Universidad Nacional de Colombia
Sede Medellín

Facultad de Ciencias

Tesis de Maestría en Ciencias - Matemáticas

A mathematical framework of physics-informed neural networks for the solution of parabolic PDEs.

Un marco matemático de redes neuronales basadas en la física para la solución de ecuaciones diferenciales parciales (EDP) parabólicas.

Por: Josué David Acosta Castrillón

Directora: Dr. Manuela Bastidas Olivares
Codirector: Dr. Diego Alejandro Muñoz Durango

A la memoria de mi padre. Símbolo de fortaleza.
Julio Cesar Acosta Ruiz. 1963 - 2022.

Acknowledgements

I would like to express my sincere gratitude to my advisors, Dr. Manuela Bastidas Olivares and Diego Alejandro Muñoz Durango, whose insightful guidance and constant support have been instrumental in shaping this research. Their expertise and encouragement were vital throughout the development of this thesis.

I also wish to acknowledge the invaluable contributions of my colleagues and friends, whose stimulating discussions and constructive feedback have enriched this work. Special thanks are due to my family, whose love and belief in me have been a continual source of strength during both challenging and triumphant moments.

Finally, I appreciate the support provided by the Universidad Nacional de Colombia, which has afforded me the necessary resources and environment to pursue this study.

Abstract

Partial differential equations are some of the most useful mathematical tools to describe physical phenomena. Yet useful, many partial differential equations are difficult to solve in a classical way, even the solutions are difficult to find just by using analytical methods. For this reason many numerical methods are used in order to find numerical approximations. In the last years, the interest for developing numerical methods for solving PDEs using artificial neural networks have risen, following the accessibility to more user friendly frameworks for their development using home computers. Many methods have been documented that solve PDE in its strong and variational form for parabolic problems, however, there is not many progress in using neural networks for the solution of parabolic partial differential equations. Here we make use recent ideas for the development of variational physics informed neural networks for the solution of parabolic PDEs and expand on the mathematical background that supports their robustness. We obtained an neural network architecture that via time discretization, minimizes a loss function which achieves good approximations of the solution of time dependent problems in each time-step of the approximation maintaining coherence with the main idea behind classical VPINNs and the deep Fourier method. We show the effectiveness of our method in the solution of the freezing of coffee extracts in a industrial context and make use of measurements data for validation. We anticipate our work as a introductory material for any mathematician who wants to begin working on physics informed neural networks, by including an exhaustive and rigorous treatment on all the functional analysis topics necessary to lay the foundations in a mathematical way of the subject and also for any machine learning enthusiast who needs to make a bridge between the main ideas of data driven learning and the mathematical machinery behind residual minimization methods or partial differential equations in general.

Keywords: Partial differential equations, neural networks, functional analysis, residual minimization, physics informed neural networks, variational physics informed neural networks, numerical methods.

Resumen

Las ecuaciones diferenciales parciales son algunas de las herramientas matemáticas más útiles para describir fenómenos físicos. Sin embargo, aunque resultan muy útiles, muchas ecuaciones diferenciales parciales son difíciles de resolver de manera clásica, e incluso sus soluciones son complicadas de encontrar mediante métodos analíticos. Por esta razón, se emplean numerosos métodos numéricos con el fin de obtener aproximaciones numéricas. En los últimos años, ha crecido el interés por desarrollar métodos numéricos para la resolución de EDPs utilizando redes neuronales artificiales, impulsado por la accesibilidad a marcos de trabajo más amigables para su desarrollo en computadoras personales. Se han documentado muchos métodos que resuelven EDPs en su forma fuerte y variacional para problemas parabólicos; sin embargo, no se ha avanzado tanto en el uso de redes neuronales para la solución de ecuaciones diferenciales parciales parabólicas. En este trabajo hacemos uso de ideas recientes para el desarrollo de redes neuronales informadas por la física en su formulación variacional (VPINNs) para la solución de EDPs parabólicas, y profundizamos en los fundamentos matemáticos que respaldan su robustez. Obtuvimos una arquitectura de red neuronal que, mediante la discretización temporal, minimiza una función de pérdida que logra buenas aproximaciones de la solución de problemas dependientes del tiempo en cada paso temporal de la aproximación, manteniendo coherencia con la idea principal detrás de las VPINNs clásicas y del método de Fourier profundo (Deep Fourier Method). Mostramos la efectividad de nuestro método en la solución del proceso de congelación de extractos de café en un contexto industrial, utilizando datos experimentales para su validación. Anticipamos que nuestro trabajo sirva como material introductorio para cualquier matemático que desee comenzar a trabajar con redes neuronales informadas por la física, al incluir un tratamiento exhaustivo y riguroso de todos los temas de análisis funcional necesarios para sentar las bases matemáticas del tema, así como para cualquier entusiasta del aprendizaje automático que necesite tender un puente entre las ideas principales del aprendizaje basado en datos y la maquinaria matemática detrás de los métodos de minimización residual o de las ecuaciones diferenciales parciales en general.

Palabras clave: Ecuaciones diferenciales parciales parabólicas, redes neuronales, análisis funcional, minimización residual, redes neuronales informadas por la física, redes neuronales informadas por la física variacionales, métodos numéricos.

