KASDI MERBAH
UNIVERSITY - OUARGLA

Faculty of Mathematics and
Material Sciences
Department of Mathematics

MASTER

Path: Mathematics

Speciality: Modeling and Numerical Analysis

Presented by: Guendouz Chaima

Theme

Mathematical Modeling and Approximation of PDEs Problems Using PINNs

Presented in: 10/06/2026

Jury Members:

Dr.	Mzabia Mohamed El Hadi	Prof. Kasdi Merbah University-Ouargla	Chairman
Dr.	Kaliche Keltoum	MCB. Kasdi Merbah University-Ouargla	Examiner
Dr.	Rezzag Bara Rayhana	MCB. Kasdi Merbah University-Ouargla	Supervisor

Dedication

All praise and thanks are due to Allah at the beginning and the end. The journey was neither short nor easy, but by Allah's grace I was able to succeed and reach the finish. I dedicate this achievement first to my ambitious self, and then to everyone who supported me throughout my academic journey and stood by me along the way.

To my mother... you were not just a support; you were a whole homeland when the world felt too tight for me.

To my father... who taught me that determination can create miracles, planted in my heart a love for knowledge, and stood by me in every step of the way.

To my life companion, thank you for your support during the final stage of my journey.

To my two uncles who have passed away and who were my support and strength, I dedicate this work in loyalty to their memory.

To the rest of my family, thank you for being my support and standing by me

throughout this path.

Thanks to everyone who was a helping hand and a source of strength along the way.

I dedicate to you this achievement and the fruit of my success.

Praise be to Allah until You are pleased, praise be to Allah when You are pleased, and praise be to Allah after You are pleased.

Acknowledgement

I begin by praising and thanking Allah, the Ever-Living and Sustainer, first and last, in accordance with the saying of the Prophet Muhammad (peace be upon him):

"Whoever does not thank people does not thank Allah."

I would like to express my deepest gratitude and sincere appreciation to **Dr.Rezzag bara Rayhana**, who kindly accepted to supervise this dissertation and for all her guidance, remarks, and valuable advice.

I also extend our respect and appreciation to the honorable members of the examination committee,to professors **Mzabia Mohammed El Hadi** and **Kaliche Keltoum**, for their effort in reading, reviewing, and evaluating this dissertation.

I further express our sincere thanks to all the professors of the faculty who taught and supported us. Finally,I thank everyone who helped and supported us,whether near or far, and we ask Allah Almighty to place that in the scale of their good deeds. Indeed, He is near and Responsive.

Contents

List of Figures

List of Tables

Abbreviations

- **PDE** : Partial differential equation.
- **ODE** : Ordinary differential equation.
- **FEM** : Finite element methods.
- **FDM** : Finite difference methods.
- **ANN** : Artificial Neural Network.
- **Tanh** : Hyperbolic Tangent Activation Function.
- **ReLU** : Rectified Linear Unit (Activation Function).
- **MAE** : Mean Absolute Error.
- **MSE** : Mean Squared Error.
- **BCE** : Binary cross-entropy.
- **CCE** : Categorical cross-entropy.
- **PINNs** : Physics-Informed Neural Networks.
- **Adam** : Adaptive moment estimation

- **DeepXDE** : Deep learning for X-type Differential Equations

Notations

- Ω: A bounded open set of $\mathbb{R}^n$.
- $(\cdot,\cdot)$: The inner product.
- $L^2(\Omega) = \left\{v : \Omega \to \mathbb{R}; \int_\Omega |v(x)|^2 dx < +\infty\right\}$: The space of square-integrable functions on Ω.
- $H^1(\Omega) = \{v \in L^2(\Omega)\ ;\ \nabla v \in (L^2(\Omega))^2\}$: The Sobolev space H^1.
- $H_0^1(\Omega) = \{v \in H^1(\Omega); v = 0 \text{ on } \partial\Omega\}$: The Sobolev space H_0^1.
- $H^2(\Omega) = \{v \in L^2(\Omega)\ ;\ D^\alpha v \in L^2(\Omega),\ |\alpha| \leq 2\}$: The Sobolev space H^2.
- ρ : The density of the material.
- J : The microscopic displacement inertia .
- μ : The elasticity coefficient.
- δ : The porous stiffness coefficient.
- ξ : The coefficient related to the energy stored in the pores.
- b: The coupling coefficient between u and φ.

- I_n : The n-dimensional unit cube $[0,1]^n$.
- $C(I_n)$: The space of continuous functions on I_n.

Introduction

Partial differential equations PDEs are considered a cornerstone of mathematical modeling. They are widely used in various fields such as physics, biology, and economics to describe many physical phenomena, including heat transfer and wave propagation. In recent years, physics-based problems have undergone a significant shift. In the past, researchers relied on exact solutions [14]; later, classical numerical methods emerged to solve these problems, such as the Finite Element Method FEM and the Finite Difference Method FDM. However, these methods faced difficulties when dealing with high-dimensional problems.

With the advancement of computing power, it became necessary to seek alternative approaches. This led to the emergence of artificial neural networks as a means of representing and approximating functions, supported by the Universal Approximation Theorem [5]. Consequently, a new approach combining physical laws and machine learning appeared, namely Physics-Informed Neural Networks PINNs, which are capable of solving high-dimensional problems without requiring meshes by embedding physical constraints directly into the loss function during the training process [9]. The main problem addressed in this thesis is how to use PINNs to approximate solutions of differential equations and to what extent the obtained solutions are accurate and stable compared to classical numerical methods .Since our objective is to investigate the accuracy and efficiency of PINNs in

solving differential equations, we adopted the one-dimensional poroelasticity system as a fundamental model in this work. This system consists of a set of coupled partial differential equations with viscous damping, which capture the complex dynamic interaction between the mechanical deformation of the material and the variation in the volume of its internal pores.

This work is divided into three main chapters as follows:

- **Chapter 1**: This chapter will be dedicated to a study covering several aspects of partial differential equations. We will highlight their classification and explain the Dirichlet and Neumann conditions. We will also explore numerical methods, leading up to the most important part, which is the proof of the properties of the one-dimensional prous-elastic system and its exponential stability. [2, 4, 6, 12].

- **Chapter 2**: This chapter discusses artificial neural networks (ANNs) and physics-based neural networks, and we highlight the theory of universal approximation in order to understand the theoretical foundations upon which it is based [1,5,9,11,14].

- **Chapter 3**: This chapter represents a continuation of the previous two chapters. We will specify the software tools used and present the numerical results obtained for solving differential equations using PINNs. We will focus on simulating the one-dimensional prous-elastic system and its exponential stability in order to assess the accuracy and efficiency of this modern technique compared to classical numerical methods.

Chapter 1

Generalities on Partial Differential Equations

Many natural, human or biological, chemical, mechanical, economical or financial systems and processes can be described at a macroscopic level by a set of partial differential equations governing averaged quantities such as density, temperature, concentration, velocity, etc [6].

In this chapter, before addressing the one-dimensional poroelastic system, it is necessary to provide a general overview of the differential equation, its classification, and the main associated conditions (Dirichlet and Neumann conditions).

1.1 General introduction to PDE

Recall that a differential equation is an equation for an unknown function of one or several independent variables that relates the value of the function and its derivatives of various orders.

An **ordinary differential equation (ODE)** is a differential equation in which the

unknown function depends on a single independent variable.

A **partial differential equation (PDE)** is a differential equation in which the unknown function

$$u : \Omega \to \mathbb{R}$$

is a function of several independent variables and involves its partial derivatives.

Like ordinary derivatives, a partial derivative is defined as a limit. More precisely, let $\Omega \subset \mathbb{R}^d$ be an open subset and let $u : \Omega \to \mathbb{R}$. The partial derivative of u at $x = (x_1, \ldots, x_d) \in \Omega$ with respect to the variable x_i is defined by

$$\frac{\partial u}{\partial x_i}(x) = \lim_{\Delta x \to 0} \frac{u(x_1, \ldots, x_i + \Delta x, \ldots, x_d) - u(x)}{\Delta x}$$

the function u is said to be **totally differentiable** (or of class C^1) if all its partial derivatives exist in a neighborhood of x and are continuous.

Using the standard notation adopted in this textbook, a typical PDE can be written symbolically as follows.

Let Ω be an open subset of $\mathbb{R}^n$. Given a function

$$F : \mathbb{R}^{n_k} \times \mathbb{R}^{n_{k-1}} \times \cdots \times \mathbb{R}^n \times \mathbb{R} \times \Omega \longrightarrow \mathbb{R},$$

and an unknown function $u : \Omega \to \mathbb{R}$, a general PDE of order k can be written symbolically as

$$F\big(D^k u(x), D^{k-1} u(x), \ldots, Du(x), u(x), x\big) = 0, \qquad x \in \Omega.$$

The particular feature of a partial differential equation (PDE) is that it involves functions of several variables

$$(y, x, \ldots) \longmapsto u(x, y, \ldots).$$

A PDE is therefore a relation between the variables and the partial derivatives of u.

Definition 1.1.1 *: higher-order partial derivatives*

Second-order derivatives are defined recursively. For example:

$$\partial^2_{xx}u = \partial_x(\partial_x u), \qquad \partial^2_{yx}u = \partial_x(\partial_y u).$$

If u is sufficiently smooth (for example C^2), then

$$\partial^2_{xy}u = \partial^2_{yx}u.$$

Definition 1.1.2 *PDE of First and Second Order*

A first-order PDE in two variables has the form:

$$F(x, y, u, \partial_x u, \partial_y u) = 0.$$

A second-order PDE has the form:

$$F(x, y, u, \partial_x u, \partial_y u, \partial^2_{xx}u, \partial^2_{xy}u, \partial^2_{yy}u) = 0.$$

More generally, a PDE may involve derivatives of the form

$$\partial^m_x \partial^n_y u.$$

*The **order** of a PDE is the highest value of $m + n$ appearing in the equation.*

Solving a PDE in a domain $\Omega \subset \mathbb{R}^d$ means finding a sufficiently differentiable function u satisfying the equation in Ω.

1.2 Boundary value problems and associated conditions

Boundary conditions are a key step in the study of any partial differential equation, as the problem cannot be fully described without specifying these conditions, which control the behavior of the solution on the boundary of the domain and ensure a well-posed solution. Starting from the theoretical framework introduced by Arendt and Warma [3], the understanding of the Laplacian operator on the space $L^2(\Omega)$ is based on two fundamental boundary formulations. This depends on whether we control the behavior of the function itself or its outward normal derivatives on the boundary.

1.2.1 Dirichlet conditions

Dirichlet boundary conditions, also referred to as boundary conditions of the first kind, are fundamental components in defining physical and mathematical problems governed by partial differential equations. The primary objective of these conditions is to explicitly prescribe the values of the dependent variables on the boundary of the computational domain.

Definition 1.2.1 *Consider a physical domain $\Omega \subset \mathbb{R}^d$ with a smooth boundary Γ. The Dirichlet boundary condition is imposed by setting the solution u equal to a given function g on the boundary Γ:*

$$u = g \quad on \quad \Gamma \tag{1.1}$$

where $g : \Gamma \to \mathbb{R}$ represents the prescribed boundary data.

1.2.2 Neumann conditions

Neumann boundary conditions, also known as boundary conditions of the second type, specify the value of the outward normal derivative of the solution along the boundary of the domain. According to Arendt and Warma (2003), these conditions can be described as follows:

Definition 1.2.2 *Let Ω be an open subset of $\mathbb{R}^n$. The homogeneous Neumann boundary condition is obtained by imposing that the outward normal derivative of the solution u vanishes on the boundary $\partial\Omega$: For the homogeneous Neumann condition, it is written as:*

$$\frac{\partial u}{\partial n} = 0 \quad on\ \partial\Omega.$$

For the non-homogeneous Neumann condition, it is written as:

$$\frac{\partial u}{\partial n} = g(x) \quad on\ \partial\Omega.$$

where ν represents the unit outward normal vector to the boundary.

