \end{aligned} \tag{1}$$

with differential operator D, function $u(x)$, source term $f(x)$ and boundary data $g(x)$.

Collocation Method

Collocation Methods consider a finite approximation of the problem. To do this we consider the collocation points $x_i \in \Omega$ for $i \in \{0, 1, \dots, N\}$ with $\{x_0, x_N\} = \partial\Omega$. An approximate solution, $\tilde{u}$, is then built up from a finite dimensional space of candidate solutions - usually polynomials. If the approximate solution satisfied the collocation methods formulation of the BVP at these points it is a numerical solution, $\hat{u}$, to the problem. That is, if $u(x_i) = \tilde{u}(x_i) \quad \forall i \in \{0, 1, \dots, N\}$ then $\tilde{u} = \hat{u}$. We note here that $\hat{u}$ may not be unique, even if u is a unique solution to (1).

Finite Difference Method

FDM approximates derivatives with finite differences, given a step size between points, h, one could approximate the derivative of a function u at a point x_i as

$$u'(x_i) \approx \frac{u(x_{i+1}) - u(x_i)}{h} \tag{2}$$

where $x_{i+1} = x_i + h$. This is known as a first order forward difference approximation. A first order backwards difference approximation would be given by

$$u'(x_i) \approx \frac{u(x_i) - u(x_{i-1})}{h} \tag{3}$$

and a first order central difference approximation is given by

$$u'(x_i) \approx \frac{u(x_{i+1}) - u(x_{i-1})}{2h} \tag{4}$$

Higher order approximations (approximating $u^{[k]}(x_i)$ with values of $u(x_k)$ for $k \in 1, \ldots, N$) exist and can be used for higher accuracy as well.

Finite Element Method

FEM discretises the solution domain into simple 'elements' and constructs a piecewise approximation of the solution using basis functions on these elements. The solution is found by minimising the residual error given by the linear combination of basis functions. This transforms the original problem into a set of algebraic equations by enforcing the differential equation in a weak form over each element.

Physics Informed Neural Networks

PINNs act similar to collocation methods, aiming to find an approximate solution to (1) using a set of collocation points. Instead of building a solution from a linear class of candidate functions such as polynomials, a neural network is used as a non-linear approximation of this function. A variation of gradient descent is used with a loss function containing the residual of (1). That is $R(\tilde{u}(x_i)) = |D\tilde{u}(x_i) - f(x_i)| \quad \forall i \in \{0, 1, \ldots, N\}$ evaluated at the collocation points. Minimising this, with $R(\tilde{u}) = 0$, is equivalent to finding $\tilde{u}$ such that $u(x_i) = \tilde{u}(x_i) \quad \forall i \in \{0, 1, \ldots, N\}$ as in the collocation methods case.

PINNs represent a significant advancement in the numerical solution of differential equations, particularly for problems where traditional methods face limitations. Classical numerical methods such as FDM and FEM rely on mesh-based discretisation, which can become computationally expensive and inflexible when dealing with high dimensional problems or complex geometries. In contrast, PINNs leverage the expressive power of neural networks to approximate solutions over the entire domain without explicit meshing. This mesh-free formulation allows PINNs to handle irregular domains and higher-dimensional PDEs with minimal adjustments. Furthermore, PINNs can incorporate both initial and boundary conditions directly into the loss function, ensuring they are respected throughout the solution space. Making them particularly well-suited for solving inverse problems, where parameters of the PDE are unknown and need to be inferred during training. The combination of universal function approximation with embedded physical laws enables PINNs to achieve high accuracy while reducing the computational overhead typically associated with classical methods. This flexibility and efficiency are driving growing interest in PINNs as a powerful alternative to traditional numerical solvers.

2 Overview: Classical Methods

We use the term "Classical Methods" here to describe non-neural network oriented approaches and for each method identify convergence theorems showing that $\tilde{u}$ well approximates the true solution to

(1) u.

2.1 Finite Difference Method

Two simplifying assumptions are made within FDM when solving a BVP. The first approximates the domain Ω with a finite number of mesh points $\pi = \{x_0, x_1, \ldots, x_N\} \in \Omega$ where $\{x_0, x_N\} \in \partial\Omega$. The second and defining feature of finite difference methods is the approximation of derivatives with finite differences. The first derivative of a function $u \in C^1(\mathbb{R})$ at a point x_i can be given by

$$\frac{du}{dx}(x_i) = \lim_{h \to 0} \frac{f(x_i + h) - f(x_i)}{h}. \tag{5}$$

Assuming a finite step size h this gives us

$$\frac{du}{dx}(x_i) \approx \frac{f(x_i + h) - f(x_i)}{h} \tag{6}$$

and if we assume a uniform sampling of our domain Ω with a step size of h between mesh points (i.e. $x_{i+1} = x_i + h$) then we have further

$$\frac{du}{dx}(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{h}. \tag{7}$$

This is known as the first order forward difference approximation. Forward approximations often suffer from numerical instabilities, we therefore may want to instead consider a different approximation of the derivative. A backwards difference approximation can be obtained similarly making use of the equality of the left and right derivatives of C^1 functions which is given by

$$\frac{du}{dx}(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{h}. \tag{8}$$

Taking the mean of these two approximations gives the frequently numerically stable centre difference approximation

$$\frac{du}{dx}(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2h}. \tag{9}$$

One can similarly approximate higher order derivatives in a recursive manner, the second order centre difference approximation is given by

$$\frac{d^2u}{dx^2}(x_i) \approx \frac{\frac{u(x_{i+1}) - u(x_i)}{h} - \frac{u(x_i) - u(x_{i-1})}{h}}{h} = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2}. \tag{10}$$

Partial Derivatives can be approximated similarly in the case of functions of multiple variables where the finite differences are derived from definitions of the partial derivative. Derivatives at the boundaries need to be treated with care and instead of the finite differences used to approximate differences in the interior of the domain, boundary data from the BVP can instead be used. If no such data is given a one sided derivative may be used in place.

2.1.1 Definition

These finite differences can be used to reformulate (1) as

$$D_\pi \tilde{\mathbf{u}} = \mathbf{f} \tag{11}$$

where $D_\pi \in \mathbb{R}^{N \times N}$ is a matrix operator containing the interior and boundary finite difference approximations for a given mesh π. $\tilde{\mathbf{u}} = [\tilde{u}(x_0), \tilde{u}(x_1), \ldots, \tilde{u}(x_N)]^T$ is a vector of approximate solutions at our mesh points and $\mathbf{f} = [f(x_0), f(x_1), \ldots, f(x_N)]^T$ is a vector of the values our source term takes at the mesh points. This can now be solved as a linear algebra problem using either explicit or implicit methods.

We note here that if centre difference approximations are used then $\tilde{D}$ is a constant tridiagonal matrix provided the BVP is a second order problem or lower. This allows for fast inversion of $\tilde{D}$ to obtain a solution $\tilde{\mathbf{u}}$.

Definition 2.1 (Finite Difference Method) *A finite difference method is one that follows the outlined basic steps.*
Input: A BVP defined on $[a, b]$
Output: An approximate solution $\tilde{u}(x)$

1. *Choose a mesh $\pi : a = x_0 < x_1 < \ldots < x_{N-1} < x_N = b$*
2. *Form a set of algebraic equations for the approximate solution values by replacing derivatives with finite difference approximations.*
3. *Solve the resulting system of equations for the approximate solution $\tilde{\mathbf{u}}$.*

This gives a set of discrete solution values satisfying the BVP at the mesh points π.