Contents

Chapter 1

Introduction

The study of partial differential equations (PDEs) has long been noutrished by rigorous frameworks based on functional analysis and variational formulations. Classical methods—such as the finite element method (FEM)—rely on Hilbert space theory and the construction of appropriate bases (see, e.g., [11, 19]) to guarantee well-posedness and derive error estimates. In parallel, recent developments in neural network techniques have opened new paths for approximating PDE solutions in high-dimensional and geometrically complex settings. However, purely data-driven approaches often lack the robustness provided by the underlying physical principles.

In this thesis, we develop a comprehensive mathematical framework that integrates variational formulations with neural network approximations. Central to this approach is the formulation of the governing PDEs in a weak form and the subsequent construction of robust variational physics-informed neural networks (RVPINNs). This method leverage the spectral decomposition of compact self-adjoint operators to construct an orthonormal Hilbert basis in Sobolev spaces (see Chapter 2). This basis is used both in classical FEM and to discretize the test space in the variational formulations, thereby enabling precise error estimation via the dual norm of the residual (cf. (3.34) and (6.13)).

Chapter 3 is devoted to the derivation of both the strong and weak formulations for model problems such as the Poisson and heat equations. In this context, we establish the continuity and coercivity properties of the associated bilinear forms and derive error bounds using the inf-sup stability condition. These results provide the theoretical foundation required to control the approximation error when replacing the infinite-dimensional test space with a finite-dimensional subspace.

In Chapter 4, we describe the architecture of physics-informed neural networks and formulate loss functions based on the weak residual. By incorporating automatic differentiation into the variational framework, we obtain a loss function that reflects the true error measured in the H^{-1}-norm. Extensions of this idea lead to the robust variational PINNs (RVPINNs) and the deep Fourier residual method (DFRM), where the use of an orthonormal basis (via Parseval's identity) allows for an efficient and reliable evaluation of the residual.

Finally, the developed methodology is applied to a industrial-scale problem, the freezing of coffee extracts. In this application, accurate prediction of spatiotemporal temperature profiles is critical for preserving product quality and ensuring process efficiency. By merging classical numerical analysis with contemporary deep learning techniques, this thesis offers a new, mathematically rigorous pathway for solving parabolic PDEs.

The following chapters detail the theoretical foundations, methodological innovations, and numerical experiments that together form the basis of the approach.

Chapter 2

Hilbert spaces

In this chapter, we develop a functional analysis framework that serves as the foundation for the analysis of variational formulations and numerical approximations of partial differential equations. We begin by introducing Banach and Hilbert spaces, two fundamental classes of complete normed and inner product spaces, respectively, with an emphasis on their structural properties, dual spaces, and the norms that govern their geometry. For a detailed exposition of these concepts, the reader is referred to standard texts such as [2] and [11].

A central aim of this chapter is the construction of a Hilbert basis for any separable Hilbert space via the spectral decomposition of a compact self-adjoint operator (see, e.g., [19]). This result, which follows from the application of the Riesz–Fréchet representation theorem (5) and classical spectral theory, guarantees that one may represent elements in a Hilbert space as a convergent series of orthogonal eigenfunctions. Such a basis is of intrinsic theoretical interest but also plays a crucial role in the development of numerical methods, such as finite element, included in Appendix B, and variational formulations, as well as in the formulation of physics-informed neural networks in later chapters. The foundational results presented here will be vital in deriving error estimates and ensuring the convergence of the methods that follow.

2.1 Banach and Hilbert spaces

We start by defining a norm in a vector space and discuss the properties of the norm that naturally lead to the notion of completeness. A vector space that is complete with respect to a given norm is termed a Banach space. In this context, classical examples—ranging from the Euclidean space $\mathbb{R}^n$ to spaces of continuous functions and sequence spaces such as ℓ^p—serve to illustrate the diversity and utility of Banach spaces.

We then proceed to introduce inner product spaces, where the inner product induces a norm and provides a rich geometric structure. A Hilbert space is defined as a complete inner product space, and key properties such as the parallelogram law, the Cauchy–Schwarz inequality, and the existence of orthogonal projections are established. These properties are

central to the subsequent construction of an orthonormal (Hilbert) basis via the spectral decomposition of compact self-adjoint operators, a result that is fundamental to the overall approach.

Definition 1. Let V be a real vector space. A **norm**, denoted by $\|\cdot\|$, is a function $\|\cdot\| : V \to \mathbb{R}_+$, where $\mathbb{R}_+ = \{x \in \mathbb{R} \mid x \geq 0\}$, with the following properties:

i) $\|u + v\|_V \leq \|u\|_V + \|v\|_V$, (Triangle Inequality)

ii) $\|c \cdot u\| = |c|\|u\|_V$, (Scaling property)

iii) $\|u\|_V \geq 0$, with equality if and only if $u = 0$.

Example 1: The Euclidean space $\mathbb{R}^n$
A typical example of a normed vector space is $\mathbb{R}^n$, with the *Euclidean norm* defined as:

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

for $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$.