As discussed in the reference, the exterior normal derivative may only be defined in a weak sense, especially when the boundary lacks sufficient smoothness for the classical normal derivative to exist. In this framework, the Neumann Laplacian is formulated as a self-adjoint operator on $L^2(\Omega)$. This approach is particularly important for the numerical techniques presented in Chapter 3, since it enables the boundary conditions to be incorporated naturally into the mathematical formulation.

1.2.3 Initial Conditions

According to Walter Strauss, initial conditions specify the physical state at a particular time t_0.

1. **Diffusion and heat equation** For the diffusion equation, the initial condition is:

$$u(\mathbf{x}, t_0) = \phi(\mathbf{x})$$

where $\phi(\mathbf{x})$ is the initial temperature or concentration.

2. **Wave equation** For the wave equation, a pair of initial conditions is required:

$$u(\mathbf{x}, t_0) = \phi(\mathbf{x}) \quad \text{and} \quad \frac{\partial u}{\partial t}(\mathbf{x}, t_0) = \psi(\mathbf{x})$$

where $\phi(\mathbf{x})$ is the initial position and $\psi(\mathbf{x})$ is the initial velocity.

1.3 Classical methods for solving PDEs

Partial differential equations form the basis of mathematical modeling, however, due to the increasing complexity of problems, it has become difficult to obtain exact solutions. This has led to the emergence of alternative numerical methods.

1.3.1 Analytical methods

Analytical method is a set of logical steps followed to solve a specific problem to find an accurate solution for it. This solution can be expressed using known mathematical

functions and formulas. It allows for obtaining highly accurate solutions, which usually take longer to find, and its use lies in finding other solutions.

Example: Consider the following linear first-order differential equation:

$$\frac{du}{dt} = \lambda u$$

where λ is a constant. By separating the variables, we get:

$$\frac{du}{u} = \lambda \, dt$$

Integrating both sides:

$$\ln |u| = \lambda t + C_0$$

where C_0 is the constant of integration. Consequently, the general solution is expressed as:

$$u(t) = Ce^{\lambda t}$$

where $C = \pm e^{C_0}$ is an arbitrary constant.

1.3.2 Finite difference method

The finite difference method (FDM) is a numerical method used to solve differential equations by approximating derivatives with difference formulas on a discrete grid.

Consider the general partial differential equation

$$F\left(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial y^2}\right) = 0$$

defined on a domain Ω with suitable boundary conditions.

The domain is discretized into grid points

$$x_i = ih, \qquad y_j = jk$$

where h and k are the step sizes in the x- and y-directions, respectively.

The derivatives are approximated using finite differences. For example,

$$\frac{\partial u}{\partial x}(x_i, y_j) \approx \frac{u_{i+1,j} - u_{i,j}}{h}$$

and

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}.$$

Thus, the differential equation is transformed into a system of algebraic equations that can be solved numerically.

1.3.3 Finite element methods

The finite element method (FEM) is a numerical technique widely used for solving partial differential equations, particularly elliptic and parabolic problems. The main idea of FEM is to transform the differential equation from its strong form into a weak form and then approximate the solution in a finite-dimensional space.

Consider a general linear partial differential equation in a bounded domain $\Omega \subset \mathbb{R}^d$:

$$\mathcal{A}u = f \quad \text{in } \Omega$$

with suitable homogeneous boundary conditions on $\partial\Omega$, where $\mathcal{A}$ is a linear differential operator and $f \in L^2(\Omega)$.

To derive the weak formulation, we introduce an appropriate Hilbert space V incorporating the boundary conditions. We multiply the equation by a test function $v \in V$ and integrate over the domain Ω:

$$\int_\Omega (\mathcal{A}u)v \, dx = \int_\Omega fv \, dx$$

By applying Green's formula or integration by parts to shift the derivatives onto the test function v, we define a bilinear form $a(\cdot, \cdot)$ and a linear functional $L(\cdot)$. Hence, the abstract weak formulation is written as: Find $u \in V$ such that

$$a(u, v) = L(v), \quad \forall v \in V \tag{1.2}$$

where the continuous bilinear form $a : V \times V \to \mathbb{R}$ captures the differential operator's action, and the linear functional $L : V \to \mathbb{R}$ is defined by:

$$L(v) = \int_\Omega f v \, dx$$

The computational domain is then divided into finite elements, and the solution is approximated in a finite-dimensional subspace $V_h \subset V$. In the Galerkin approach, the approximate solution and the test functions are chosen from the same finite-dimensional space V_h. Therefore, the discrete problem consists of finding $u_h \in V_h$ such that:

$$a(u_h, v_h) = L(v_h), \quad \forall v_h \in V_h$$

As a result, the continuous differential problem is transformed into a system of algebraic equations that can be solved numerically.

1.4 One-Dimensional Porous-Elastic System

In this section,we investigate the one-dimensional porous-elastic system, which has attracted significant attention from numerous researchers [2]. This system represents a mathematical model for elastic solids with internal voids and pores, combining two fundamental effects: the classical elastic effect and the microstructure. The evolution of this system is governed by a system of coupled partial differential equations:

$$\begin{cases} \rho u_{tt} - \mu u_{xx} - b\phi_x = 0 & \text{in } (0,1) \times (0,\infty), \\ J\phi_{tt} - \delta\phi_{xx} + bu_x + \xi\phi + \tau\phi_t = 0 & \text{in } (0,1) \times (0,\infty), \end{cases} \tag{1.3}$$

Initial conditions:

$$u(x,0) = u_0(x), u_t(x,0) = u_1(x), \phi(x,0) = \phi_0(x), \phi_t(x,0) = \phi_1(x) \tag{1.4}$$

Boundary conditions (Dirichlet-Dirichlet):

$$u(0,t) = u(1,t) = \phi(0,t) = \phi(1,t) = 0 \ \ \forall t > 0 \tag{1.5}$$

The physical system is defined by several variables and constants. The variables include the function $u(x,t)$, which represents the displacement of the elastic solid material, and

the function $\phi(x,t)$, which represents the volumetric fraction of voids and internal pores, controlling the porosity of the medium.

Regarding the physical constants, ρ denotes the density of the solid material, while J represents the microstructural equilibrium parameter of the voids; both are positive quantities. The parameters μ, b, δ, ξ, and τ are positive constitutive constants. The coefficient b represents the coupling between displacement and porosity, and these constants satisfy the condition $\mu\xi > b^2$. And τ represents the porosity viscosity coefficient.

We will first prove the well-posedness of the problem, and then study the exponential stability in order to establish the exponential decay.

1.4.1 The well-posedness

In this subsection, we prove the existence, uniqueness and smoothness of solutions for the system (1.3)-(1.4)-)(1.5).Introducing the vector function $U = (u, v, \phi, \psi)^T$, where $v = u_t, \psi = \phi_t$ System (1.3)-(1.4)-(1.5) can be written as

$$\begin{cases} U_t & = \mathcal{A}U \\ U(0) & = U_0 = (u_0, u_1, \phi_0, \phi_1) \end{cases} \tag{1.6}$$

and the operator $\mathcal{A}$ is defined by

$$\mathcal{A}\begin{pmatrix} u \\ v \\ \phi \\ \psi \end{pmatrix} = \begin{pmatrix} v \\ \frac{1}{\rho}(\mu u_{xx} + b\phi_x) \\ \psi \\ \frac{1}{J}(\delta\phi_{xx} - bu_x - \xi\phi - \tau\psi) \end{pmatrix}, \tag{1.7}$$

We consider the energy space

$$\mathcal{H} = H_0^1(0,1) \times L^2(0,1) \times H_0^1(0,1) \times L^2(0,1),$$

endowed with the inner product defined for $U = (u, v, \phi, \psi)$ and $\widetilde{U} = (\tilde{u}, \tilde{v}, \tilde{\phi}, \tilde{\psi})$ by

$$\langle U, \widetilde{U}\rangle_{\mathcal{H}} = \int_0^1 \Big(\rho v\tilde{v} + J\psi\tilde{\psi} + \mu u_x\tilde{u}_x + \delta\phi_x\tilde{\phi}_x + \xi\phi\tilde{\phi} + b(u_x\tilde{\phi} + \tilde{u}_x\phi)\Big)\, dx.$$

Under the hypothesis $\mu\xi > b^2$, it is easy to see that $(U.\tilde{U})_{\mathcal{H}}$defines an inner product.

$$\|U\|_{\mathcal{H}} = \langle U, U\rangle_{\mathcal{H}} = \rho\int_0^1 v^2dx + J\int_0^1 \psi^2dx + \delta\int_0^1 \phi_x^2dx + \left(\xi - \frac{b^2}{\mu}\right)\int_0^1 \phi^2 + \mu\int_0^1 \left(u_x + \frac{b}{\mu}\phi_x\right)^2 dx.$$

The domain of $\mathcal{A}$ is given by

$$D(\mathcal{A}) = \Big\{(u, v, \phi, \psi) \in \mathcal{H} : u, \phi \in H^2(0,1) \cap H_0^1(0,1),\ v, \psi \in H_0^1(0,1)\Big\}.$$

Theorem 1.4.1 *Let $U_0 \in \mathcal{H}$. Then, there exists a unique solution $U \in C(R^+, \mathcal{H})$ of problem (1.3)-(1.4)-(1.5). Moreover, if $U_0 \in D(\mathcal{A})$, then $U \in C(R^+, D(\mathcal{A})) \cap C^1(R^+, \mathcal{H})$.*

Proof The result follows from Lumer-Phillips theorem provided we prove that $\mathcal{A}$ is a maximal dissipative operator, that $\mathcal{A}$ is dissipative and that $(I - \mathcal{A})$ is surjective. Thus, for any $U \in D(\mathcal{A})$, we compute

$$\langle \mathcal{A}U, U\rangle_{\mathcal{H}}.$$

To prove that the operator $\mathcal{A}$ is dissipative, i.e., to verify the following inequality:

$$\langle \mathcal{A}U, U\rangle_{\mathcal{H}} = -\tau \int_0^1 \phi_t^2\, dx \leq 0$$

Let $U = (u, v, \phi, \psi)^T \in D(\mathcal{A})$. From the definitions given in the system, we have:

$$v = u_t, \quad \psi = \phi_t$$

Using the definition of the operator $\mathcal{A}$ from equation(1.7) and $\mathcal{A}U = (\tilde{u}, \tilde{v}, \tilde{\phi}, \tilde{\psi})^T$ where:

$$\tilde{u} = v, \quad \tilde{v} = \frac{1}{\rho}(\mu u_{xx} + b\phi_x), \quad \tilde{\phi} = \psi, \quad \tilde{\psi} = \frac{1}{J}(\delta\phi_{xx} - bu_x - \xi\phi - \tau\psi)$$

The inner product

$$\langle \mathcal{A}U, U\rangle_{\mathcal{H}} = \int_0^1 \Big(\rho\tilde{v}v + J\tilde{\psi}\psi + \mu\tilde{u}_x u_x + \delta\tilde{\phi}_x\phi_x + \xi\tilde{\phi}\phi + b(\tilde{u}_x\phi + u_x\tilde{\phi})\Big)\, dx$$