2.1.2 Simple Example

We make use of the simple one dimensional problem

$$\begin{gathered} \frac{d^2u}{dx^2}(x) + \frac{du}{dx}(x) + u(x) = f(x) \\ u(0) = u(1) = 0 \end{gathered} \tag{12}$$

where $u : [0, 1] \to \mathbb{R}$ and $A \in \mathbb{R}$ is a constant as a motivating example. We shall let the forcing term be $f(x) = 1$, $\forall x \in [0, 1]$. First considering the interior mesh points $x_i \in \{x_1, \ldots, x_{N-1}\}$, we can formulate the problem as

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \frac{u_{i+1} - u_{i-1}}{h} + u_i = 1 \tag{13}$$

which can be simplified to

$$(1 - h)u_{i-1} + (2 - h^2)u_i + (1 + h)u_{i+1} = h^2 \tag{14}$$

where here we are using the notation u_i to denote $\tilde{u}(x_i)$. We can now build our matrix as

$$D_\pi = \begin{pmatrix} 1 & 0 & \ldots & \ldots & \ldots & 0 \\ 1-h & h^2-2 & 1+h & 0 & & \\ & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \\ & & 0 & 1-h & h^2-2 & 1+h \\ 0 & & \ldots & & 0 & 1 \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ \vdots \\ 1 \\ 0 \end{pmatrix} \tag{15}$$

where the first and last rows have the boundary conditions directly built into the matrix. Numerical methods can now be applied to solve for $\tilde{\mathbf{u}}$ in (11).

2.1.3 Convergence Theorem

Following Ascher, Mattheij and Russell [2] we will show that under sufficient conditions for a first order BVP, the approximate solution returned by a finite difference method converges to the true solution of the BVP. In this section we shall only consider the linear first order BVP

$$\begin{aligned}\mathbf{u}'(x) - A(x)\mathbf{u}(x) &= \mathbf{q}(x) \qquad a < x < b \\ B_a\mathbf{u}(a) + B_b\mathbf{u}(b) &= \boldsymbol{\beta}\end{aligned} \tag{16}$$

where $A(x), B_a, B_b \in \mathbb{R}^{N\times N}$. In this case we have $D = \frac{d}{dx} - A(x)$ and D_π its approximation under the given mesh π. Under the FDM scheme this becomes

$$D_\pi \tilde{\mathbf{u}} = \mathbf{q} \tag{17}$$

We first define two concepts/properties of a finite difference method before showing that they are sufficient for convergence. Briefly, these concepts can be understood as

- Consistency: The difference operator D_π approaches the differential operator D as $h \to 0$
- Stability: The inverse of the difference operator D_π is suitably bounded

but will be more formally defined later. This definition of convergence can then be generalised to higher order and non-linear problems (such as the second order problem defined in section 2.1.2).

Definition 2.2 (Consistency) *The local truncation error $\boldsymbol{\tau}_i[\mathbf{u}]$ is defined as*

$$\boldsymbol{\tau}_i[\mathbf{u}] := D_\pi\mathbf{u}(x_i) - \mathbf{q}(x_i), \qquad \forall\, 1 \le i \le N-1 \tag{18}$$

and a finite difference scheme is said to be consistent of order $p > 0$ if for every solution of (16) there exists constants c and h_0 such that for all meshes π with $h := \max_{0\le i\le N} h_i \le h_0$,

$$\tau[\mathbf{y}] := \max_{0\le i\le N} |\boldsymbol{\tau}_i[\mathbf{y}]| \le ch^p \tag{19}$$

Definition 2.3 (Stable) *A scheme is said to be stable if there exists constants K and h_0 such that for all meshes π with $h \le h_0$ and all mesh functions $\mathbf{v}_\pi$,*

$$|\mathbf{v}_i| \le K \max\left\{ |B_a\mathbf{v}_0 + B_b\mathbf{v}_N|\, ,\, \max_{0\le i\le N} |D_\pi \mathbf{v}_j| \right\} \tag{20}$$

and K is of moderate size if the BVP conditioning constant κ is.

Definition 2.4 (Convergent Scheme) *The solution of 17 is said to converge to the solution of 16 if*

$$\max_{0\le i\le N} |\tilde{\mathbf{u}}_i - \mathbf{u}(x_i)| \to 0 \qquad \textit{as} \quad h \to 0 \tag{21}$$

It can now be shown, but omitted here for brevity (see Ascher, Mattheij and Russell [2] for proof) that

Theorem 2.5

Consistency + Stability = Convergence $\square$

This theorem can then be generalised to both non-linear and higher order systems [2].
a

2.1.4 FDM Drawbacks

FDM's largest flaw is its reliance on grid discretisation. We can clearly see that sharp gradients and boundary layers will cause issues for a system like this and a fine mesh would be required globally in order to maintain accuracy in such a case. Furthermore, FDM is confined to simple geometries due to the formulation of the mesh making it impractical for complex domains.

Higher order problems generate their own issues. Finite difference approximations of higher order differences often contain terms such as $u(x_{i\pm 2})$ or $u(x_{i\pm 3})$ which makes formulation of the problem near boundaries an issue. Enforcing boundary conditions is also an awkward endeavour, often problem specific and may require fine tuning or repeated attempts in order to formulate the problem efficiently.

2.2 Finite Element Method

Finite element methods are the current industry standard for solving many differential equations and continue to evolve as computers do, from barely managing simple three dimensional problems in the 1980's to modelling complex multidimensional geometries in the present day. As a certification of this fact we need not look any further than the many packages and programs that exist purely to solve problems through the FEM schema.

FEM is likely best identified through its meshes and for many the name likely conjures images of triangulations of complex domains. FEM subdivides the problem domain into simple pieces called finite elements. These elements are often simple shapes such as triangles or rectangles in two dimensions and tetrahedra or hexahedra in three dimensions. By assembling these elements, FEM constructs a global approximation of the solution over the entire domain, allowing it to handle complex geometries and irregular boundaries with more ease than a FDM.

The core idea behind FEM is to reformulate the differential equation into a variational form, which involves integrating the equation against a set of test functions. This variational form transforms the problem of solving a differential equation into solving a system of algebraic equations. If we consider 1, FEM seeks an approximate solution u_h in a finite dimensional subspace V_h such that:

$$d(u_h, v) = (f, v) \tag{22}$$

where $d(\cdot,\cdot)$ is a bilinear form representing the differential operator, D, and $(\cdot,\cdot)$ is the inner product. The choice of basis functions for V_h defines the accuracy and smoothness of the solution but often at the cost of computational time. These functions are normally choses as piecewise polynomials (splines) defined to be non-zero only within specific elements. This means that when formulated as a matrix problem the matrix is sparse and furthermore, symmetric positive definite allowing for easier applications of sparse matrix techniques.

2.2.1 Definition

Through this section we shall give the definition of a FEM through the lens of a the simple problem as worked through by Larsen and Bengzon [3]

$$\begin{aligned} -u''(x) &= f(x), \qquad x \in I = [0, L] \\ u(0) &= u(L) = 0. \end{aligned} \tag{23}$$

To obtain the variational form we multiply the problem by a test function v which is assumed to vanish at $x = 0$ and $x = L$. Integrating by parts gives us

$$\begin{aligned}\int_0^L fv\,dx &= -\int_0^L u''v\,dx \\ &= \int_0^L u'v'\,dx - u'(L)v(L) + u'(0)v(0) \\ &= \int_0^L u'v'\,dx\end{aligned} \tag{24}$$

Considering the space of functions that satisfy the above behaviours we have

$$V_0 = \{v : ||v||_{L^2(I)} \leq \infty, ||v'||_{L^2(I)} \leq \infty, v(0) = v(L) = 0\} \tag{25}$$

which one can see is clearly an infinite dimensional space. To reduce this we introduce a mesh on the interval I made of n subintervals and the corresponding space V_h of all continuous piecewise linear functions on this mesh. Applying our boundary condition to the test functions gives us the finite dimensional space

$$V_{h,0} = \{v \in V_h : v(0) = v(L) = 0\} \tag{26}$$

where we can note a basis for this space is the set of triangular functions defined at the interior points of the mesh $\{\varphi_j\}_{j=1}^{N-1}$. Using this space we obtain the following FEM: find $u_h \in V_{h,0}$ such that

$$\int_I u_h' v'\,dx = \int_I fv\,dx, \qquad \forall v \in V_{h,0}. \tag{27}$$

Since $u_h \in V_{h,0}$ we can write it as a linear combination

$$u_h = \sum_{j=1}^{n-1} \xi_j \varphi_j \tag{28}$$

and making use of the basis functions again we have

$$\int_I f\varphi_i\,dx = \int_I \left(\sum_{j=1}^{n-1} \xi_j \varphi_j'\right) \varphi_i\,dx, \qquad i = 1, \dots, n-1. \tag{29}$$

We can define the stiffness matrix A and load vector b as

$$A_{ij} = \int_I \varphi_i' \varphi_j'\,dx, \qquad i,j = 1, \dots, n-1 \tag{30}$$

$$b_i = \int_I f\varphi_i\,dx \qquad i = 1, \dots, n-1. \tag{31}$$

This allows us to formulate the problem once again as a linear system

$$A\xi = b \tag{32}$$

.