Example 2: The space of continuous functions $C([a, b])$
Let $C([a, b])$ be the space of continuous functions on $[a, b]$. The norm is typically given by the *supremum norm*:

$$\|f\|_\infty = \max_{x \in [a,b]} |f(x)|.$$

Definition 2. A **linear functional** on a real vector space V is a function $l : V \to \mathbb{R}$ that satisfies the following properties:

i) $l(u + v) = l(u) + l(v)$ (Additivity),

ii) $l(\alpha\, v) = \alpha\, l(v)$ (Homogeneity),

for all $u, v \in V$ and every scalar $\alpha \in \mathbb{R}$.

Example: The integral operator
For the space of continuous functions $C([a, b])$, a typical linear functional is:

$$l(f) = \int_a^b f(x)\, dx.$$

This satisfies both linearity properties.

Definition 3. A linear functional $l : V \to \mathbb{R}$ is **continuous** or **bounded** if there exists a constant C such that:

$$|l(v)| \leq C\, \|v\|_V \quad \forall v \in V.$$

Definition 4. A sequence $\{u_i\}_{i=1}^{\infty}$ in a metric space V converges to a limit $u \in V$ if it satisfies:

$$\lim_{k\to\infty} \|u_k - u\|_V = 0.$$

Such a sequence is called **convergent**.

Definition 5. A sequence $\{u_i\}_{i=1}^{\infty} \subset V$ is called a **Cauchy sequence** if for every $\varepsilon > 0$, there exists $N \in \mathbb{N}$ such that:

$$\|u_m - u_n\|_V < \varepsilon \quad \text{for all } m, n > N.$$

Definition 6. A metric space V is called **complete** if every Cauchy sequence converges to a limit in V.

Example: Completeness of ℓ^2 space
The space of square-summable sequences ℓ^2 is an example of a complete space. A sequence $\mathbf{x} = (x_i)_{i=1}^{\infty}$ belongs to ℓ^2 if

$$\sum_{i=1}^{\infty} |x_i|^2 < \infty.$$

Every Cauchy sequence in ℓ^2 converges within ℓ^2, making it a complete space.

Definition 7. A **Banach space** is a vector space E equipped with a norm $\|\cdot\|_E$ that is **complete**. We denote this space as $(E, \|\cdot\|_E)$.

Definition 8. The **dual space** of E, denoted by E^*, is the space of all *continuous linear functionals* on E. The norm on E^* is defined as:

$$\|l\|_{E^*} = \sup_{u\in E\setminus\{0\}} \frac{|l(u)|}{\|u\|_E} = \sup_{\|u\|_E \le 1} |l(u)|.$$

So a linear functional is a function $l(\cdot) : E \to \mathbb{R}$ that is both linear and continuous. The dual space of a Banach space, with the norm defined previously, turns out to be a Banach space itself [11], so it is interesting to inquire about the dual space of the dual space of a Banach space.
Consider such a space, $(E^*)^* := \{\omega : E^* \to \mathbb{R} \mid \omega \text{ is continuous and linear}\}$. At first glance, it is not easy to make sense of this space, but note that given $u \in E$, there exists a "natural" or "canonical" way to find an element in $(E^*)^*$, which is intuitively an 'operator that operates on operators', and indeed, this is simply evaluating at u. That is:

$$\begin{aligned} \Psi : & E \longrightarrow (E^*)^* \\ & u \longmapsto \phi_u : \quad E^* \longrightarrow \mathbb{R} \\ & \qquad\qquad\quad \gamma \longmapsto \gamma(u) \end{aligned} \tag{2.1}$$

This mapping turns out to be injective since if $\Psi(u) = \Psi(v)$, it suffices to evaluate the induced mappings at the identity to determine that $u = v$.

Definition 9. Let E be a Banach space, and let $J : E \to E^{**}$ be the canonical injective mapping from E to E^{**}. The space E is said to be **reflexive** if this mapping is surjective, i.e., $J(E) = E^{**}$. When E is reflexive, E^{**} is usually identified with E.

Since L^p spaces with $1 < p < \infty$ are also **uniformly convex**, they satisfy the Milman–Pettis theorem (1), further confirming their reflexivity. Uniform convexity plays a crucial role in ensuring that every bounded sequence in L^p has a weakly convergent subsequence, which is another important property of reflexive spaces.

Definition 10. A Banach space E is said to be **uniformly convex** if

$$\begin{aligned} &\forall \varepsilon > 0,\ \exists \delta > 0, \quad \text{such that,} \\ &x, y \in E,\ \|x\| \leq 1,\ \|y\| \leq 1 \quad \text{and} \quad \|x - y\| > \varepsilon \to \left\| \frac{x+y}{2} \right\| < 1 - \delta. \end{aligned} \tag{2.2}$$

Uniform convexity is a geometric property of the unit ball in the space. If we draw a line of length $\varepsilon > 0$ in the ball, then the midpoint of this line must also lie within a ball of radius $1 - \delta$ for some $\delta > 0$. In particular, this means that the unit sphere must be *round* and cannot contain line segments.

It is important to note that the property of uniform convexity depends entirely on the norm. The same vector space equipped with different norms may or may not be uniformly convex. For example, $E = \mathbb{R}^2$ with the norms $\|x\|_2 = (|x_1|^2 + |x_2|^2)^{\frac{1}{2}}$ and $\|x\|_1 = |x_1| + |x_2|$. The first pair is uniformly convex while the second one is not.