By substituting the components:

$$\langle \mathcal{A}U, U\rangle_{\mathcal{H}} = \int_0^1 [(\mu u_{xx} + b\phi_x)v + (\delta\phi_{xx} - bu_x - \xi\phi - \tau\psi)\psi + \mu v_x u_x + \delta\psi_x\phi_x + \xi\psi\phi + b(v_x\phi + u_x\psi)] \tag{1.8}$$

By expanding the expression (1.8)and simplifying it, we obtain:

$$\langle \mathcal{A}U, U\rangle_{\mathcal{H}} = \int_0^1 \Big[\mu(u_{xx}v + u_x v_x) + \delta(\phi_{xx}\psi + \phi_x\psi_x) + b(\phi_x v + v_x\phi) - bu_x\psi + bu_x\psi - \xi\phi\psi + \xi\phi\psi - \tau\psi^2\Big]\, dx$$

$$\langle \mathcal{A}U, U\rangle_{\mathcal{H}} = \int_0^1 \left(\mu(u_{xx}v + u_x v_x) + \delta(\phi_{xx}\psi + \phi_x\psi_x) + b(\phi_x v + v_x\phi) - \tau\psi^2\right) dx$$

Using integration by parts and the boundary conditions $u = \phi = 0$ on $\{0, 1\}$, we obtain

$$\int_0^1 \mu u_{xx} v\, dx + \int_0^1 \delta\phi_{xx}\psi\, dx + \int_0^1 b v_x \phi\, dx$$

$$= [[\mu u_x v]_0^1 - \int_0^1 \mu u_x v_x\, dx] + [[\delta\phi_x\psi]_0^1 - \int_0^1 \delta\phi_x\psi_x\, dx] + [[bv\phi]_0^1 - \int_0^1 bv\phi_x\, dx]$$

$$= -\int_0^1 \mu u_x v_x\, dx - \int_0^1 \delta\phi_x\psi_x\, dx - \int_0^1 bv\phi_x\, dx$$

Thus: $\int_0^1 \mu(u_{xx}v + u_x v_x)\, dx = \int_0^1 \delta(\phi_{xx}\psi + \phi_x\psi_x)\, dx = \int_0^1 b(\phi_x v + v_x\phi)\, dx = 0$

$$\langle \mathcal{A}U, U\rangle_{\mathcal{H}} = -\tau \int_0^1 \phi_t^2 dx \leq 0.$$

$$\langle \mathcal{A}U, U\rangle_{\mathcal{H}} = -\tau \int_0^1 \psi^2\, dx$$

Since $\psi = \phi_t$, we arrive at the final required result:

$$\langle \mathcal{A}U, U\rangle_{\mathcal{H}} = -\tau \int_0^1 \phi_t^2\, dx$$

Given that the parameter $\tau > 0$ and the squared term under the integral is always non-negative ($\phi_t^2 \geq 0$), it follows that:

$$\langle \mathcal{A}U, U\rangle_{\mathcal{H}} \leq 0$$

Thus, $\mathcal{A}$ is dissipative. Next, we prove that the operator $(I - \mathcal{A})$ is surjective. $G = (g_1, g_2, g_3, g_4) \in \mathcal{H}$. We solve

$$(I - \mathcal{A})U = G.$$

This leads to an elliptic system:

$$\begin{cases} u - v = g_1, \\ v - \frac{1}{\rho}(\mu u_{xx} + b\phi_x) = g_2, \\ \phi - \psi = g_3, \\ \psi - \frac{1}{J}(\delta\phi_{xx} - bu_x - \xi\phi - \tau\psi) = g_4. \end{cases}$$

Eliminating v and w, we obtain a coercive elliptic system in (u, ϕ). Using Lax-Milgram theorem and the condition $\mu\xi > b^2$, we obtain existence and uniqueness of (u, ϕ), hence of U. Finally, using Lumer-Phillips theorem we deduce that $\mathcal{A}$ is an infinitesimal generator of a contraction semigroup in $\mathcal{H}$. Hence the system is well-posed and this complete the proof. ■

1.4.2 Exponential stability

In this subection, we study stability of the system (1.3) under the initial and boundary conditions given in (1.4) and (1.5).

Lemma 1.4.2 *[2] Let (u, ϕ) be the solution of(1.3)–(1.4) with(1.5). Then the energy functional E, defined by*

$$E(t) = \frac{1}{2}\int_0^1 \left[\rho u_t^2 + \mu u_x^2 + J\phi_t^2 + \delta\phi_x^2 + \xi\phi^2 + 2bu_x\phi\right] dx, \tag{1.9}$$

satisfies

$$E'(t) = -\tau\int_0^1 \phi_t^2 dx \leq 0. \tag{1.10}$$

Proof [2] Equation(1.10) follows by multiplying the first and the second equations of (1.3) by u_t and ϕ_t:

$$\rho u_{tt}u_t - \mu u_{xx}u_t - b\phi_x u_t = 0 \tag{1.11}$$

$$J\phi_{tt}\phi_t - \delta\phi_{xx}\phi_t + \xi\phi\phi_t + bu_x\phi_t + \tau\phi_t^2 = 0 \tag{1.12}$$

By using the identities $v_{tt}v_t = \frac{1}{2}\frac{d}{dt}(v_t^2)$ and $vv_t = \frac{1}{2}\frac{d}{dt}(v^2)$, equations (1.11) and (1.12) can be rewritten respectively as:

$$\frac{1}{2}\rho\frac{d}{dt}(u_t^2) - \mu u_{xx}u_t - b\phi_x u_t = 0$$

$$\frac{1}{2}J\frac{d}{dt}(\phi_t^2) - \delta\phi_{xx}\phi_t + \frac{1}{2}\xi\frac{d}{dt}(\phi^2) + bu_x\phi_t + \tau\phi_t^2 = 0$$

Then, we integrate both equations by parts over the interval $(0, 1)$ with respect to dx. For the first equation, applying integration by parts to the term $\int_0^1 \mu u_{xx}u_t\, dx$ by choosing $f = u_t$ and $g' = \mu u_{xx}$ yields:

$$-\int_0^1 \mu u_{xx}u_t\, dx = -\left[\mu u_x u_t\right]_0^1 + \int_0^1 \mu u_x u_{tx}\, dx$$

Using the homogeneous boundary conditions(1.5) , we have $u_t(0,t) = u_t(1,t) = 0$, hence the boundary term vanishes: $[\mu u_x u_t]_0^1 = 0$. Since $u_x u_{tx} = \frac{1}{2}\frac{d}{dt}(u_x^2)$, we obtain:

$$\frac{1}{2}\frac{d}{dt}\int_0^1 \rho u_t^2\,dx + \frac{1}{2}\frac{d}{dt}\int_0^1 \mu u_x^2\,dx - \int_0^1 b\phi_x u_t\,dx = 0 \tag{1.13}$$

For the second equation, applying integration by parts to the term $\int_0^1 \delta\phi_{xx}\phi_t\,dx$ by choosing $f = \phi_t$ and $g' = \delta\phi_{xx}$ gives:

$$-\int_0^1 \delta\phi_{xx}\phi_t\,dx = -\left[\delta\phi_x\phi_t\right]_0^1 + \int_0^1 \delta\phi_x\phi_{tx}\,dx$$

Using the boundary conditions (1.5), the boundary term vanishes: $[\delta\phi_x\phi_t]_0^1 = 0$. Since $\phi_x\phi_{tx} = \frac{1}{2}\frac{d}{dt}(\phi_x^2)$, it yields:

$$\frac{1}{2}\frac{d}{dt}\int_0^1 J\phi_t^2\,dx + \frac{1}{2}\frac{d}{dt}\int_0^1 \delta\phi_x^2\,dx + \frac{1}{2}\frac{d}{dt}\int_0^1 \xi\phi^2\,dx + \int_0^1 bu_x\phi_t\,dx + \int_0^1 \tau\phi_t^2\,dx = 0 \tag{1.14}$$

Summing (1.13) and (1.14), the remaining coupling terms are:

$$-\int_0^1 b\phi_x u_t\,dx + \int_0^1 bu_x\phi_t\,dx$$

By performing integration by parts once on the first coupling term, we obtain $-\int_0^1 b\phi_x u_t\,dx = \int_0^1 b\phi u_{xt}\,dx$. Combining this with the second coupling term gives:

$$\int_0^1 b\left(\phi u_{xt} + u_x\phi_t\right)dx = \frac{d}{dt}\int_0^1 bu_x\phi\,dx = \frac{1}{2}\frac{d}{dt}\int_0^1 2bu_x\phi\,dx$$

Finally, by grouping all the time derivative terms together under $\frac{d}{dt}$, we get:

$$\frac{d}{dt}\left(\frac{1}{2}\int_0^1 \left[\rho u_t^2 + \mu u_x^2 + J\phi_t^2 + \delta\phi_x^2 + \xi\phi^2 + 2bu_x\phi\right]dx\right) = -\tau\int_0^1 \phi_t^2\,dx$$

Thus, we obtain the energy equation $E(t)$ in (1.9), which indicates that:

$E'(t) = -\tau\int_0^1 \phi_t^2\,dx \le 0$. ■

Lemma 1.4.3 *[2] Let (u,ϕ) be the solution of (1.3)–(1.4) with (1.5). Then the functional*

$$G_1(t) := J\int_0^1 \phi_t\phi dx + \frac{\tau}{2}\int_0^1 \phi^2 dx$$

satisfies

$$G_1'(t) = -\delta\int_0^1 \phi_x^2 dx - \xi\int_0^1 \phi^2 dx - b\int_0^1 u_x\phi dx + J\int_0^1 \phi_t^2 dx. \tag{1.15}$$

Lemma 1.4.4 *[2]*

$$\frac{\mu}{\rho} = \frac{\delta}{J}. \tag{1.16}$$

Assume (1.16)holds and let (u, ϕ) be the solution of (1.3)–(1.4) with(1.5). Then the functional

$$G_2(t) := \int_0^1 \phi_x u_t dx + \int_0^1 u_x \phi_t dx$$

satisfies

$$G_2'(t) \leq -\frac{b}{2J}\int_0^1 u_x^2 dx - \frac{\xi}{J}\int_0^1 u_x \phi dx + c\int_0^1 \phi_x^2 dx + c\int_0^1 \phi_t^2 dx + \frac{\delta}{J} u_x \phi_x \Big|_0^1. \tag{1.17}$$

In consideration of the boundary terms that appear in (1.17), we define the function $m(x) = 2 - 4x$, $x \in [0; 1]$. Consequently, we have the following result:

Lemma 1.4.5 *[2] Let (u, ϕ) be the solution of (1.3)–(1.4) with(1.5). Then the functional*

$$G_3(t) := \frac{\rho\delta\varepsilon_1}{J\mu}\int_0^1 m(x) u_t u_x dx + \frac{1}{4\varepsilon_1}\int_0^1 m(x)\phi_t \phi_x dx \tag{1.18}$$

satisfies, for any $\varepsilon_1, \varepsilon_2 > 0$, the estimate

$$\begin{aligned} G_3'(t) \leq & -\frac{\delta}{J} u_x \phi_x \Big|_0^1 - \frac{\xi}{2J\varepsilon_1}\int_0^1 \phi^2 dx + (c\varepsilon_1 + \varepsilon_2)\int_0^1 u_x^2 dx + c\varepsilon_1 \int_0^1 u_t^2 dx \\ & + \frac{c}{\varepsilon_1}\int_0^1 \phi_t^2 dx + c\left(\varepsilon_1 + \frac{1}{\varepsilon_1} + \frac{1}{\varepsilon_1^2 \varepsilon_2}\right)\int_0^1 \phi_x^2 dx. \end{aligned} \tag{1.19}$$