Definition 2.6 (Finite Element Method) *A one-dimensional finite element method is one satisfying the following steps:*

1. *Create a n element mesh on the domain I and define its corresponding finite dimensional solution space $V_{h,0}$.*

2. Compute the $(n-1) \times (n-1)$ stiffness matrix A and load vector b.

3. Solve the linear system 32.

4. Calculate u_h as in 28.

This gives a piecewise linear solution u_h over the mesh that is the best approximation of the true solution within the finite space $V_{h,0}$.

2.2.2 Convergence Theorem

Here we introduce Cea's Lemma [4] which bounds above the error in the approximate solution by the error in the interpolant. In the limit as the mesh size $h \to 0$ we have that the interpolant tends to the true solution, therefore so must the approximate solution. We define the one dimensional linear interpolant as

$$\pi_u = \sum_{i=0}^{n-1} \mathbb{1}_{[x_i, x_{i+1}]} \left[u(x_i) + \frac{u(x_{i+1}) - u(x_i)}{x_{i+1} - x_i}(x - x_i) \right]. \tag{33}$$

Theorem 2.7 (Cea's Lemma) *Given a true solution to a BVP u, the best approximate solution within a finite dimensional class of functions $u_h \in V_h$, and the linear interpolant $\pi_u \in V_h$ we have:*

$$||u - u_h||^2 \leq |||u - \pi_u||^2 \tag{34}$$

Proof: Consider the error in the interpolant:

$$\begin{aligned} ||u - \pi_u||^2 &= \langle u - \pi_u, u - \pi_u \rangle \\ &= \langle u - u_h + u_h + \pi_u, u - u_h + u_h + \pi_u \rangle \\ &= \langle u - u_h, u - u_h \rangle + 2\langle u - u_h, u_h - \pi_u \rangle + \langle u_h - \pi_u, u_h - \pi_u \rangle \\ &= ||u - u_h||^2 + ||u_h - \pi_u||^2 + 2\langle u - u_h, v \rangle \end{aligned}$$

where $v = u_h - \pi_u \in V_h$. As u_h is the best approximation of u in V_h we have that $u - u_h$ must be orthogonal to any vector in V_h including v. Therefore we have $\langle u - u_h, v \rangle = 0$ giving us

$$||u - \pi_u||^2 = ||u - u_h||^2 + ||u_h - \pi_u||^2$$

and as $||u_h - \pi_u||^2 \geq 0$ we must have that

$$||u - u_h||^2 \leq |||u - \pi_u||^2 \qquad \square$$

Remark 2.8 *Although derived for a one dimensional interval, both the definition and convergence theorem proposed in this section generalise to multiple dimensions, non-linear systems and much more abstract classes of meshes.*

2.2.3 FEM Drawbacks

Due to working with often complex meshes, FEM can generate large, sparse, linear systems which require significant computational power to solve. Mesh refinement to obtain higher accuracy solutions directly increases the size of the system leading to greater memory usage and longer computational times.

The generation of high quality meshes for complex geometries often requires specific software and mesh generation algorithms, without which can lead to poorly generated meshes, in turn leading to ill-conditioned matrices and inaccurate solutions. Many physical properties that one may wish to model such as thin boundary layers or large deformations can often cause issues with mesh generation as well.

FEM relies on numerical quadrature to approximate integrals. Whilst in the case of the stiffness matrix this usually isn't an issue due to the simplicity of the functions, the load vector calculation has no such guarantee providing potential for the introduction of further errors into the solving process. For higher order elements or very curved geometries numerical integration may also cause further issues.

FEM can struggle to encode multi-scale behaviours into its solution. Consider a function that oscillates rapidly but is enveloped by a slower oscillating function (i.e. $u(x) = \sin(100x) \cdot \sin(x)$). Here a mesh would have to be globally very fine in order to well capture the fast oscillations whilst still observing the slower oscillating enveloping function requiring large amounts of memory and computational power. Much of the computational power saving in FEM is due to relative mesh sizes which are fine when required and coarser when less "behaviour" in the function is occuring.

2.3 Collocation Method

Unlike the finite methods mentioned earlier, which approximate derivatives or are composed of piecewise functions, collocation methods construct approximate solutions by enforcing the differential equation to be satisfied exactly at a discrete set of points called collocation points. These points are strategically chosen to capture the behaviour of the solution to a BVP across the domain, often corresponding to quadrature nodes such as Gauss-Lobatto or Chebyshev points.

The essence of the collocation method is to approximate the BVP solution u with a finite dimensional trial function u_N expressed as the sum of basis functions

$$u_N(x) = \sum_{j=0}^{N} c_j \phi_j(x) \tag{35}$$

where ϕ_j are choses basis functions, and c_j are coefficients to be determined. The differential equation 1 is then enforced at each collocation point x_i

$$Du_N(x_i) = f(x_i), \qquad 0 \leq i \leq N. \tag{36}$$

Satisfying the BVP at precisely $N+1$ collocation points provides $N+1$ conditions which can be used to uniquely specify a polynomial of degree N. This can be done by solving the finite system of algebraic equations for the coefficients c_j using standard linear algebra techniques.

2.3.1 Definition

To define a collocation method we must first define the implicit Runge-Kutta scheme. This is because collocation methods are a subclass of Runge-Kutta schemes and in-fact are a subclass of Runge-Kutta schemes satisfying a specific assumption on the choice of collocation points and precision of the quadrature rule within the scheme.

We will consider the problem

$$\mathbf{u}' = \mathbf{f}(x, \mathbf{u}). \tag{37}$$

Definition 2.9 (k-stage Runge-Kutta scheme) *We define the k-stage Runge-Kutta scheme for 37 by*

$$\mathbf{u}_{i+1} = \mathbf{u}_i + h_i \sum_{j=1}^{k} \beta_j \mathbf{f}_{ij} \qquad 1 \leq i \leq N \tag{38a}$$

$$\mathbf{f}_{ij} = \mathbf{f}(x_{ij}, \mathbf{u}_{ij}) \qquad 1 \leq j \leq k \tag{38b}$$

where

$$x_{ij} = x_i + h_i \rho_j, \qquad 1 \leq j \leq k, 1 \leq i \leq N \tag{39a}$$

$$\mathbf{u}_{ij} = \mathbf{u}_i + h_i \sum_{l=1}^{k} \alpha_{jl} \mathbf{f}_{il} \qquad 1 \leq j \leq k. \tag{39b}$$

We have

$$0 \leq \rho_1 \leq \rho_2 \leq \ldots \leq \rho_k \leq 1$$

as a set of scalable points to define the collocation points x_{ij}.

Here the sum in 38a is a quadrature rule for $\int_{x_i}^{x_{i+1}} \mathbf{f}$ and the sum in 39b is a quadrature rule for $\int_{x_i}^{x_{ij}} \mathbf{f}$.

We now introduce the assumption on the class of k-stage Runge-Kutta schemes that identifies the subclass of schemes that are collocation methods.

Assumption 2.10 *The points ρ_j are distinct. i.e.*

$$0 \leq \rho_1 < \rho_2 < \ldots < \rho_k \leq 1$$

and the precision, s, of the quadrature rule given by 39b equals k.

Here the precision of a quadrature rule is the order of polynomial a it can exactly approximate.