Theorem 1 (Milman-Pettis / *Brezis* [11] Theorem 3.31)**.** Every uniformly convex Banach space is reflexive.

As mentioned, uniform convexity is a *geometric* property of the norm. If one changes the norm in the space to an equivalent one, the space may no longer be uniformly convex. On the other hand, reflexivity is a *topological* property; a reflexive space remains reflexive for any equivalent norm in the space. Therefore, Theorem 1, is quite interesting, as it shows that a geometric property implies a topological one.
Uniform convexity is commonly used to prove reflexivity, but it is not always the best approach. There are some **rare** spaces that are reflexive but do not admit a uniformly convex norm.

Definition 11. A **bilinear form** on E is a function $a(\cdot, \cdot) : E \times E \to \mathbb{R}$ such that

i) $a(u + v, w) = a(u, w) + a(v, w)$,

ii) $a(u, v + w) = a(u, v) + a(u, w)$,

iii) $a(\lambda u, v) = \lambda a(u, v)$,

iv) $a(u, \lambda v) = \lambda a(u, v)$,

for all $u, v, w \in E$ and $\lambda \in \mathbb{R}$.

Definition 12. Let V be a vector space. An **inner product** (u, v) is a bilinear form on $E \times E$ with values in $\mathbb{R}$ that satisfies:

i) $(u, v) = \overline{(v, u)}$, $\forall\ u, v \in E$, (symmetry)

ii) $(u, u) \geq 0$, $\forall u \in E$ and $(u, u) = 0$ if and only if $u = 0$. (positive definiteness)

The inner product on E defines a norm, given by

$$\|u\|_E = (u, u)^{1/2}.$$

A space equipped with an inner product is called a **pre-Hilbert** In the particular case where the space is finite-dimensional and defined over the real numbers, it is called a **Euclidean space.**

Definition 13. The **Cauchy-Schwarz Inequality** states that:

$$(u, v) \leq \|u\|_E \|v\|_E,$$

for all $u, v \in V$.

Definition 14. A **Hilbert space** is a vector space H equipped with an inner product, such that H is **complete** with respect to the induced norm $\|\cdot\|_H$.

It should be noted that this implies that every Hilbert space is by definition a Banach space. However, a Banach space may be complete with respect to some norm that is not induced by an inner product, hence not being a Hilbert space.

Proposition 2. Every Hilbert space H is uniformly convex, hence *reflexive*.

Proof. Let $\varepsilon > 0$, and let $u, v \in H$ such that $\|u\|_H \leq 1$, $\|v\|_H \leq 1$, and $\|u - v\|_H > \varepsilon$. By the parallelogram law, we have:

$$\left\| \frac{u+v}{2} \right\|_H^2 \leq \varepsilon - \frac{\varepsilon^2}{4}, \tag{2.3}$$

which implies

$$\left\| \frac{u+v}{2} \right\|_H < 1 - \delta, \tag{2.4}$$

where $\delta = 1 - \left(1 - \frac{\varepsilon^2}{4}\right)^{1/2} > 0$. □

Theorem 3 (Projection onto a Closed Convex Set)**.** Let $K \subset H$ be a closed convex set of a Hilbert space H. Then for every $f \in H$, there exists a unique element $u \in K$ such that

$$\|f - u\|_H = \min_{v \in K} \|f - u\|_H = \operatorname{dist}(f, K). \tag{2.5}$$

Additionally, u is characterized by the property

$$u \in K \ \text{ and } \ (f - u, v - u) \leq 0 \qquad \forall v \in K. \tag{2.6}$$

This element u is called the **projection** of f onto K, denoted by

$$u = P_K f.$$

For the proof of this theorem we refer to *Brezis* [11], Theorem 5.2.

Proposition 4. Let $K \subset H$ be a non-empty closed convex set. Then P_K does not increase distance:

$$\|P_K f_1 - P_K f_2\|_H \leq \|f_1 - f_2\|_H \qquad \forall f_1, f_2 \in H.$$

Again, for a complete proof of this theorem, refer to *Brezis*[11], Proposition 5.3.

Remark 1. If we require $K \subset E$ to be *uniformly convex* in a Banach space E, then for every $f \in E$, there exists a unique element u that "achieves" the distance from f to K, i.e.:

$$\|f - u\|_H = \min_{v \in K} \|f - v\|_H = \operatorname{dist}(f, K).$$

Corollary 4.1. Suppose $M \subset H$ is a closed linear subspace. Let $f \in H$. Then $u = P_M f$ is characterized by

$$u \in M \quad \text{and} \quad (f - u, v) = 0 \qquad \forall v \in M.$$

Additionally, P_M is a linear operator, called the **orthogonal projection**.

The proof can be found in *Brezis* [11] Corollary 5.4.

It is easy to find continuous linear functionals in a Hilbert space. For any $f \in H$, the assignment $u \mapsto (f, u)$ is a continuous linear functional on H. What is interesting is that *all* linear functionals on H are obtained this way.