Lemma 1.4.6 *[2] Let (u, ϕ) be the solution of (1.3)–(1.4) with(1.5). Then the functional*

$$G_4(t) := -\rho\int_0^1 u_t u dx \tag{1.20}$$

satisfies

$$G_4'(t) = -\rho\int_0^1 u_t^2 dx + \mu\int_0^1 u_x^2 dx + b\int_0^1 u_x \phi dx. \tag{1.21}$$

Theorem 1.4.7 *[2] Assume (1.16) holds and let (u, ϕ) be the solution of (1.3)–(1.4) with(1.5). Then there exist positive constants a and d for which the energy functional(1.9)satisfies*

$$E(t) \leq a e^{-dt}, \quad t \geq 0. \tag{1.22}$$

Proof [2] We define a Lyapunov functional

$$\mathcal{L}(t) := NE(t) + N_1 G_1(t) + N_2(G_2(t) + G_3(t)) + G_4, \tag{1.23}$$

where N, N_1, and N_2 are positive constants to be properly chosen later. By differentiating (1.23), recalling (1.10)–(1.21), (1.21), and setting

$$\varepsilon_1 = \frac{\rho}{2cN_2}, \quad \varepsilon_2 = \frac{b}{4J},$$

we obtain

$$\begin{aligned}\mathcal{L}'(t) \leq & -\left[\tau N - JN_1 - cN_2(N_1+N_2)\right]\int_0^1 \phi_t^2 dx - \xi\left[N_1 + \frac{cN_2^2}{J\rho}\right]\int_0^1 \phi^2 dx \\ & - \frac{\rho}{2}\int_0^1 u_t^2 dx - \left[\delta N_1 - cN_2\left(1 + N_2 + N_2^2 + \frac{1}{N_2}\right)\right]\int_0^1 \phi_x^2 dx \\ & - \left[\frac{b}{4J}N_2 - \frac{\rho}{2} - \mu\right]\int_0^1 u_x^2 dx - \left[bN_1 + \frac{\xi N_2}{J} - b\right]\int_0^1 u_x\phi dx.\end{aligned}$$

We choose N_2 large enough such that

$$\alpha_1 = \frac{b}{4J}N_2 - \frac{\rho}{2} - \mu > 0,$$

then we choose N_1 large enough such that

$$\alpha_2 = \delta N_1 - cN_2\left(1 + N_2 + N_2^2 + \frac{1}{N_2}\right) > 0 \quad \text{and} \quad \alpha_3 = bN_1 + \frac{\xi N_2}{J} - b > 0.$$

So, we arrive at

$$\begin{aligned}\mathcal{L}'(t) \leq & -\left[\tau N - c\right]\int_0^1 \phi_t^2 dx - \alpha_1\int_0^1 u_x^2 dx - \alpha_2\int_0^1 \phi_x^2 dx - \alpha_3\int_0^1 u_x\phi dx \\ & - \alpha_4\int_0^1 \phi^2 dx - \frac{\rho}{2}\int_0^1 u_t^2 dx.\end{aligned} \tag{1.24}$$

where $\alpha_4 = \xi N_1 + \frac{c\xi N_2^2}{J\rho}$.

On the other hand, if we let $\mathscr{L}(t) = N_1 G_1(t) + N_2(G_2(t) + G_3(t)) + G_4(t)$, then

$$\begin{aligned}|\mathscr{L}(t)| \leq & N_1\int_0^1 \left|J\phi_t\phi + \frac{\tau}{2}\phi^2\right| dx + N_2\int_0^1 |\phi_x u_t + u_x\phi_t| dx \\ & + N_2\int_0^1 \left|\frac{\rho\delta\varepsilon_1}{J\mu}m(x)u_t u_x + \frac{1}{4\varepsilon_1}m(x)\phi_t\phi_x\right| dx + \rho\int_0^1 |u_t u| dx.\end{aligned}$$

Exploiting Young's and Poincaré's inequalities, we obtain

$$|\mathscr{L}(t)| \leq c \int_0^1 \left(u_t^2 + \phi_t^2 + \phi_x^2 + (u_x + \phi)^2\right) dx \leq cE(t).$$

Consequently, we obtain

$$|\mathcal{L}(t) - NE(t)| \leq cE(t),$$

that is

$$(N - c)E(t) \leq \mathcal{L}(t) \leq (N + c)E(t). \tag{1.25}$$

Now, by choosing N large enough such that

$$\tau N - c > 0, \quad N - c > 0,$$

and exploiting (1.9), estimates (1.4.5) and (1.25), respectively, give

$$\mathcal{L}'(t) \leq -k_1 E(t), \quad \forall t \geq 0 \tag{1.26}$$

and

$$c_1 E(t) \leq \mathcal{L}(t) \leq c_2 E(t), \quad \forall t \geq 0, \tag{1.27}$$

for some $k_1, c_1, c_2 > 0$.

A combination of (1.26) and (1.27), gives

$$\mathcal{L}'(t) \leq -d\mathcal{L}(t), \quad \forall t \geq 0, \tag{1.28}$$

where $d = \frac{k_1}{c_2}$. A simple integration of (1.28) over $(0, t)$ yields

$$\mathcal{L}(t) \leq \mathcal{L}(0)e^{-dt}, \quad \forall t \geq 0. \tag{1.29}$$

Further use of (1.27) gives (1.22) with $a = \frac{c_2 E(0)}{c_1}$. ■

Chapter 2

Fundamentals of ANN and PINNs

In this chapter, we will take a closer look at the fundamental concepts of neural networks in general, with particular focus on Physics-Informed Neural Networks PINNs. We will also highlight the approximation by superpositions of a sigmoidal function and the transition from the special case to the generalized case.

2.1 Artificial Neural Networks (ANN)

2.1.1 Introduction

The history of artificial neural networks is rich with distinguished individuals from various fields striving to develop the concepts. The modern perspective on neural networks began around 1940 with the work of Warren McCulloch and Walter Pitts, who demonstrated that neural networks could compute any mathematical or logical function.

In the early 1950s, with the invention of the Perceptron by Frank Rosenblatt, it became possible to perform pattern recognition. However, interest in neural networks de-

clined during the 1960s due to a lack of new ideas and the limited computing power of available computers for experimentation. In the 1980s, research in neural networks gained momentum again as computer capabilities grew rapidly and new concepts were introduced. Neural networks have clearly taken a permanent place as important mathematical/engineering tools. They don't provide solutions to every problem, but they are essential tools to be used in appropriate situations.

2.1.2 Definition of Artificial Neural Networks

Artificial Neural Networks ANN are computational models that attempt to imitate the way biological neural networks operate in the human brain. They can also be used to solve some difficult problems, particularly in finding solutions to partial differential equations PDEs.

They are also used to approximate certain functions, and we will see that their training involves approximating solutions for some partial differential equations.

2.1.3 Architecture of Neural Networks

The topology of a network refers to the way neurons are connected to each other in order to form the overall neural network. In general, an artificial neural network consists of three main layers:

- **Input layer:** Receiving raw data from the external environment.
- **Hidden layer :** processing data.
- **Output layer:** Conclusion of the final result. Each artificial neuron determines whether it will pass a signal to the subsequent neurons by aggregating the information or not.

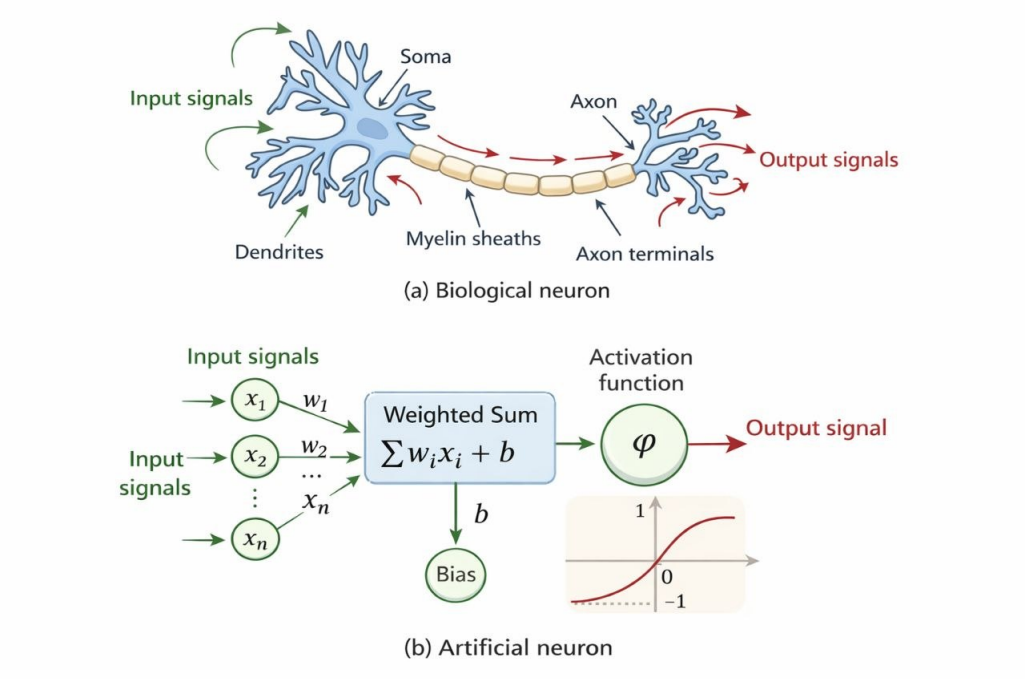

Figure 2.1: **Artificial Neural Networks Struction**

Weights

An ANN receives multiple inputs, but it does not treat all inputs with the same importance. Here, weights come into play: they are numerical values assigned to each input in the neural network, determining the significance of each input in making the final decision. The effect of inputs varies; some have a greater impact on the outcome than others.

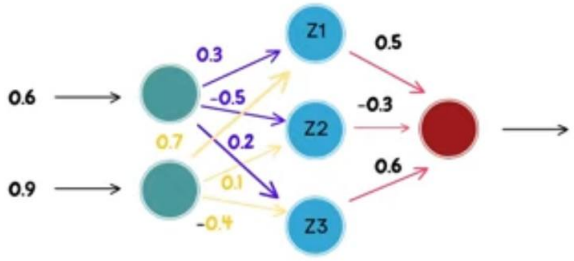

Figure 2.2: **Weights**

Bias

During the learning process, the neural network adjusts the weights to improve prediction accuracy, reducing the less important weights and increasing the more important ones. In some cases, relying solely on weighted inputs may not be sufficient to make a decision. Here, the bias comes into play: it is an additional auxiliary input that helps adjust the activation in a neuron, allowing it to make decisions even when the inputs are zero, and it helps detect subtle patterns.

$$z = \sum_{i=1}^{n} w_i x_i + b$$

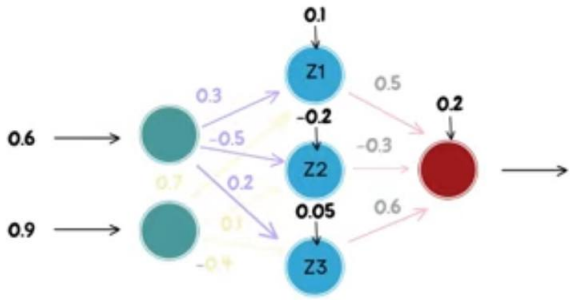

Figure 2.3: **Bias**

$$Z_1 = (0.6 * 0.3) + (0.9 * 0.7) + 0.1 = 0.91$$

$$Z_2 = (0.6 * (-0.5)) + (0.9 * 0.1) - 0.2 = 0.41$$

$$Z_3 = (0.6 * 0.2) + (0.9 * -0.4) + 0.05 = 0.4$$

Activation function

The neuron decides whether to transmit a signal or not, and it executes this decision using the activation function, which determines how the neuron responds to the outputs. There are several types. For example:

for example type Sigmoid, we take: Based on the values we obtained for the bias, we substitute them into the following formula.