We now let $\mathbf{u}_\pi(x)$ be a polynomial of order $k+1$ defined on $[x_i, x_{i+1}]$ by the interpolation conditions

$$\begin{aligned} \mathbf{u}_\pi(x_i) &= \mathbf{u}_i \\ \mathbf{u}'_\pi(x_{ij}) &= \mathbf{f}(x_{ij}, \mathbf{u}_{ij}) \qquad 1 \leq j \leq k. \end{aligned} \tag{40}$$

where it can be easily shown (see Ascher, Mattheij and Russell [2]) that this is indeed well defined. By extending the polynomial to $[x_{i+1}, x_{i+2}]$ using the same method and continuing in a similar fashion we can obtain a continuous piecewise polynomial function of order $k+1$ on the interval $[a, b]$. Furthermore one can use the fact that

$$\mathbf{u}_\pi(x_{ij}) = \mathbf{u}_{ij} \tag{41}$$

to show that our piecewise polynomial $\mathbf{u}_\pi$ satisfies 37 precisely at the collocation points:

$$\mathbf{u}_\pi(x_{ij}) = \mathbf{f}(x_{ij}, \mathbf{u}_\pi(x_{ij})) \qquad 1 \leq j \leq k, 1 \leq i \leq N \tag{42}$$

Definition 2.11 (Collocation Method) *A collocation method is a method that follows the following steps.*
Input: A BVP defined on $[a, b]$
Output: An approximate solution $\tilde{u}$

1. *Define a mesh $\pi : a = x_0 < x_1 < \ldots < x_N = b$ and a finite dimensional space X_N spanned by basis functions, ϕ_j,(likely polynomials).*
2. *Select a set of collocation points $0 \leq \rho_1 < \rho_2 < \ldots < \rho_k \leq 1$.*
3. *Form a system of algebraic equations by enforcing the differential equation at each collocation point; i.e. $D\tilde{u}(x_{ij}) = f(x_{ij})$*
4. *Solve the resulting system for the coefficients of the trial function: $\tilde{u}_i(x) = \sum_{j=1}^{k} c_j \phi_j(x)$*
5. *Stitch together the $\tilde{u}_i$ terms to obtain $\tilde{u}(x)$.*

2.3.2 Convergence Theorem

Here we introduce two theorems, both without proof identifying how collocation methods converge.

Theorem 2.12 *Given a mesh π, the unique Runge-Kutta method 38a satisfying assumption 2.10 is equivalent to the collocation method 42. Furthermore:*

$$\mathbf{u}_\pi(x_i) = \mathbf{u}_i, \quad \mathbf{u}_\pi(x_{ij}) = \mathbf{u}_{ij}, \qquad 1 \le j \le k, 1 \le i \le N$$

Theorem 2.13 *If a Runge-Kutta scheme satisfies assumption 2.10 then, as a one step scheme, it is accurate to order p for a BVP satisfying $A(x), q(x) \in C^{(p)}[a,b]$ as in 16. Therefore:*

$$|u_i - u(x_i)| = O(h^p) \qquad 1 \le i \le N \tag{43a}$$

$$|u_{ij} - u(x_{ij})| = O(h_i^{k+1}) + O(h^p), \qquad 1 \le j \le k, 1 \le i \le N \tag{43b}$$

2.3.3 Collocation Method Drawbacks

Collocation methods rely on global polynomial interpolants to approximate the solution across the domain. If the true solution contains discontinuities, sharp gradients, or boundary layers, the method often struggles to maintain accuracy. This is particularly evident in high-order polynomial interpolants, which can suffer from Runge's phenomenon near the boundaries.

The choice of quadrature and mesh can also cause issues. While collocation methods are effective for simple geometries, they are less flexible for irregular or highly complex domains. Mapping these geometries to a structured set of collocation points often requires complex transformations, which increases implementation complexity and potential error. Collocation methods traditionally employ fixed collocation points, either equally spaced or based on quadrature rules like Gauss or Lobatto nodes. This rigid structure makes it difficult to refine the approximation locally in regions where the solution changes rapidly. Techniques like adaptive mesh refinement, which are standard in finite element methods (FEM), are not easily applicable without reconstructing the entire solution.

When applied to nonlinear differential equations, collocation methods produce nonlinear algebraic systems that require iterative solvers. Convergence of these solvers is not guaranteed and is heavily dependent on the initial guess, which complicates their application to complex nonlinear problems.

These limitations make collocation methods less suitable for highly irregular domains, problems with sharp transitions, or applications requiring localized error control. In such cases, methods like the finite element method (FEM) or finite difference method (FDM) are often preferred.

3 Overview: PINN's

3.1 Introduction to PINNs

The advent of neural networks has brought with it a family of new methods to find solutions to differential equations. This includes Neural ODEs, Neural Operators, PINNs, Deep Ritz Method and more, all attempting to leverage universal approximation theorems [5–7] to use neural networks as a form of function approximator.

Rassi, Perdikaris and Karniadakis [1] formalised the PINN approach in 2019 and demonstrated its capacity to solve both forward and inverse problems by embedding the differential equation within the loss function of the neural network. They demonstrated its ability using a range of traditional "test" problems such as the Schrodinger, Allen-Cahn and 2D Navier-Stokes equations. By embedding the differential equation within the loss function, the PINN was enabled to approximate solutions without requiring an explicit discretisation of the spatial domain - a "mesh-free" method.

3.1.1 PINN Definition

Consider the BVP given by

$$\begin{aligned} Du(x) &= f(x), \qquad x \in \Omega \\ Bu(x) &= g(x), \qquad x \in \partial\Omega. \end{aligned} \tag{44}$$

We define the residual of a function $\tilde{u}$ at a point $x_0 \in \Omega$ as

$$R(\tilde{u}, x_0) = |D\tilde{u}(x_0) - f(x_0)| \tag{45}$$

and the boundary residual at a point $\bar{x} \in \partial\Omega$

$$\bar{R}(\tilde{u}, \bar{x}) = |D\tilde{u}(\bar{x}) - g(\bar{x})|. \tag{46}$$

Considering a single layer neural network

$$u_\theta = \sigma\left(\sum_i w_i x_i + b_i\right) \tag{47}$$

where σ is a non-linear activation function, w_i, b_i are the weights and biases and x_i is our input. The non-linear activation function allows us is the requirement for the use of the universal approximation theorem. This in turn means we can use our neural network as our approximation $u_\theta = \tilde{u}$.

Much like a collocation method, we next choose sets of collocation points in the interior $\{x_i\}_{0\leq i\leq N_r}$ and on the boundary $\{\bar{x}_j\}_{0\leq j\leq N_b}$ as points to evaluate our approximate function at. We can now define our loss function for training the neural network

$$\begin{aligned} \mathcal{L}(\theta) &= \mathcal{L}_r(\theta) + \mathcal{L}_b(\theta) \\ &= \frac{1}{N_r}\sum_{i=1}^{N_r} R(u_\theta, x_i)^2 + \frac{1}{N_b}\sum_{j=1}^{N_b} \bar{R}(u_\theta, \bar{x}_j)^2 \end{aligned} \tag{48}$$

which combines the mean squared error of the residual at the interior and boundary collocation points. Training the neural network then consists of finding the optimal parameters θ^* such that

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta). \tag{49}$$

An optimal value of θ^* here would provide $u_\theta(x_i) = u(x_i)$ for all collocation points x_i, much like a collocation method would.

Definition 3.1 (Physics Informed Neural Network) *A PINN to solve 44 is a method that adheres to the following steps:*
Input: A BVP defined on domain Ω Output: A trained neural network u_{θ^*} that approximates the solution to the BVP.

1. *Choose sets of collocation points for both the interior $\{x_i\}_{1\leq i\leq N_r}$ and boundary $\{\bar{x}_j\}_{1\leq j\leq N_b}$ of the domain.*

2. *Define a loss function similar to that in 48.*

3. *Use a form of gradient descent to iteratively train the neural network u_θ to minimise the loss function.*

4. *Return the trained neural network u_{θ^*}*

3.2 Current Theory

Unlike the classical methods outlined in section 2 there does not exist a general convergence theory underpinning the numerical method. There has however been work done within this field, primarily bounding the generalisation error in the solution. This error will be better defined later but can best be seen as the theoretical error in the approximation. We will see two approaches to bounding this generalisation error, first by Shin, Darbon and Karniadakis [8] and then by Mishra and Molinaro [9,10].

3.2.1 Shin, Darbon and Karniadakis (2020)

We start by defining the generalisation error

$$\varepsilon_g = ||u - u_{\theta^*,m}|| \tag{50}$$

under an appropriate norm (L^2, H^1). $u_{\theta^*,m}$ is the minimiser of the loss function with precisely m data points. The generalisation error can be split into two smaller errors.