Theorem 5 (Riesz-Fréchet Representation Theorem)**.** Given any $\varphi \in H^*$, there exists a unique $f \in H$ such that

$$\varphi(u) = (f, u) \quad \forall u \in H.$$

Additionally,

$$\|f\|_H = \|\varphi\|_{H^*}.$$

Proof. Define $M = \varphi^{-1}(\{0\})$, then since φ is continuous, M is a closed subspace of H. We can assume, moreover, that $M \neq H$ because otherwise $\varphi = 0$ and then $f = 0$. The claim is that there exists some element $g \in H$ such that

$$\|g\|_H = 1 \text{ and } (g, v) = 0 \ \forall v \in M \text{ and thus } g \notin M.$$

For this, let $g_0 \in H$ such that $g_0 \notin M$. Let $g_1 = P_M g_0$.
Then

$$g = \frac{g_0 - g_1}{\|g_0 - g_1\|_H},$$

satisfies the required properties.
For any $u \in H$, define

$$v = u - \lambda g \quad \text{with} \quad \lambda = \frac{\varphi(u)}{\varphi(g)}.$$

Note that v is well-defined, since $\varphi(g) \neq 0$, and moreover, $v \in M$ since $\varphi(v) = 0$.
So $(g, v) = 0$ hence

$$\varphi(u) = \varphi(g)(g, u) \quad \forall u \in H,$$

thus, we can define f as

$$f = \varphi(g)g.$$

□

Remark 2. Last theorem asserts that there is a canonical isometry from H onto H^*. It is therefore adequate to identify H and H^*. It is usual to do so but not *always*. Here's a typical situation, which later will come handy, where one should be cautious with identification. Assume that H is a Hilbert space with scalar product $(\cdot, \cdot)_H$ and a corresponding norm $\|\cdot\|_H$. Assume that $V \subset H$ is a linear subspace that is dense in H. Assume that V has its own norm $\|\cdot\|_V$ and that V is a Banach space with $\|\cdot\|_V$. Assume that the injection $V \subset H$ is continuous, i.e.

$$\|v\|_H \leq C\|v\|_V \quad \forall v \in V.$$

There is a canonical map $T : H^* \to V^*$ that is simply the restriction to V of the continuous linear functionals φ on H, i.e.,

$$(T\varphi, v)_{V^* \times V} = (\varphi, v)_{H^* \times H}.$$

Where T holds the following properties:

i) $\|T\varphi\|_{V^*} \leq C\|\varphi\|_{H^*}, \qquad \forall \varphi \in H^*,$

ii) T is injective,

iii) $R(T)$ is dense in V^* if V is reflexive.

Identifying H^* with H and using T as a canonical embedding from H^* into V^*, is usually written as

$$V \subset H \cong H^* \subset V^*, \tag{2.7}$$

where all the injections are continuous and dense. H is called the **pivot space**. Notice that the scalar products $(\cdot, \cdot)_{V^* \times V}$ and $(\cdot, \cdot)_H$ coincide whenever both make sense, i.e.,

$$(f, v)_{V^* \times V} = (f, v)_V \quad \forall f \in H, \quad v \in V.$$

The situation becomes more delicate if V turns out to be a Hilbert space with its own scalar product, associated to the norm as Banach space. We could, of course, identify V and V^* with the help of its inner product. However, equation (2.7) becomes absurd since it implies that V is a proper subset of itself. This shows that one cannot identify simultaneously V and H with their dual spaces: one has to make a choice.

2.1.1 Lax–Milgram theorem

We state the Lax–Milgram theorem, a result in the theory of variational formulations of boundary value problems. The theorem provides sufficient conditions, namely, the continuity and coercivity of the bilinear form, for the existence and uniqueness of weak solutions in a variational problems. In order to set the stage, we begin by recalling the definitions of a continuous and coercive bilinear form. ([11], [20])

Definition 15. A bilinear form $a : H \times H \to R$ is said to be

1. *continuous* if there exists a constant C_a such that
$$a(u, v) \leq C_a \|u\|_H \|v\|_H.$$
2. *coercive* if there exists a constant $\alpha > 0$ such that
$$a(v, v) \geq \alpha \|v\|_H^2 \qquad \forall v \in H.$$

Theorem 6 (Banach Fixed-Point Theorem)**.** Let X be a non-empty complete metric space, and let $S : X \to X$ be a **contraction**, i.e.
$$d(Sv_1, Sv_2) \leq k d(v_1, v_2), \qquad \forall v_1, v_2 \in X \ \ , k < 1.$$
Then S has a unique fixed point $u = Su$

A proof can be obtained from *Apostol* [2], Theorem 4.48.