$$H_1 = \frac{1}{1 + e^{-0.91}} = 0.713$$

$$H_2 = \frac{1}{1 + e^{0.41}} = 0.399$$

$$H_3 = \frac{1}{1 + e^{0.19}} = 0.453$$

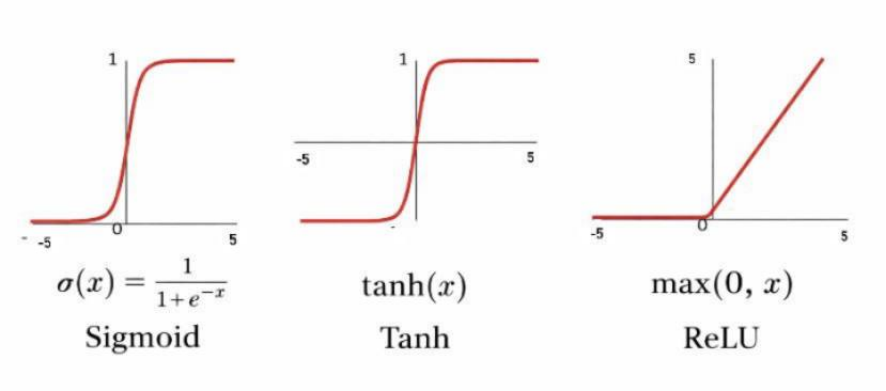

Figure 2.4: **Activation function**

Z= (0.713 * 0.5) + (0.399 * (- 0.3)) + (0.453 * 0.6) + 0.2 = 0.708

$$\sigma(Z) = \frac{1}{1 + e^{-Z}} = 0.670$$

Subsequently, a threshold is applied , which is usually between 0 and 1 often taken as 0.5. If the final result is greater than the threshold, the neuron is activated; otherwise, it remains inactive.

2.1.4 Loss function

Loss function is a method used to evaluate how well the network performs and whether its predicted outputs are accurate or not. Its main role is to compute the difference between the values predicted by the network and the target values we aim to reach. The objective is to minimize the error function; the smaller the error, the better the network performs, and vice versa.

Moreover, the choice of the appropriate type of loss function depends on the task and the network architecture. There are many types, among which the most commonly used ones include:

- ***Mean Squared Error(MSE):***

  The mean squared error is a function that measures the squared difference between the actual values and predicted outputs. It is given by the following relationship

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- ***Binary Cross-Entropy (BCE):***

  Binary Cross-Entropy measures the error between the predicted and actual values, but it is restricted to only two outcomes or two classes to choose between, such as yes or no .

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p(y_i)) + (1 - y_i)\log(1 - p(y_i))]$$

  There are other types of loss function, such as Huber loss,Categorical Cross-Entropy (CCE) and Mean Absolute Error (MAE) .

2.1.5 Backpropagation

Backpropagation is algorithm works by moderating the weights, which is done by reducing the difference between the value the network predicted and the actual values when calculating this error at the output. To estimate the impact of each weight on the occurrence of this error, we use the chain rule.

$$\frac{\partial \text{L}}{\partial w} = \frac{\partial \text{L}}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w}$$

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

Where η represents learning rate

Weights are adjusted in two main stages:

Stage	Description
1. Forward propagation	It begins with forward propagation, which involves propagating the input data from the input to the output end. Here, the error is calculated as the difference between the value estimated by the network and the actual value.
2. Backpropagation	When calculating the error, the network is reversed, a process known as backpropagation. At this stage, we calculate the gradients, and through these, adjusting weights.

Table 2.1: Forward and Backpropagation Stages

The figure below summarizes the backpropagation proces:

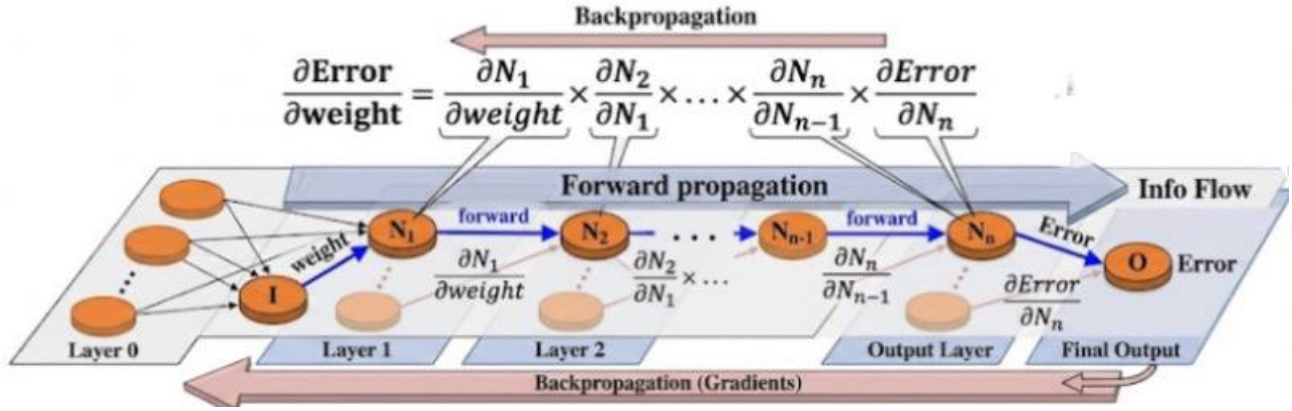

Figure 2.5: **The backward pass of a neural network**

2.1.6 Optimizer

The optimizer determines how neural networks learn; it finds parameters that minimize the loss function, thereby minimizing the loss function and increasing accuracy. It has

several types; we will mention the two most important types:

Algorithm 1: Method of Steepest Descent

Require: objective function $\mathcal{L}(\Theta)$
Require: Initialization: Θ_0

$k \leftarrow 0$ ▷ Initialize iteration step;
$\alpha_k > 0$ ▷ Compute step size via line search;
while Θ_k not converged **do**
  $k \leftarrow k + 1$;
  $g_k \leftarrow \nabla_\Theta \mathcal{L}(\Theta_{k-1})$ ▷ Compute gradient at iteration step k;
  $\Theta_k \leftarrow \Theta_{k-1} - \alpha_{k-1} g_k$ ▷ Update parameters;
  $\alpha_k \leftarrow \alpha_{k-1}$ ▷ Compute new step size;
return Θ_k ▷ Resulting parameters;

Algorithm 2: Adam

Require: stepsize α, decay rates $\beta_1, \beta_2 \in [0, 1)$, objective function $\mathcal{L}(\Theta)$
Require: Initialization: Θ_0

$k \leftarrow 0$ ▷ Initialize iteration step;
$m_0 \leftarrow 0$ ▷ Initialize 1st moment vector;
$v_0 \leftarrow 0$ ▷ Initialize 2nd moment vector;
while Θ_k not converged **do**
  $k \leftarrow k + 1$;
  $g_k \leftarrow \nabla_\Theta \mathcal{L}(\Theta_{k-1})$ ▷ Compute gradient at iteration step k;
  $m_k \leftarrow \beta_1 \cdot m_{k-1} + (1 - \beta_1) \cdot g_k$ ▷ Update biased 1st moment estimate;
  $v_k \leftarrow \beta_2 \cdot v_{k-1} + (1 - \beta_2) \cdot g_k^2$ ▷ Update biased 2nd raw moment estimate;
  $\hat{m}_k \leftarrow \frac{m_k}{1-\beta_1^k}$ ▷ Compute bias-corrected 1st moment estimate;
  $\hat{v}_k \leftarrow \frac{v_k}{1-\beta_2^k}$ ▷ Compute bias-corrected 2nd raw moment estimate;
  $\Theta_k \leftarrow \Theta_{k-1} - \alpha \cdot \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \epsilon}$ ▷ Update parameters;
return Θ_k ▷ Resulting parameters;

2.2 Physics-Informed Neural Networks

2.2.1 Introduction

After discussing the study of ANNs as an introduction to PINNs, it is worth noting that recent decades have witnessed remarkable advances in deep learning, which have reshaped the fields of computational physics and engineering. When solving partial differential equations using traditional methods,obstacles arise due to the complexity of mesh generation for irregular geometries, the curse of dimensionality, and high computational costs, leading to. This has paved the way for PINNs as a qualitative solution, where physical laws are incorporated during the learning process, enabling accurate and efficient solutions and supporting new opportunities for scientific and engineering applications.

2.2.2 Definition of Physics-Informed Neural Networks

Physics-Informed Neural Networks are a type of neural network that incorporate physical laws when solving partial differential equations during the learning phase, unlike traditional methods which rely entirely on data. PINNs are also based on improving the learning process by incorporating governing equations, which control the system's behavior. By integrating these physical laws into the training, the network's predictions become more consistent with the physical reality, leading to more accurate and reliable results.

2.2.3 Architecture of Physics-Informed Neural Networks

- **Inputs:** The independent variables, such as time (t) and space (x), are fed into the Neural Network (NN).
- **Neural Network (NN):** The Network is responsible for approximating the solution, denoted as $\hat{u}(x, t; \Theta)$, where Θ represents the set of trainable weights and

biases.

- **Automatic differentiation:** This mechanism is used to compute the exact partial derivatives of the network's output $\hat{u}$ with respect to its independent input variables (x and t). Unlike traditional numerical methods, it is based on the chain rule, which avoids discretization errors and allows PINNs to seamlessly incorporate the physical laws into the loss function, as illustrated in the architecture below:

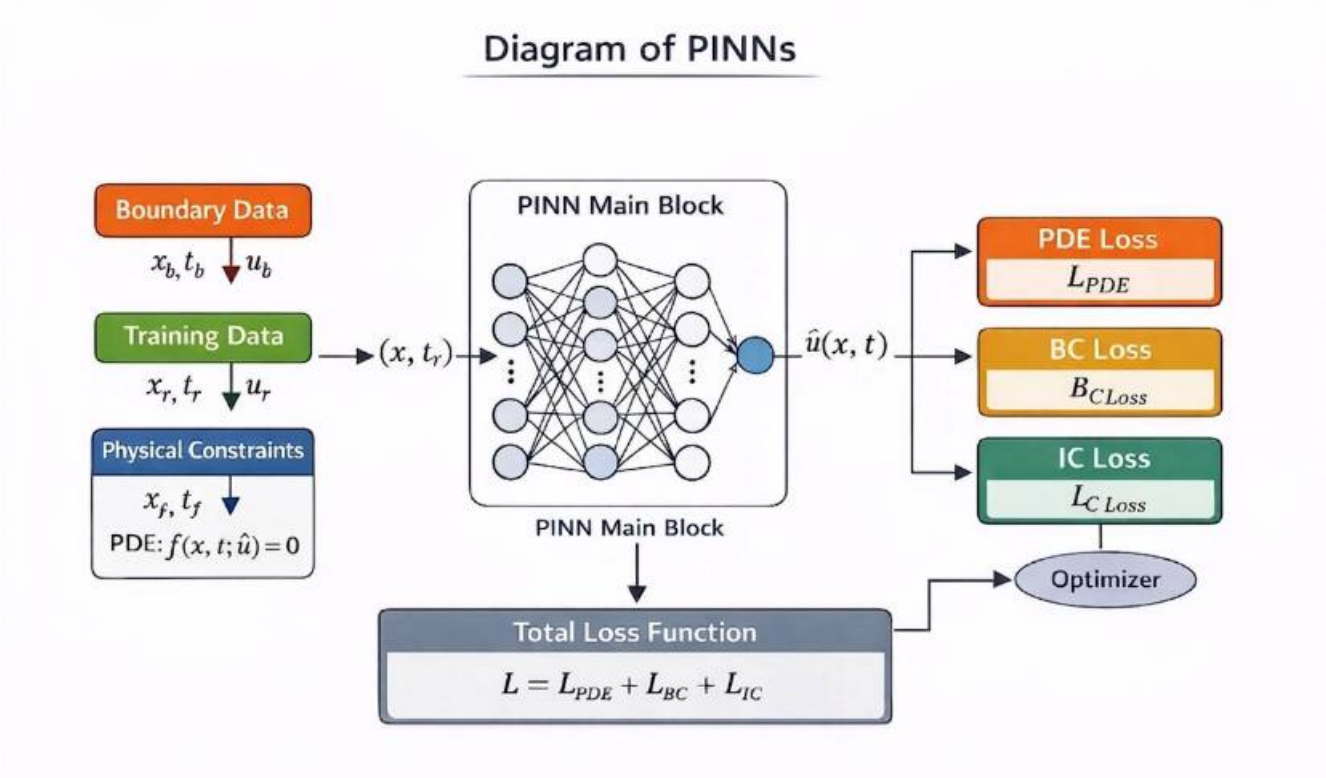

Figure 2.6: **PINNs Architecture**

2.2.4 Training

We previously discussed how ANNs are trained, and PINNs are trained in a similar way. Since PINNs are a special case of ANNs, their similarity lies in their reliance on the same structural framework, employing activation functions, backpropagation, and optimization algorithms. The difference lies in the fact that PINNs do not rely solely on data, unlike traditional ANNs, but rather on constraint-based learning, which is combined with differential equations to find a solution that satisfies those constraints. The training process in Physics-Informed Neural Networks (PINNs) begins with formulating the governing differential equation, then incorporating the boundary and initial conditions into the loss function. We select collocation points, which represent specific locations within the domain where the residual of the differential equation is evaluated, while boundary conditions are evaluated at the boundaries. We generate data at these collocation points to train the network, and we use these points to enforce the governing differential equation throughout the domain. After identifying these points, we calculate the loss function. The loss function contains two parts: one part pertains to data, as previously discussed in the neural network section, which works by obtaining values that approach or are close to the real values stored in the network. The other part concerns physics.

$$\mathcal{L} = \mathcal{L}_{\text{physics}} + \mathcal{L}_{\text{data}} \tag{1}$$

We incorporate these conditions Initial Conditions and Boundary Conditions and Residual PDE into the neural network as an additional loss.

Here, $\mathcal{L}_{\text{physics}}$ enforces the PDE constraints as follows:

$$\mathcal{L}_{\text{physics}} = w_f \mathcal{L}_f(\theta; T_f) + w_b \mathcal{L}_b(\theta; T_b) + w_i \mathcal{L}_i(\theta; T_i) \tag{2}$$

with:

$$\mathcal{L}_f(\theta; T_f) = \frac{1}{|T_f|} \sum_{x_f \in T_f} |f(x_f, t, \lambda)|^2 \tag{2.1}$$

$$\mathcal{L}_b(\theta; T_b) = \frac{1}{|T_b|} \sum_{x_b \in T_b} |\tilde{u}(x_b, t; \theta) - g(x_b, t)|^2 \tag{2.2}$$

$$\mathcal{L}_i(\theta; T_i) = \frac{1}{|T_i|} \sum_{x_i \in T_i} |\tilde{u}(x_i, 0; \theta) - h(x_i)|^2 \tag{2.3}$$

We define u as the unknown solution to the PDE problem within the domain Ω, where x represents the spatial coordinate and t denotes the time variable for time-dependent problems. The physical system is characterized by parameters λ, while the neural network is governed by the parameters θ. The optimization process minimizes the total loss, which consists of the residual loss $\mathcal{L}_f$, the boundary condition loss $\mathcal{L}_h$, and the initial condition loss $\mathcal{L}_i$.

2.2.5 Comparison between the finite element method and PINNs

Based on the numerical results [7] obtained from the simulation of the three equations, namely the Poisson equation in its different dimensions, the Allen–Cahn equation, and the Schrödinger equation, we analyzed the performance and computational cost, then proceeded to the evaluation stage in order to compare the Finite Element Method FEM and Physics-Informed Neural Networks PINNs.

Several criteria were considered in this evaluation, including the accuracy level, the required training time, and the speed of obtaining results after training the model.

From the obtained results, we identified the strengths and weaknesses of both methods. The following table summarizes the main differences that were recorded throughout the various stages of the study.

PINNs	FEM
It does not require large amounts of data.	It does require large amounts of data
It is mesh-free and does not rely on a computational mesh.	It is mesh-based and relies on a computational mesh.
It takes a long time to reach the optimal solution	It takes less time than PINNs to reach the optimal solution
It reduces the likelihood of producing non-physical or unrealistic results	More accurate than PINNs
It has strong generalization capability.	It faces problems with high dimensions.
Very limited in accuracy, once a certain level is reached it becomes difficult to improve.	Efficient for low dimensions (1D, 2D, 3D) but suffers from the curse of dimensionality.

Table 2.2: Comparison between PINNs and FEM

2.3 Universal approximation theorems

2.3.1 Introduction

Through the study conducted by G.Cybenko in 1989, which paved the way for understanding neural networks and specifically the theory of total approximation, it was established that we can approximate any continuous function using a simple neural network with only one hidden layer. The degree of complexity does not matter because the simple neural network is capable of approximating it, provided there is a suitable number of neurons, by imposing only simple conditions on a univariate function.

The general functions of a real variable of dimension n are represented by finite linear constructs of this form:

$$\sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j) \tag{1}$$

The main techniques we use are derived from standard functional analysis. The proof of the main theorem is as follows: We begin by noting that the finite sums of Equation (1) define a subspace in the space of all continuous functions on the unit hypercube of R. Using the Hahn-Banach and Ries theorems for representation

2.3.2 Universal approximation theorems

Definition 2.3.1 *we say that σ is sigmoidal if*

$$\sigma(t) \to \begin{cases} 1 & as\ t \to +\infty, \\ 0 & as\ t \to -\infty. \end{cases}$$

1.Cybenko's theorem: the case of continuous sigmoidal functions

Theorem 2.3.2 *[5] Let σ be any continuous discriminatory function. Then finite sums of the form*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j) \tag{2}$$

are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\epsilon > 0$, there is a sum, $G(x)$, of the above form, for which

$$|G(x) - f(x)| < \epsilon \quad for\ all \quad x \in I_n.$$

Proof. [5] Let $S \subset C(I_n)$ be the set of functions of the form $G(x)$ as in 2. Clearly S is a linear subspace of $C(I_n)$. We claim that the closure of S is all of $C(I_n)$. Assume that the closure of S is not all of $C(I_n)$. Then the closure of S, say R, is a closed proper subspace of $C(I_n)$. By the Hahn-Banach theorem, there is a bounded linear functional on $C(I_n)$, call it L, with the property that $L \neq 0$ but $L(R) = L(S) = 0$.

By the Riesz Representation Theorem, this bounded linear functional, L, is of the form

$$L(h) = \int_{I_n} h(x) d\mu(x) \tag{2.4}$$

for some $\mu \in M(I_n)$, for all $h \in C(I_n)$. In particular, since $\sigma(y^T x + \theta)$ is in R for all y and θ, we must have that

$$\int_{I_n} \sigma(y^T x + \theta) d\mu(x) = 0 \tag{2.5}$$

for all y and θ. However, we assumed that σ was discriminatory so that this condition implies that $\mu = 0$ contradicting our assumption. Hence, the subspace S must be dense in $C(I_n)$. ■

Lemma 2.3.3 *Any bounded, measurable sigmoidal function, σ, is discriminatory. In particular, any continuous sigmoidal function is discriminatory.*

Theorem 2.3.4 *Let σ be any continuous sigmoidal function. Then finite sums of the form*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma\left(y_j^T x + \theta_j\right) \tag{2.6}$$

are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\epsilon > 0$, there is a sum, $G(x)$, of the above form, for which

$$|G(x) - f(x)| < \epsilon \quad \textit{for all } x \in I_n. \tag{2.7}$$

Proof. Combine Theorem 1 and Lemma 1, noting that continuous sigmoidals satisfy the conditions of that lemma. ■

We now demonstrate the implications of these results in the context of decision regions. Let m denote Lebesgue measure in I_n. Let $P_1, P_2, \ldots, P_k$ be a partition of I_n into k disjoint, measurable subsets of I_n. Define the decision function, f, according to

$$f(x) = j \quad \text{if and only if } x \in P_j. \tag{2.8}$$

This function f can be viewed as a decision function for classification: if $f(x) = j$, then we know that $x \in P_j$ and we can classify x accordingly. The issue is whether such a decision function can be implemented by a network with a single internal layer.

We have the following fundamental result.

2.Hornik's theorem: the general case

After George Cybenko proved that single-hidden-layer neural networks are capable of approximating any continuous function with arbitrary accuracy by introducing continuous sigmoidal activation functions into this problem, this work represented the first foundation of what is known today as the Universal Approximation Theorem. Cybenko also mentioned that there were related studies on the same issue conducted by Kurt Hornik and his collaborators, stating: "A number of other current works are devoted to the same kinds of questions addressed in this paper."

Theorem 2.3.5 (Hornik, 1991) *[5] Let σ be any continuous bounded nonconstant function. Then $\Sigma_r(\sigma)$ is dense in $C(K)$ for every compact subset K of $\mathbb{R}^r$.*

In other words, for any $f \in C(K)$ and given any $\epsilon > 0$, there exists a function $G \in \Sigma_r(\sigma)$ such that:

$$\max_{x \in K} |G(x) - f(x)| < \epsilon \tag{2.9}$$

Where $G(x)$ is defined as the neural network output:

$$G(x) = \sum_{j=1}^{n} \beta_j \sigma \left(w_j^T x + b_j \right) \tag{2.10}$$

According to Hornik, the result is not restricted to sigmoidal functions only. In other words, the type of activation function is not important, provided that the function is continuous, bounded, and non-constant.

Chapter 3

Numerical results

In this chapter will be dedicated to a study covering several aspects of partial differential equations. We will highlight their classification and explain the Dirichlet and Neumann conditions.

3.1 Introduction to the Computational Tools

To illustrate what we discussed in the previous two chapters, we use computational tools to apply the PINNs method. These tools consist of:

3.1.1 Python programming language

Python is one of the most popular languages that is spreading and growing rapidly. Its use is not limited to developers only, but also includes data analysts, mathematicians, accountants, and network engineers. It is also used in several fields, most notably artificial intelligence, machine learning, and data analysis.

3.1.2 DeepXDE

DeepXDE is a software tool inside Python used to solve ordinary differential equations ODEs and partial differential equations PDEs using deep learning techniques. The main idea is that, instead of solving differential equations using traditional numerical methods, we rely on a neural network that learns the solution on its own to approximate the result.It is also easy to use, as any researcher or even a student can use it simply by defining the equation and setting the boundary and initial conditions. After that, the library provides the solution by training neural networks.It also offers efficient solutions for scientific and engineering problems with practical applications

3.2 Practical examples and numerical results

3.2.1 Application to second order ODE

We have the following ODE system:

$$\begin{cases} y''(t) - 3y'(t) + 2y(t) = 2t, \quad t \in [0,1] \\ \\ y(0) = 1, \quad y'(0) = 2 \end{cases}$$

with the exact solution:

$$y(t) = -2e^t + \frac{3}{2}e^{2t} + t + \frac{3}{2}$$

We define a fully connected feedforward neural network with 3 hidden layers of 20 neurons each, using the hyperbolic tangent activation function and Glorot normal weight initialization. We compile the model using the Adam optimizer with a learning rate of 0.001, we also compute the L^2 relative error as a metric during training.