Firstly the estimation error which is similar to the error that Cea's Lemma bounds in section 2.2.2 noticing that as the mesh size tends to 0 for a FEM, the approximation becomes the best possible approximation built out of that class of basis functions. The estimation error is given by

$$\varepsilon_E = ||u_{\theta^*,m} - u_{\theta^*,\infty}||$$

again under a suitable norm. Here $u_{\theta^*,\infty}$ is the best possible approximation of u under the class of functions used. With PINNs this amounts to the best possible trained neural network of a specific architecture.

The second error is the approximation error, this is defined by

$$\varepsilon_A = ||u - u_{\theta^*,\infty}||$$

and is the error due to the choice of PINN architecture. Figure 1 visualises each of these errors.

Shin *et al.* show that increasing the number of data m decreases the generalisation error ε_G under two assumptions. This can be written as

$$\varepsilon_G \to 0 \text{ as } m \to \infty.$$

Firstly, the approximated solution is sufficiently smooth. This is controlled via the "Holder regularised loss" which appends regularisation terms to the original PINN loss function punishing large variations in higher derivatives. The second assumption is a space-filling assumption. The authors require that the empirical distribution of training points approximates the full domain. Their methodology here prevents holes and gaps where the BVP is not well sampled.

The authors choose to use two simple one dimensional equations as empirical evidence for their theoretical findings. The first is the one dimensional poisson equation and the second the one dimensional heat equation. In both cases it feels like these examples have been chosen as to not stress the optimisation error and if more complex systems were chosen, the optimisation error would dominate the error in the approximate solution.

3.2.2 Mishra + Molinaro (2020-2021)

Mishra and Molinaro also provide a bound on the generalisation error but in terms of the training error (residuals at collocation points), the number of collocation points and stability estimates in terms of the underlying differential equation. The main result presented is a bound given as

$$\varepsilon_g \leq C_{PDE}\left(\varepsilon_{train} + C_{quad}^{1/p} N^{-\alpha/p}\right) \tag{51}$$

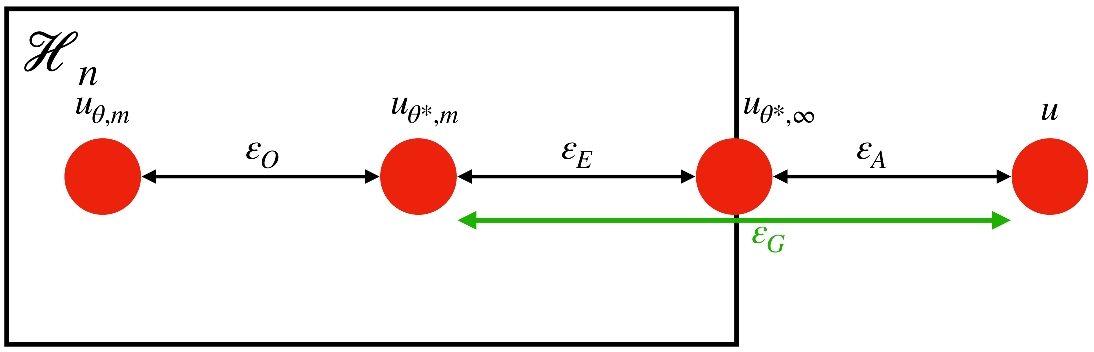


Figure 1: Illustration of the total errors. $\mathcal{H}_n$ is the chosen function class. u is the solution to the underlying PDE. The number of training data is m. $u_{\theta^*,m}$ is a minimiser of the loss with m data. $u_{\theta^*,\infty}$ is a function in $\mathcal{H}_n$ that minimises the loss with infinitely many data. $u_{\theta,m}$ is the approximation that one obtains in practice.

where C_{PDE} is a constant dependant on the stability of the differential equation, C is a constant that is part of an assumed upper bound on the quadrature error. N is the number of quadrature points and $0 < \alpha, 1 \leq p < \infty$ are positive constants.

Much like Shin *et al.*, Mishra and Molinaro also include a space-filling assumption to guarantee no significant gaps in the sampeling, preventing unobserved errors.

Mishra and Molinaro chose more adventurous computational examples including the viscous Burger's equation, an equation with a sharp drop in the interior of the function, and the Taylor Vortex, a fluid dynamics example. For each of these they also provided calculated, problem specific bounds. The paper does however shy away from showing direct plots of errors in the approximation, perhaps again hiding the optimisation error.

A companion paper to the original was also written by Mishra and Molinaro, this time focusing on specific inverse problem applications with a mildly more complex bound on the generalisation error owing to observations on the boundary of the domain.

3.2.3 Grossmann (2024)

Grossmann *et al.* [11] take a different approach to investigating PINNs. Here they look at directly comparing PINNs to FEM over a range of one and two dimensional test problems. When the problems have analytical solutions (in the case of the Poisson equation) the accuracy of the approximations is directly determined. When the ground truth function is not known, a very fine mesh FEM solution will be computed, take as the "ground truth" and compared against lower mesh resolution FEM solutions alongside the PINN solutions. This is the case for both the Allen-Cahn and semi-linear Schrödinger problems. The PINN implemented computationally closely follows the "vanilla" PINN as laid out by Rassi *et al.* [1] and makes use of the commonly used ADAM and L-BFGS optimisers. ADAM is used on a first pass before swapping to the L-BFGS optimiser for fine tuning. The FEM implementations are calculated at a range of mesh resolutions to investigate the tradeoff between resolution and accuracy alongside computation time. The FEM implementations and differ depending on whether the problem is time-dependant or not. In the case of a time-dependant problem, a semi-implicit strategy is used and detailed within the paper. The python package FEniCS is used to implement the FEMs. The authors identify three metrics of interest for comparison: time taken to generate a solution, time taken to evaluate the solution and accuracy of the solution.

With the Poisson equation in one dimension, FEM outclassed the PINN in terms of both computation time and accuracy. Moving to two dimensions saw the PINN draw ahead in computation time but still with a lower accuracy and moving to three dimensions saw the PINN draw far ahead in terms computation time and draw equal to/move ahead of FEM with regards to accuracy of the predicted solution.

The Allen-Cahn equation caused problems for the PINN as the PINN could not predict close to "ground truth" solutions for $\varepsilon \leq 0.001$. For $\varepsilon = 0.01$ in the one dimensional case FEM was many orders of magnitude quicker than the PINN, most likely due to the sheer size of neural network required to approximate the solution well. Moving to two dimensions sees the gap close slightly but not by much.

The semi-linear Schrödinger equation again saw FEM perform better than PINNs in both the one and two dimension case in both computation time and accuracy. PINNs struggled to encapsulate finer details here, even with large neural network sizes.

Overall the paper shows that PINNs still have some way to go before becoming a viable replacement for FEM in many areas however it is worth noting that many FEM programs such as FEniCS have been iterated and improved upon for longer timescales than PINNs have existed for. One final thing to raise is the issue that PINNs had with small epsilon values in the case of the Allen-Cahn equation.

3.2.4 Freirichs-Mihov, Zainelabdeen and John (2023)

This paper [12] explores the use of PINNs for solving convection-dominated convection-diffusion problems. These problems are characterised by sharp gradients and thin layers where the solution changes rapidly. Traditional numerical methods often use Shishkin meshes, layer-adapted grids that cluster points near steep gradients to resolve layers effectively. Inspired by this classical method, the authors propose training PINNs with layer-adapted collocation points and compare the performance against uniformly and randomly distributed collocation points.

The authors employ a hard-constrained PINN formulation to satisfy the boundary conditions exactly. The network representation of the solution is defined as:

$$u_N(x) = g_D(x) + \text{hind}(x) \cdot \tilde{u}_N(x) \tag{52}$$

where $g_D(x)$ is an extension of the dirichlet boundary conditions, hind(x) is an indicator function equal to zero at the boundaries and positive within the domain and $\tilde{u}_N(x)$ is the neural network approximation of the interior solution. This formulations guarantees that the boundary conditions are satisfied regardless of the training process.