Theorem 7 (Stampacchia)**.** Assume that $a(u, v)$ is a continuous and coercive bilinear form on H. Let $K \subset H$ be a closed convex subset. Then, given any $\varphi \in H^*$, there exists a unique element $u \in K$ such that
$$a(u, v - u) \geq \varphi(v - u) \ \ \forall v \in K. \tag{2.8}$$

Proof. From the Riesz-Fréchet representation theorem (5), we have that there exists a unique $f \in H$ such that
$$\varphi(v) = (f, v) \ \ \forall v \in H.$$
On the other hand, fixing any $u \in H$, the mapping $v \mapsto a(a, v)$ is a bounded linear functional on H. Again using Riesz-Fréchet, we can find a unique element in H, let us call it Au, such that $a(u, v) = (Au, v)\ \forall v \in H$. Here we must stretch our imagination and try to understand A as a *linear operator* from H to H which maps $u \mapsto Au$ for every $u \in H$ via Riesz-Fréchet. As Au arises from $a(\cdot, \cdot)$,
$$\begin{aligned} \|Au\|_H \leq & C_a \|u\|_H, \\ a(u, u) = (Au, u) \geq & \alpha \|u\|_H^2, \end{aligned} \tag{2.9}$$
so Au is continuous and coercive. We want to find $u \in K$ such that
$$a(u, v - u) = (Au, v - u) \geq (f, v - u) = \varphi(v - u) \ \ \forall v \in K,$$

and then we would have the first part of the theorem.
let us take $\rho > 0$ as a constant ($\in \mathbb{R}$) that we won't determine yet. By linearity

$$\begin{aligned} (Au, v-u) &\geq (f, v-u) \\ (f, v-u) - (Au, v-u) &\leq 0 \\ \rho((f, v-u) - (Au, v-u) &\leq 0 \\ (\rho f, v-u) - (\rho Au, v-u) &\leq 0 \\ (\rho f - \rho Au, v-u) &\leq 0 \\ (\rho f - \rho Au + u - u, v-u) &\leq 0. \end{aligned} \tag{2.10}$$

This gives us a clue, since we have by the theorem of projection onto a closed convex set (3), that if we show that there exists a u such that

$$u = P_K(\rho f - \rho Au + u),$$

then we would validate the last line of the previous expression and find the fugitive.
For this purpose let us define for every $v \in K$, $Sv = P_K(\rho f - \rho Av + v)$. We are going to try to adjust ρ so that S turns out to be a contraction. Which is not surprising since, as we stated previously, projections do not increase distances. So

$$\begin{aligned} \|Sv_1 - Sv_2\|_H &= \|P_K(\rho f - \rho Av_1 + v_1) - P_K(\rho f - \rho Av_2 + v_2)\|_H \\ &\leq \|\rho f - \rho Av_1 + v_1 - \rho f + \rho Av_2 - v_2\|_H = \|(v_1 - v_2) - \rho(Av_1 - Av_2)\|_H. \end{aligned} \tag{2.11}$$

Then, squaring both sides:

$$\begin{aligned} \|Sv_1 - Sv_2\|_H^2 &\leq \|(v_1 - v_2) - \rho(Av_1 - Av_2)\|_H^2 \\ &= \|v_1 - v_2\|_H^2 - 2\rho(Av_1 - Av_2, v_1 - v_2) + \rho^2\|Av_1 - Av_2\|_H^2. \end{aligned} \tag{2.12}$$

Finally, taking advantage of the coercivity and continuity, in the second and third terms respectively, we have that

$$\begin{aligned} \|Sv_1 - Sv_2\|_H^2 &\leq \|(v_1 - v_2) - \rho(Av_1 - Av_2)\|_H^2 \\ &= \|v_1 - v_2\|_H^2 - 2\rho(Av_1 - Av_2, v_1 - v_2) + \rho^2\|Av_1 - Av_2\|_H^2 \\ &\leq \|v_1 - v_2\|_H^2(1 - 2\rho\alpha + \rho^2 C^2) \end{aligned} \tag{2.13}$$

Then choosing $\rho > 0$ such that $1 - 2\rho\alpha + \rho^2 C^2 = k^2 < 1$, i.e. $0 < \rho < 2\alpha/C^2$ we can conclude that there exists $u \in K$ such that S has a fixed point, by the Banach fixed point theorem (6); that is $u = P_K(\rho f - \rho Au + u)$ thus proving the first part of the theorem. □

Corollary 7.1 (Lax-Milgram)**.** Suppose $a(u, v)$ is a continuous and coercive bilinear form on H. Then, for any $\varphi \in H^*$, there exists a unique element $u \in H$ such that

$$a(u, v) = \varphi(v) \qquad \forall v \in H. \tag{2.14}$$

Moreover, if a is symmetric, then u is characterized by the property

$$u \in H, \quad \frac{1}{2}a(u, u) - \varphi(u) = \min_{v \in H}\left\{\frac{1}{2}a(v, v) - \varphi(v)\right\}.$$

Proof. Let $w \in H$ be arbitrary. Since H is closed and convex in H, then by Stampacchia's result, for any $\varphi \in H^*$ there exists a unique $u \in H$ such that

$$\begin{aligned} a(u,(w+u)) &\geq \phi((w+u)-u) \\ a(u,w) &\geq \phi(w) \end{aligned} \tag{2.15}$$

Similarly,

$$\begin{aligned} a(u,(-w+u)) &\geq \phi((-w+u)-u) \\ a(u,-w) &\geq \phi(-w) \\ -a(u,w) &\geq -\phi(w) \\ a(u,w) &\leq \phi(w) \end{aligned} \tag{2.16}$$

therefore we have the result. □

2.2 Sobolev spaces

We introduce the concept of Sobolev spaces. These, extend the notion of differentiability by incorporating weak (distributional) derivatives. This approach is essential for defining solutions to partial differential equations in situations where classical derivatives may not exist. We begin with the definition of the Lebesgue space $L^p(\Omega)$ and from there we define the Sobolev spaces $W^{p,k}(\Omega)$ as the set of functions whose weak derivative up to order k belong to $L^p(\Omega)$. In the particular case $p = 2$, the space $W^{k,2}(\Omega)$ becomes a Hilbert space and inherits an inner product that accounts for both the functions and its derivatives.