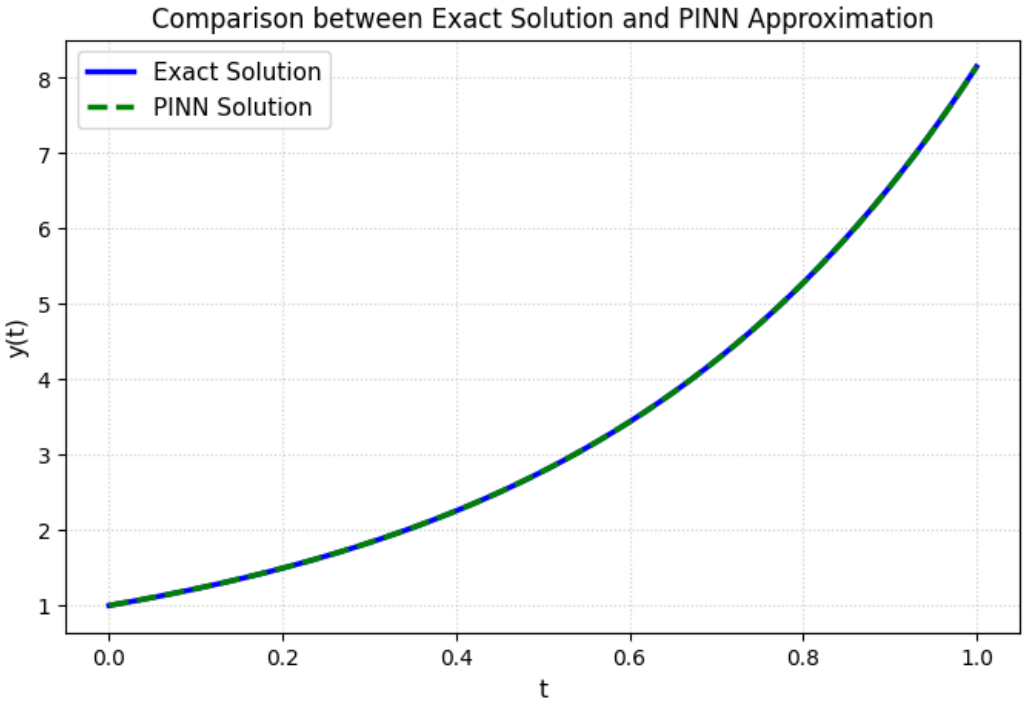

Figure 3.1: **The exact solution and approximation solutions (ODE problem).**

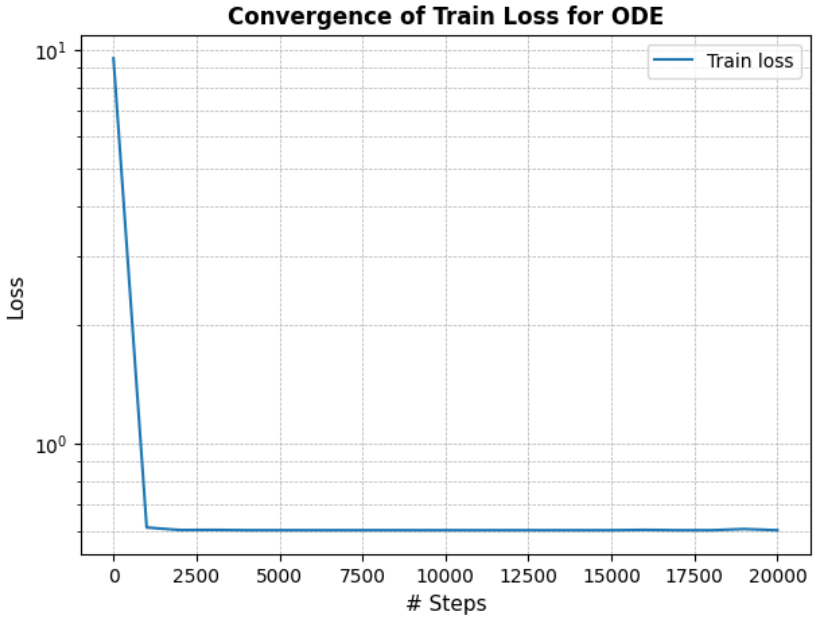

Figure 3.2: **The loss history for ODE .**

3.2.2 Application to the heat equation

Example 1: Consider the one-dimensional heat equation:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \alpha \dfrac{\partial^2 u}{\partial x^2} & x \in [0,1], \quad t \in [0,1] \\ u(0,t) = u(1,t) = 0 \\ u(x,0) = \sin(\pi x) + x \end{cases} \tag{3.1}$$

where $u(x,t)$ is the temperature at position x and time t, α is the thermal diffusivity, where $\alpha = 0.4$

The exact solution is: $e^{-\pi^2 \alpha t} \sin(\pi x) + x$

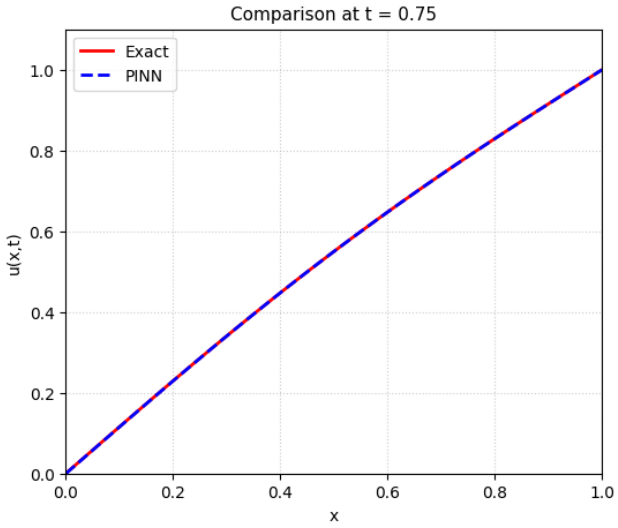

Figure 3.3: **The exact and approximation solutions (heat problem 1).**

The combination of the Adam optimizer and the L-BFGS optimizer improved the training process and reduced the loss function, enabling a good agreement at t = 0.75. These results confirm the effectiveness of Physics-Informed Neural Networks (PINNs) in solving differential equations with high accuracy and efficiency.

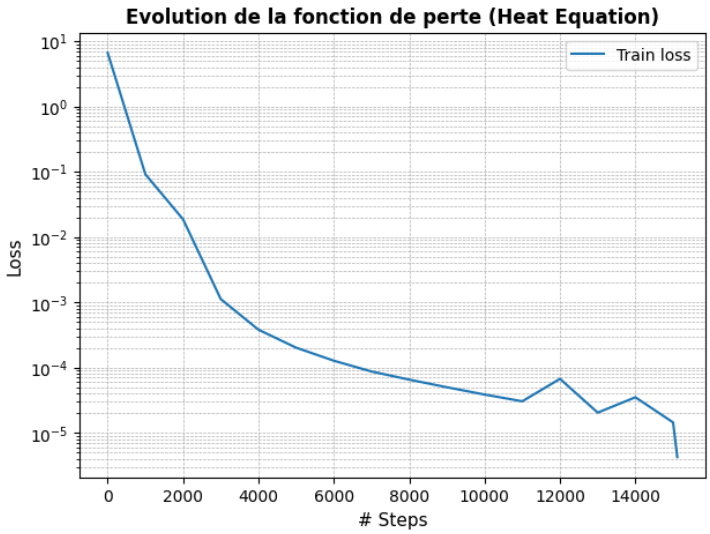

Figure 3.4: **The loss history for heat problem 1**

In the plot, if we divide it into three stages, we first observe that the Adam optimizer rapidly reduces the error from 10^{1} to 10^{-3}. Then comes the second stage, where the network becomes relatively stable and learns from the problem data, leading to a gradual decrease in the error. After that, the L-BFGS optimizer takes over, showing oscillations due to abrupt weight adjustments aimed at correcting the trajectory. This eventually drives the error down to its minimum value, below 10^{-5}, which enabled a strong agreement between the solutions.

Example 2: Consider the one-dimensional heat equation:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \alpha \dfrac{\partial^2 u}{\partial x^2} & x \in [0,1], \quad t \in [0,1] \\ u(0,t) = u(1,t) = 0 \\ u(x,0) = \sin(\pi x) \end{cases} \tag{3.2}$$

where $u(x,t)$ is the temperature at position x and time t, α is the thermal diffusivity, where $\alpha = 0.4$

The exact solution is: $e^{-\pi^2 \alpha t} \sin(\pi x)$

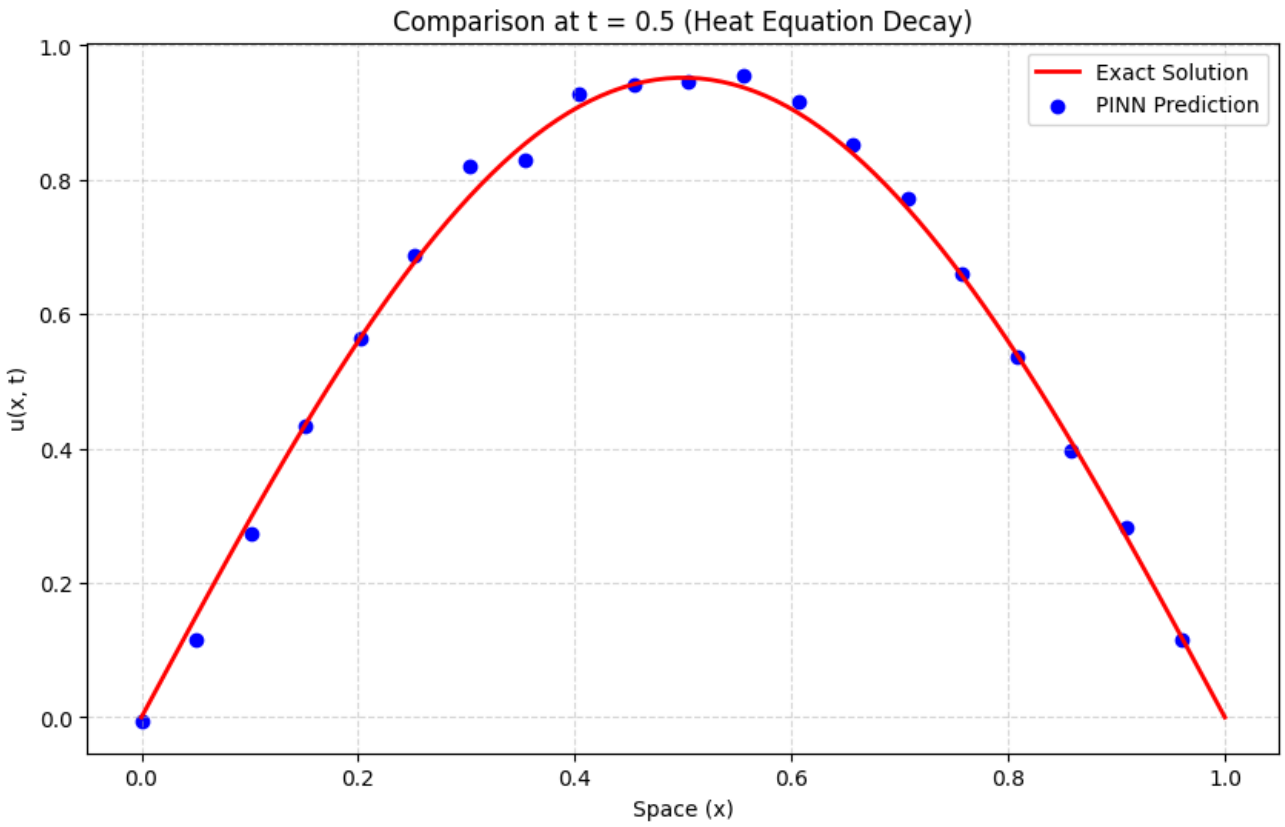

Figure 3.5: **The exact and approximation solutions (heat problem 2).**

3.2.3 Application to one-dimensional porous-elastic system

In this section, we apply the numerical model introduced earlier in Chapter 1, through which we established the well-posedness and exponential stability.