The authors use two computational problems to test the collocation point choices. Firstly, a circular internal layer. This is a PDE involving a steep gradient localised to a circular area. Here layer adapted points showed a significant improvement in the accuracy when used alongside custom loss functions as defined by the authors which emphasised learning in critical regions. However, the network struggled to approximate the solution accurately outside the internal layer, indicating overfitting to the high-gradient regions.

The second problem examines a solution with steep gradients concentrated along outflow boundaries. Here, the layer-adapted points performed worse than both uniform and random points. The authors attribute this to the method's excessive clustering in gradient regions, leading to under-sampling in smoother areas, which compromised the global approximation quality.

The authors identify that there is potential with layer-adapted points, especially in the case of resolving internal layers but this does come at the risk of overfitting to sharp regions. In this case the global accuracy is degraded in lieu of the improvements seen in the steep interior regions. The authors recommend further investigation into adaptive techniques that dynamically adjust collocation points based on the evolving error landscape during training.

4 Methodology

We now move onto a study performed investigating PINNs. From both Shin *et al.* [8] and Mishra and Molinaro [9] it is clear that any convergence method for PINNs must somehow bound the approximation one gets in practice alongside the theoretical approximations one could potential learn. This is to bound the optimisation error alongside the generalisation error. To investigate this, a two pronged approach has been taken. One from the theoretical side through the scope of a literature review as seen in sections 2 and 3 investigating both current PINNs theory alongside convergence methods for classical numerical methods. The other approach however concerns the computational implementation of PINNs for specific problems in the hopes of identifying useful and interesting behaviours.

4.1 Motivating Problem: Elliptic BVP

The first problem considered was the one dimensional elliptic BVP given by

$$\begin{aligned} -\varepsilon^2 u''(x) + u(x) &= 1 \\ u(0) = u(1) &= 0 \end{aligned} \tag{53}$$

where $u : [0,1] \to \mathbb{R}$ and $\varepsilon > 0$. This is solvable analytically with solution

$$u(x) = C_1 e^{x/\varepsilon} + C_2 e^{-x/\varepsilon} + 1 \tag{54}$$

with $C_1 = \frac{e^{-1/\varepsilon}-1}{e^{1/\varepsilon}--e^{-1/\varepsilon}}$ and $C_2 = \frac{1-e^{1/\varepsilon}}{e^{1/\varepsilon}-e^{-1/\varepsilon}}$.

This makes for an interesting problem for multiple reasons. Initially, much like the Allen-Cahn equation in Grossmann's study [11] this problem also has an ε term preceding the second derivative. This is likely to cause sharp changes in the solution to the function. Secondly, due to the boundary conditions and the ε^2 term, this problem has steep boundary layers which previous studies have struggled to encapsulate into the solution with PINNs. Thirdly, despite the ε^2, this problem is well defined with a unique solution. This should make it easier for the PINN to find the solution when compared to an ill-condition problem. The problem is also symmetric in nature and defined on a small domain.

4.2 Motivating Problem: Convection Diffusion BVP

The second problem we consider is the convection diffusion problem

$$\begin{aligned} \varepsilon u''(x) + u'(x) &= 1 \\ u(0) = u(1) &= 0 \end{aligned} \tag{55}$$

where again $u : [0,1] \to \mathbb{R}$ and $\varepsilon > 0$. The analytic solution to this is given by

$$u(x) = A(e^{-x/\varepsilon} - 1) + x \tag{56}$$

with $A = \frac{1}{1-e^{-1/\varepsilon}}$. Similar to the elliptic BVP, this problem has a steep boundary layer which may cause the PINN some issues. It is however, not symmetric which may or may not be an issue.

4.3 Pytorch Implementation

The PINNs were implemented using the Pytorch package [13] using a simple two hidden layer architecture. In both cases the original shape of the network was (1,20,1) and made use of $\tanh(x)$ as the activation function in the hopes that it may be able to encapsulate some of the exponentials within the analytic solution. The loss function was very similar to the vanilla loss function proposed by Raissi *et al.* with no inbuilt regularisation terms and the mean square error (MSE) of the residuals at the collocation points contributing the loss. The boundary terms are dealt with separately but also included in the loss function using the MSE.

ADAM was used as the optimiser of choice in both cases due its good performance far from the global minima.

4.4 Computational Experiments

To better understand how the PINNs behave, a variety of computational experiments, altering parameters of the PINN were performed. This initially included:

- Varying the number and distribution of Collocation Points
- Varying the width and depth of the neural network
- Varying the activation function used within the neural network
- Varying the learning rate of the optimiser
- Varying the value of ε
- Varying weighting of the boundary terms within the loss function.

Each of these provides a lever to pull to further understand the machinery of PINNs.

The elliptic problem PINN was then compared to implementations of the problem using FDM, FEM and a collocation method for further comparison.

5 Results

When starting this study, the author expected the implementation of the elliptic problem PINN to be straightforward and to predict a close to ground truth solution fairly quickly. However, this was not the case and the entirety of this results section will be dedicated to documenting the investigation into why this is the case. The convection-diffusion problem remains on the back-burner for the time being as it provided a less interesting problem to explore.

When plotting the first implementation of the PINN for the elliptic BVP (from now on referred to as the elliptic PINN) three phenomena were observed and can be seen in figure 2. Firstly, it is evident that the PINN could not find a good approximate solution to the problem. This is likely due to the second two behaviours. Secondly, the PINN solution exhibits what the author has dubbed a "Gibb's-like" phenomenon referencing the Gibbs phenomena seen when constructing fourier series. However, during the literature review for this report the author came across Runge's phenomena which can affect collocation methods (and is mentioned in section 2.3.3). This may be a more apt description of the behaviour seen but further research is required here.

Thirdly, and most interestingly in the authors opinion, is the fact that we can observe an asymmetry in the predicted solution. Altering the problem domain so it was symmetric about 0 (i.e. acting on the domain [-0.5,0.5]), mirroring the domain about 0, implementing a random order to the calling of the collocation points and passing backwards through the collocation points when training the PINN made no difference to this fact, suggesting it is a behaviour of the PINN itself and not the surrounding "setup" of the problem.

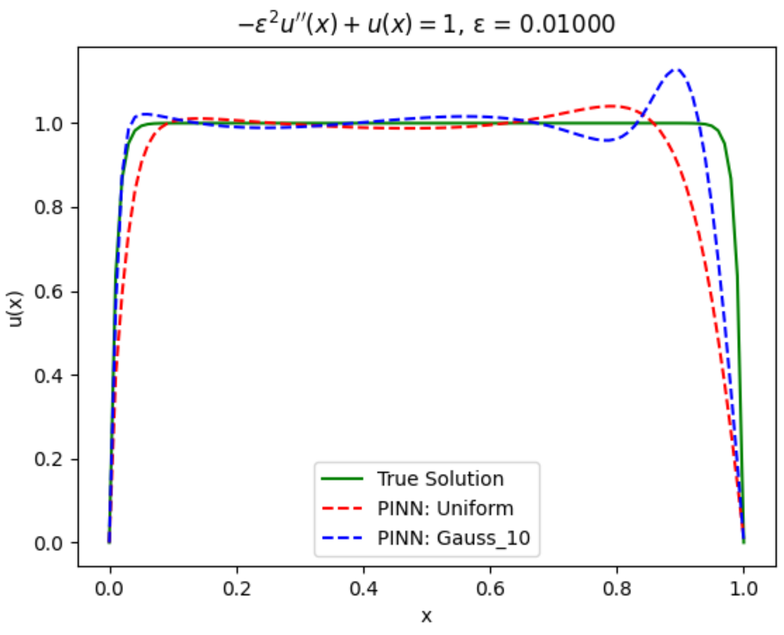


Figure 2: PINN predicted solutions to the Elliptic BVP using uniform and Gauss Legendre collocation points with 10 collocation points each. These are plotted alongside the true solution which makes a symmetric ‘n’ shape.

We note that the elements of the neural network itself should not cause any issues here as for any node within the neural network that generates an asymmetry, there should exist a weight and/or bias to nullify this effect.

One might note here that only 10 collocation points were used to generate the function seen in figure 2. However, varying the number of collocation points and indeed also the distribution of collocation points did not remove the asymmetric behaviour of the solution. It does somewhat improve the accuracy of the approximation, shown in figure 3, but this comes at the cost of a clearly asymmetric function.