Definition 16. Let $p \in \mathbb{R}$ such that $1 < p < \infty$; we define

$$L^p(\Omega) = \left\{ f : \Omega \to \mathbb{R};\ f \text{ is integrable and } |f|^p \in L^1(\Omega) \right\}, \tag{2.17}$$

where $L^1(\Omega)$ is the space of integrable functions on Ω, a bounded domain with Lipchitz boundary $\partial\Omega$, to the real numbers.

In L^p we have the norm given by

$$\|f\|_{L^p(\Omega)} = \|f\|_p = \left[\int_\Omega |f|^p dx \right]^{1/p}, \tag{2.18}$$

Theorem 8 (Fischer-Riesz)**.** L^p is a Banach space for all $1 \leq p \leq \infty$

For a proof, refer to *Brezis*[11], Theorem 4.8. Basic examples of a Hilbert space include the space $L^2(\Omega)$ equipped with the inner product

$$(u,v)_{L^2} = \int_\Omega uv \ dx.$$

Note that the case $p = 2$ is special because here $\|u\|_{L^2} = (u,u)_{L^2}$, making it a Hilbert space.

For $1 < p < \infty$, the dual space of $L^p(\Omega)$ is $L^q(\Omega)$, where $\frac{1}{p} + \frac{1}{q} = 1$. A linear functional l on L^p can be represented as:

$$l(f) = \int_\Omega f(x)g(x)\,dx \quad \text{with } g \in L^q(\Omega).$$

Definition 17. Functions in a Hilbert space, generally, are not regular, meaning that some of their derivatives may not be finite at a point in the domain of the function, and the standard definition of derivative cannot be made as naturally. For this, we need to introduce a concept called the **weak derivative**, which defines the derivative as a kind of "average".

Definition 18. Let $C^k(\Omega)$ be the space of all functions that have continuous derivatives up to order k on Ω. Also, let $\mathcal{D}$ be the space of all functions with compact support in Ω that are infinitely differentiable. This is

$$\mathcal{D}(\Omega) = \{\varphi \in C^\infty(\Omega) \mid \operatorname{supp}(\varphi) \subset\subset \Omega\}. \tag{2.19}$$

Let us break down this definition. First, the support of φ is the closure of the open set $\{x \in \Omega \mid \varphi(x) \neq 0\}$. Moreover, a function φ is said to have compact support with respect to Ω if $\operatorname{supp}(\varphi)$ is a compact set, say ω, which is a subset of the interior of Ω. When we write $\operatorname{supp}(\varphi) \subset\subset \Omega$ we mean that $\operatorname{supp}(\varphi) \subseteq \omega \subseteq \Omega$ where ω, as mentioned before, is compact and the support of φ is compactly contained in Ω. Therefore, functions in $\mathcal{D}(\Omega)$ can be differentiated infinitely and are strictly nonzero in Ω. In particular, they are zero 'close enough' to $\partial\Omega$

Let $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ be a point in the domain $\Omega \subset \mathbb{R}^n$, $\varphi \in \mathcal{D}(\Omega)$ be a smooth function with compact support), $u : \Omega \to \mathbb{R}$ a function with regularity $C^k(\Omega)$, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ a **multi-index**, where each $\alpha_i \in \mathbb{N}_0$ and $|\alpha| = \alpha_1 + \cdots + \alpha_n = k$.

The multi-index derivative $D^\alpha u$ is defined as:

$$D^\alpha u(x) = \frac{\partial^{|\alpha|} u(x)}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \ldots \partial x_n^{\alpha_n}},$$

which is a partial derivative of total order k, where the variable x_i appears α_i times in the differentiation.

The notation $\int_\Omega \cdot\, dx$ stands for integration over all spatial coordinates $x = (x_1, \ldots, x_n)$, and is explicitly:

$$\int_\Omega f(x)\,dx = \int_\Omega f(x_1, \ldots, x_n)\,dx_1 dx_2 \ldots dx_n.$$

Thus, the expression

$$\int_\Omega D^\alpha u(x)\,\varphi(x)\,dx$$

means we apply the mixed partial derivative indicated by α to u, multiply the result by $\varphi(x)$, and integrate the product over all components $x_1, \ldots, x_n$ of the domain Ω.