$$\begin{cases} \rho u_{tt} - \mu u_{xx} - b\phi_x = 0 & \text{in } (0,1)\times(0,\infty), \\ J\phi_{tt} - \delta\phi_{xx} + bu_x + \xi\phi + \tau\phi_t = 0 & \text{in } (0,1)\times(0,\infty), \end{cases}$$

In a numerical example Initial conditions:

$$u(x,0) = \sin(\pi x), u_t(x,0) = 0, \phi(x,0) = \sin(\pi x), \phi_t(x,0) = 0$$

Boundary conditions (Dirichlet-Dirichlet):

$$u(0,t) = u(1,t) = \phi(0,t) = \phi(1,t) = 0 \quad \forall t \in [0,10]$$

We choose the constants as follow:

$$\rho = \delta = J = \xi = \mu = 1$$

$$b = \tau = 0.5$$

Regarding the energy function $E(t)$ and its derivative $E'(t)$, one can refer to equations (1.9) and (1.10) in Chapter 1.

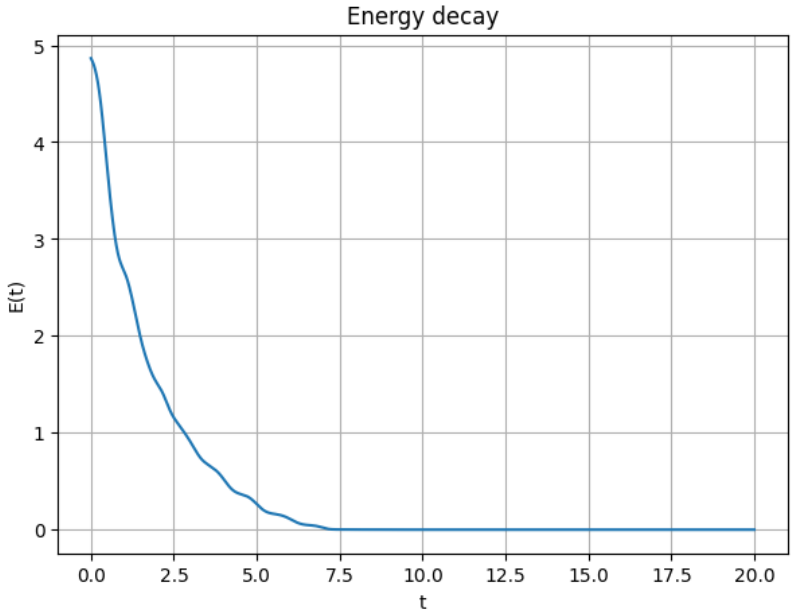

Figure 3.6: **The energy** $E(t)$**.**

At $x = 0.5$, both solutions u and ϕ exhibit a damped oscillatory behavior, where the amplitude of the oscillations decreases rapidly as time progresses. A complete rest state ($u = \phi = 0$) is achieved at $t \geq 10$. This behavior demonstrates the high efficiency of the damping coefficient ($\tau = 0.5$) in absorbing the internal dissipation.

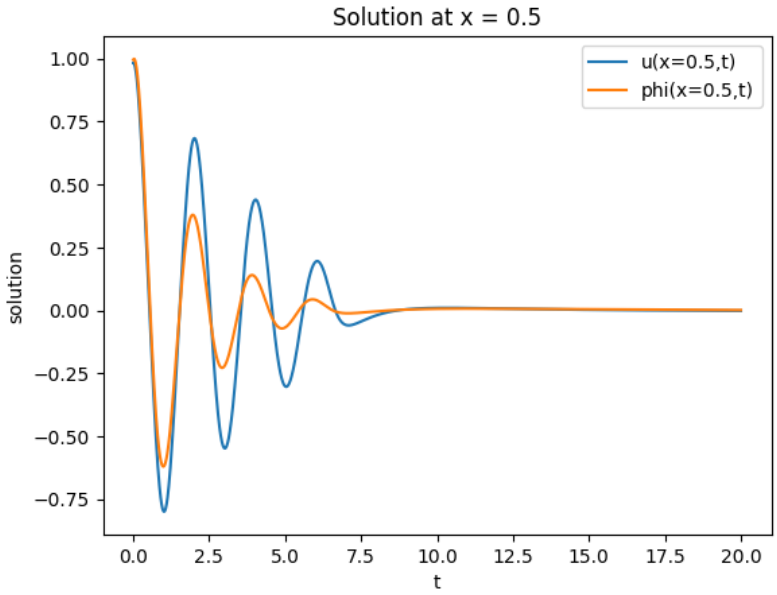

Figure 3.7: **The solutions u(x,t) and $\phi(x,t)$ at the spatial point x=0.5.**

Conclusion

Through the study we conducted, we were able to demonstrate the effectiveness of Physics-Informed Neural Networks PINNs in solving differential equations. The obtained results showed a very close agreement between the exact solution and the approximate solution. A numerical simulation of PINNs was implemented using the DeepXDE library by incorporating the physical constraints and governing laws during training, ensuring that the solutions are consistent with the underlying physics. This contributed to improving both the accuracy and the stability of the model.In addition, for the one-dimensional elasticity system, we obtained good results with good agreement and stability with the physical response of the studied system. These results remain open to further improvement. And this opens up broad and promising prospects for improving and effectively applying PINNs in this field.

Bibliography

[1] O. Aiadi, *Deep Learning Course Material*, Lecture notes for Master 2 in AI/Data Science, Faculty of New Information and Communication Technologies, Université Kasdi Merbah Ouargla, 2022/2023.

[2] T. A. Apalara, "Exponential decay in one-dimensional porous dissipation elasticity," *University of Hafr Al-Batin*, 2017.

[3] W. Arendt and M. Warma, "Dirichlet and Neumann boundary conditions: What is in between?," *Journal of Evolution Equations*, vol. 3, no. 1, pp. 119–135, 2003.

[4] Y. Bazilevs and T. J. R. Hughes, "Weak imposition of Dirichlet boundary conditions in fluid mechanics," *Computers & Fluids*, vol. 36, no. 1, pp. 12–26, 2007.

[5] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[6] L. C. Evans, *Partial Differential Equations*, Graduate Studies in Mathematics, Vol. 19, American Mathematical Society, Second Edition, 2010.

[7] T. G. Grossmann, U. J. Komorowska, J. Latz, and C.-B. Schönlieb,Can physics-informed neural networks beat the finite element method?, IMA Journal of Applied Mathematics, **89**(1), pp. 143–174, 2024.

[8] H. Lakhal, *Formulation variationnelle des problèmes elliptiques*, Cours d'Analyse Fonctionnelle 2, Département de Mathématiques, Université 20 août 1955 Skikda.

[9] K. Luo, J. Zhao, Y. Wang, J. Li, J. Wen, J. Liang, H. Soekmadji, and S. Liao, "Physics-informed neural networks for PDE problems: a comprehensive review," *Artificial Intelligence Review*, vol. 58, p. 323, 2025.

[10] S. Merabet, *Éléments Finis : Cours et Exercices*, Notes de cours, Université Kasdi Merbah – Ouargla, 2021.

[11] F. Moutaj, "Artificial Neural Networks," Lecture 3, Artificial Intelligence 2, Department of Robotics and Intelligent Systems, Manara University, 2023.

[12] I. Rezzoug, *Introduction aux Différences finies*, Lecture notes, Faculté des Sciences Exactes, Université Larbi Ben M'Hidi – Oum El Bouaghi, 2019.

[13] I. Rezzoug, *Méthode des Différences Finies*, Notes de cours, Département des Mathématiques et de l'Informatique, Université Larbi Ben M'Hidi – Oum El Bouaghi, 2019.

[14] B. Sharimbayev ,S. Kadyrov, & A. Kavokin, Development and optimization of physics-informed neural networks for solving partial differential equations. *Natural and Technical Sciences.*(2026).

[15] W. A. Strauss, *Partial Differential Equations: An Introduction*, Second Edition, John Wiley & Sons, 2007.

[16] T. J. Sullivan. A brief introduction to weak formulations of PDEs and the finite element method. Technical note, June 2020.

Abstract:

This work aims to study and approximate the solutions of partial differential equations PDEs using artificial neural networks ANNs, based on the universal approximation theory, which guarantees their ability to accurately approximate solutions. In this context, particular attention is given to Physics-Informed Neural Networks PINNs as a modern and efficient approach, and their application to the study and simulation of a one-dimensional porous elasticity system. The numerical simulations of these problems were carried out using the DeepXDE library, and the obtained results demonstrated the high accuracy and efficiency of this technique in solving PDEs, providing a precise and reliable representation of the studied Porous-Elastic System.

Keywords: Artificial Neural Networks, Physics-Informed Neural Networks, Partial Differential Equations, One-Dimensional Porous Elasticity System, DeepXDE.

Résumé:

Ce travail vise à étudier et à approcher les solutions des équations aux dérivées partielles EDP en utilisant les réseaux de neurones artificiels ANNs, en s'appuyant sur la théorie de l'approximation universelle qui garantit leur capacité à approximer les solutions avec précision. Dans ce contexte, une attention particulière est portée aux réseaux de neurones informés par la physique PINNs en tant qu'approche moderne et efficace, ainsi qu'à leur application dans l'étude et la simulation d'un système de poroélasticité unidimensionnel. Les simulations numériques de ces problèmes ont été réalisées à l'aide de la bibliothèque DeepXDE, et les résultats obtenus ont montré la grande précision et l'efficacité de cette technique dans la résolution des EDP, permettant ainsi de fournir une représentation précise et fiable du système poreux-élastique étudié.

Mots-clés: Réseaux de neurones artificiels, Réseaux de neurones informés par la physique, Équations aux dérivées partielles, Système de poroélasticité unidimensionnel,DeepXDE.

الملخص:

يهدف هذا العمل إلى دراسة وتقريب حلول المعادلات التفاضلية الجزئية PDEs بالاعتماد على الشبكات العصبية الاصطناعية ANN، مستندين إلى نظرية التقريب الشامل التي تضمن قدرتها على محاكاة الحلول بدقة. وفي هذا السياق، تم التركيز على تقنية الشبكات العصبية المستندة إلى الفيزياء PINNs كأداة حديثة وفعالة، وتطبيقها في دراسة ومحاكاة نظام المرونة المسامي أحادي الأبعاد. تمت المحاكاة العددية لهذه المسائل باستخدام مكتبة DeepXDE، حيث أظهرت النتائج دقة وفعالية عالية لهذه التقنية في معالجة الـ PDEs، مما مكّن من تقديم تمثيل دقيق وموثوق لنظام المرونة المسامي المدروس.

الكلمات المفتاحية :الشبكات العصبية الاصطناعية ،الشبكات المستندة إلى الفيزياء ، نظام المرونة المسامي أحادي الأبعاد،DeepXDE.