5.1 Quantifying Asymmetry

To analyse this asymmetric behaviour further we define a measure of how asymmetric a function is. To do this we note that our functions are defined on an interval $[a, b]$. By mirroring the function and subtracting the mirrored version from the original (and taking the appropriate norm) we can define a measure of how asymmetric a function is. This takes the form of

$$Asym(f) = ||f - \hat{f}|| \tag{57}$$

where $\hat{f}(x) = f(a + b - x)$. In the case of our domain [0,1] we would have $\hat{f}(x) = f(1 - x)$. For a finite number of collocation points as in the case of our PINN we simply make use of the symmetry in the collocations points and can define our measure of asymmetry as

$$asym(u_\theta) = \left(\sum_{i=0}^{N}(u_\theta(x_i) - u_\theta(x_{N-i}))^2\right)^{0.5}. \tag{58}$$

We now turn our attention to varying the parameter ε which gives us the results as shown in figure 4. This provides a fascinating, replicable behaviour which currently seems to have no definitive cause. The shape of the data is reminiscent of a bifurcation curve. However, unlike most bifurcation

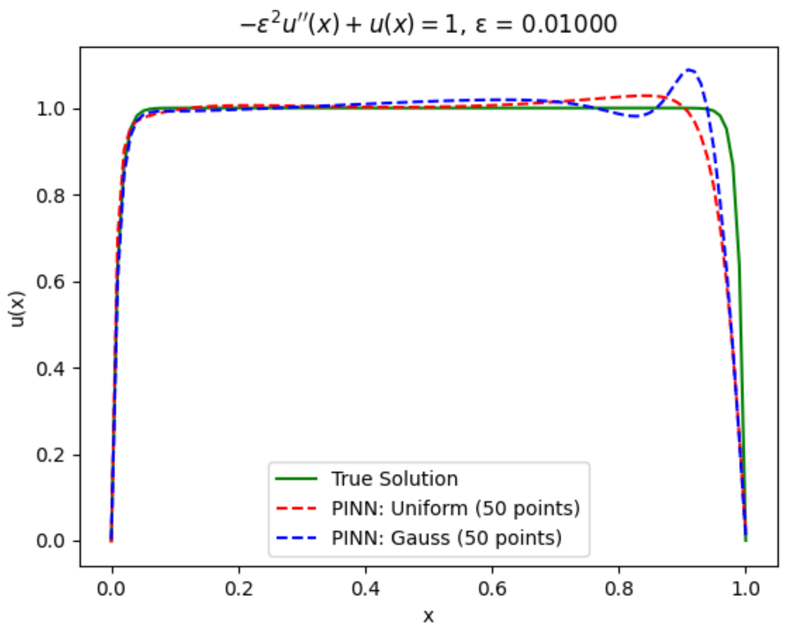


Figure 3: PINN predicted solutions to the Elliptic BVP using uniform and Gauss Legendre collocation points with 50 collocation points each. These are plotted alongside the true solution which makes a symmetric 'n' shape.

curves, there does not yet seem to be model parameter that can control the behaviour of this shape. Varying learning rate, model width and depth, number of collocation points, the weighting of terms within the loss function and collocation method (location of collocation points) all seem to have no effect on the shape of this curve.

5.2 Classical Methods

As a comparison to the PINN, each of the classical methods discussed in section 2 were implemented to solve the elliptic BVP. All three performed well and accurately. Figure 5 shows the implementation of a FDM, note that for $h = 0.1$ the step size is not small enough to accurately predict the boundary behaviour. Figure 6 shows the predicted solution as per the collocation method alongside the distribution of collocation points along the x axis. Figure 7 shows two implementations of FEM to solve the elliptic BVP. The first, figure 7a, implements a uniform mesh over the domain leading to a Gibb's-like phenomenon near the boundary layer. The second, 7b, implements a problem specific mesh which clears up this issue.

It is worth noting that similar to Grossmann *et al.*'s findings, the classical methods all performed better in terms of computational time as well as in terms of accuracy for this one dimensional problem.

6 Future Research Directions

This research project has been an enjoyable experience and has ahead of it many future avenues of research. The potential research directions have been split across two categories depending on the timescale.

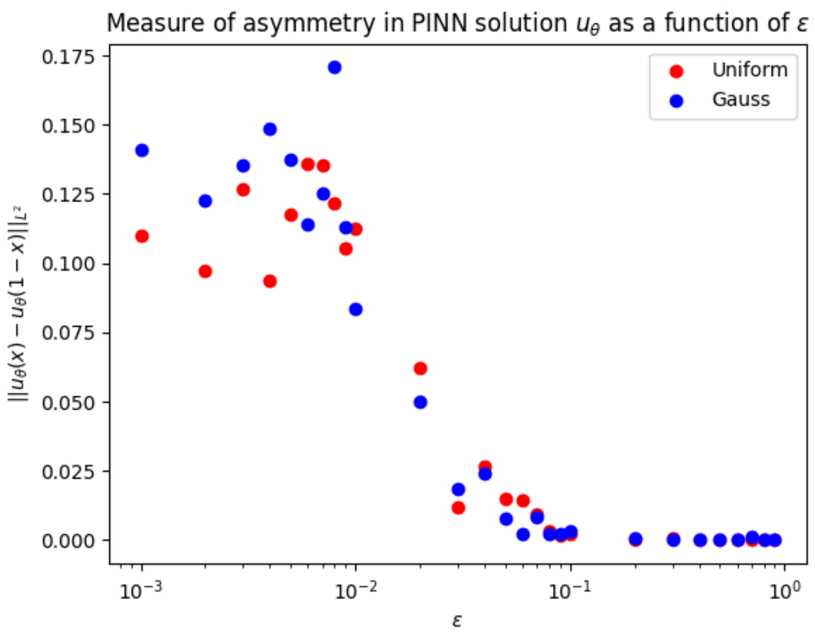


Figure 4: A measure of asymmetry in the PINN predicted solution u_θ as a function of ε as given by 58.

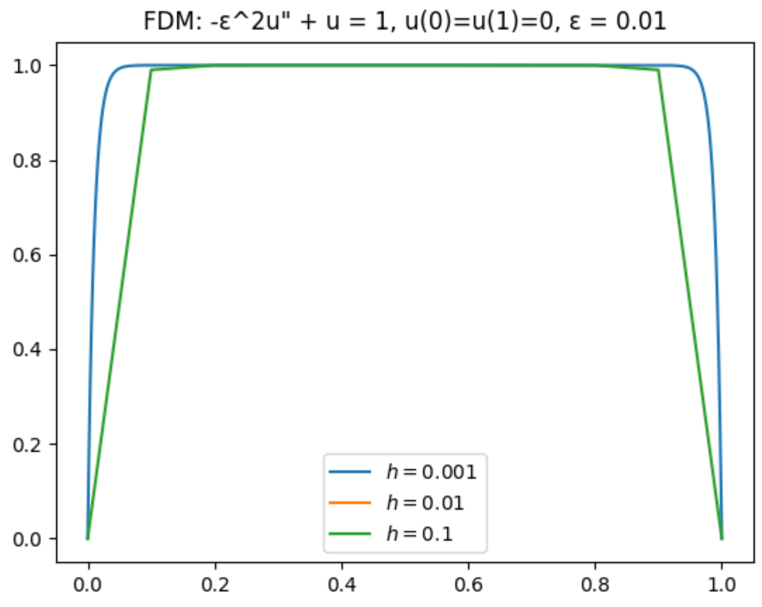


Figure 5: Predicted solutions of the elliptic BVP from FDM implementations with varying step sizes.

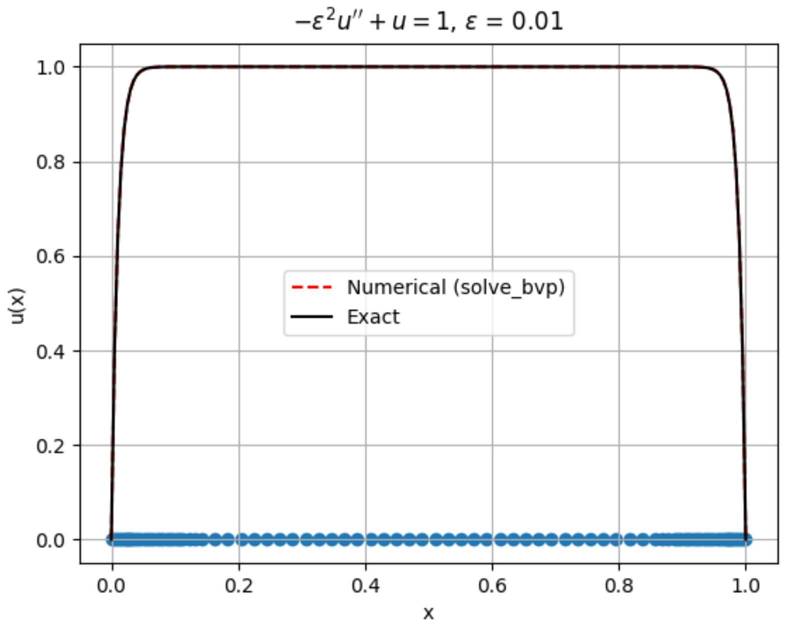


Figure 6: The approximated solution as solved by a collocation method alongside the true solution. The blue dots along the x axis show the distribution of collocation points.

6.1 Short Term

These are projects that the author intends to continue with in the immediate aftermath of this report. They are mostly answering questions raised during the development of this report.

Computational Exploration

Since implementing the methods discussed in sections 4 and 5 other possible causes for the asymmetry seen in the elliptic PINN solutions. These include exploring different optimisers, studying the auto-differentiation and implementation of the PINN using a different ML package (i.e. google JAX). Some of the analysis performed on the elliptic PINN could also be performed on convection-diffusion problem to identify whether or not any issues are problem specific or if they can be replicated in similar problems.

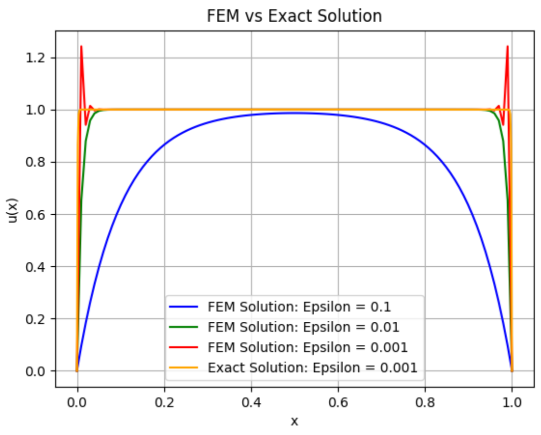
The concern with this is clearly going too far down the rabbit hole chasing this behaviour when it might not manifest itself clearly through more computational approaches. However, this does not mean a further investigation is not warranted and in fact is one of the more likely methods for identifying the cause of the asymmetrical behaviour of the PINN solution.

Implementation of Deep Ritz Method

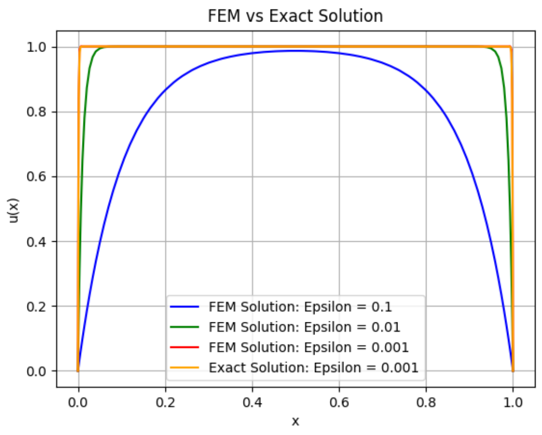
The deep ritz method (DRM) is another modern (neural network based) differential equation solver but instead of relying on the strong formulation of a differential equation, the DRM works using a variational formulation like FEM does. This means it works by minimising an energy functional which is particularly well suited for elliptic PDEs. DRM also optimises the global energy of the differential equation meaning it is less likely to overfit in comparison to the PINN possibly overfitting to the collocation points. This may reduce behaviours such as the Gibbs/Runge phenomena.

Collocation Theory style formulation for PINNs

There are many similarities between PINNs and collocation methods to the point where one could call a PINN a non-linear collocation method with a different numerical method under the hood to



(a) FEM implementation with uniform mesh.



(b) FEM implementation with problem specific mesh.

Figure 7: FEM implementations of the Elliptic BVP.

optimise it (gradient descent vs linear algebra techniques). Studying and using non-linear collocation theory may provide inroads into formulating a bound on the optimisation error for PINNs.

This is likely a larger task and one that is much easier to state than do but it currently represents the best approach that is visible from the theoretical side of this problem.

6.2 Long Term

These goals are broader in nature and look at where the scope of this project could go in the next couple of years. They are not as fleshed out as a result but represent a combination of areas of research interest, likely directions based on the projects current trajectory and potential detours dependant on what occurs.

Bounding the optimisation error

This entire report has been building towards this concept and it seems like the natural capstone for this research. To bound the optimisation error should in theory close the gap and provide a convergence theorem for PINNs alongside the bounding of the generalisation error by both Shin *et al.* and Mishra and Molinaro.

Expansion of theory to other neural network ODE/PDE solvers

Much like PINNs, many other neural network based differential equation solvers lack the theoretical backing that classical methods have. Again, most likely due to their existence within the non-linear world of neural-networks and activation functions in comparison to the classic linear methods. With theory from PINN's this may carry across well into other solving methods such as Neural ODE's, Neural Operators, and the Deep Ritz Method.

Implementation of PINNs for specific problems

This is more of a personal interest of the author with the implementation of PINNs or other neural network based methods within physics problems such as lagrangian and quantum mechanics. These fields often have as much to give back to mathematics as they take and studying and applying new numerical methods to technical and specific problems feels like an interesting approach. This could take the form of the implementation or use of Galerkin quadratures or Symplectic properties in systems in order to adhere to system specific symmetries.

7 Summary

This report/study ultimately acts as a summary for my the last 11 weeks of work. Those weeks have covered a literature review into both classical methods and their convergence theorems and the current convergence theory for PINNs. This includes finite element method, finite difference method, collocation methods and Runge-Kutta schemes. The PINNs theory currently starts and ends with a bound on the generalisation error and requires a bound on the optimisation error if it is to progress further.

The past 3 months have also covered the implementation of many variations of a PINN to solve a simple elliptic BVP and the subsequent discovery that something else is going on causing interesting asymmetries in the PINN predicted solutions alongside another boundary related behaviour. The report ends with scoping out the next steps, continuing on the computational work done here and outlining potential theoretical avenues to make inroads into developing a convergence theorem. It also scopes out longer term research directions potentially after working on a convergence theorem.

References

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

[2] Uri M. Ascher, Robert M. M. Mattheij, and Robert D. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations.* Society for Industrial and Applied Mathematics, 1995.

[3] Mats Larson and Fredrik Bengzon. *The Finite Element Method: Theory, Implementation, and Applications*, volume 10. 01 2013.

[4] Jean Cea. Approximation variationnelle des problèmes aux limites. *Annales de l'Institut Fourier*, 14(2):345–444, 1964.

[5] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, December 1989.

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[7] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[8] Yeonjong Shin Yeonjong Shin, Jérôme Darbon Jérôme Darbon, and George Em Karniadakis George Em Karniadakis. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes. *Communications in Computational Physics*, 28(5):2042–2074, January 2020.

[9] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating pdes. *IMA Journal of Numerical Analysis*, 43(1):1–43, 01 2022.

[10] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for pdes. *IMA Journal of Numerical Analysis*, 42(2):981–1022, 06 2021.

[11] Tamara G Grossmann, Urszula Julia Komorowska, Jonas Latz, and Carola-Bibiane Schönlieb. Can physics-informed neural networks beat the finite element method? *IMA Journal of Applied Mathematics*, 89(1):143–174, 05 2024.

[12] Derk Frerichs-Mihov, Linus Henning, and Volker John. On Loss Functionals for Physics-Informed Neural Networks for Steady-State Convection-Dominated Convection-Diffusion Problems. *Communications on Applied Mathematics and Computation*, August 2024.

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.