We'll use partial integration to see

$$\int_\Omega \frac{\partial u}{\partial x_i}\varphi \ dx = [u\varphi]_{\partial\Omega} - \int_\Omega u\frac{\partial \varphi}{\partial x_i} \ dx, \quad \forall \varphi \in \mathcal{D}(\Omega). \tag{2.20}$$

As we noted a moment ago, the function φ is zero close enough to the boundary of Ω, particularly at the boundary itself, so the first term on the right-hand side vanishes, yielding

$$\int_\Omega \frac{\partial u}{\partial x_i} \ \varphi \ dx = -\int_\Omega u \ \frac{\partial \varphi}{\partial x_i} \ dx, \quad \forall \varphi \in \mathcal{D}(\Omega). \tag{2.21}$$

With this in mind, let us analyze what we can do with $\int_\Omega D^\alpha u\varphi \, dx$, first, we can go back to the definition of the k-th order derivative, changing

$$\int_\Omega D^\alpha u \ \varphi \ dx = \int_\Omega \frac{\partial^k u}{\partial x_1^{\alpha_1} \ldots x_n^{\alpha_n}} \ \varphi \ dx = \int_\Omega \frac{\partial}{\partial x_1^{\alpha_1}} \frac{\partial^{k-1} u}{\partial x_2^{\alpha_2} \ldots x_n^{\alpha_n}} \ \varphi \ dx. \tag{2.22}$$

Where we can apply Green's first identity to get:

$$\int_\Omega \frac{\partial}{\partial x_1^{\alpha_1}} \frac{\partial^{k-1} u}{\partial x_2^{\alpha_2} \ldots x_n^{\alpha_n}} \ \varphi \ dx = -\int_\Omega \frac{\partial^{k-1} u}{\partial x_2^{\alpha_2} \ldots x_n^{\alpha_n}} \ \frac{\partial \varphi}{\partial x_1^{\alpha_1}} \ dx. \tag{2.23}$$

Now we can apply the Green's first identity to each of the partial derivatives that make up $D^\alpha u$

$$\int_\Omega D^\alpha u \ \varphi \ dx = (-1)^k \int_\Omega u \ D^\alpha \varphi \ dx \quad \forall \varphi \in \mathcal{D}(\Omega), \tag{2.24}$$

for any $u \in C^k(\Omega)$. In the equation above, we can notice that only the right-hand side requires the strong regularity $\varphi \in C^\infty(\Omega)$. This way we can introduce the **space of locally integrable functions**

We denote by L^1_{loc}, the space of functions of $L^1(\Omega)$ with compact support K in Ω

$$L^1_{\text{loc}}(\Omega) = \left\{ v \mid v \in L^1(K), \ \forall K \subset\subset \Omega \right\} \tag{2.25}$$

Definition 19. Let $u \in L^1_{\text{loc}}(\Omega)$. If there exists a function $g \in L^1_{\text{loc}}(\Omega)$ such that

$$\int_\Omega g \ \varphi \ dx = (-1)^k \int_\Omega u \ D^\alpha \varphi \ dx \quad \forall \varphi \in \mathcal{D}(\Omega) \tag{2.26}$$

then we say that g is the **weak derivative** $D^\alpha u$ of u. This definition turns out to be well-defined given Hölder's inequality.

$$\left| \int_\Omega u \ D^\alpha \varphi \ dx \right| \leq \|u\|_{L^1(K)} \|D^\alpha \varphi\|_{L^\infty(K)} \leq \infty \tag{2.27}$$

where $K = \text{supp}(\varphi)$.

Definition 20. Let $u \in L^1_{\text{loc}}(\Omega)$ and suppose all weak derivatives $D^\alpha u$ with $|\alpha| \leq k$, where k is a non-negative integer, exist. We define the **Sobolev norm** of u as

$$\|u\|_{W^{p,k}(\Omega)} = \left(\sum_{|\alpha| \leq k} \|D^\alpha u\|^p_{L^p(\Omega)} \right)^{1/p}, \quad 1 \leq p < \infty. \tag{2.28}$$

The **Sobolev space** $W^{p,k}(\Omega)$ is then defined as

$$W^{p,k}(\Omega) = \left\{ u \in L^1_{\text{loc}}(\Omega) \mid \|u\|_{W^{p,k}(\Omega)} < \infty \right\}. \tag{2.29}$$

So that $W^{p,k}(\Omega)$ is the space of all functions u in $L^p(\Omega)$ whose derivatives $D^\alpha u$, for $|\alpha| \leq k$, also belong to $L^p(\Omega)$.

When $\partial\Omega$ is sufficiently regular, we can demonstrate that $C^\infty(\Omega)$ is dense in $W^{p,k}(\Omega)$ for $1 \leq p < \infty$. As a consequence, we can define $W^{p,k}(\Omega)$ as the completion of $C^\infty(\Omega)$ with respect to the Sobolev norm $\|\cdot\|_{W^{p,k}(\Omega)}$. In other words, $W^{p,k}(\Omega)$ contains all the limits of sequences in $C^\infty(\Omega)$ whose norm is finite.

The Sobolev space turns out to be a Banach space for $1 \leq p < \infty$. However, for $p = 2$, i.e., $W^{2,k}(\Omega)$, this is also a Hilbert space with an inner product given by:

$$(u, v)_{W^{2,k}(\Omega)} = \sum_{|\alpha| \leq k} (D^\alpha u, D^\alpha v)_{L^2}. \tag{2.30}$$

And with a norm given by:

$$\|u\|^2_{W^{2,k}(\Omega)} = \sum_{|\alpha| \leq k} \|D^\alpha u\|_{L^2}. \tag{2.31}$$

The case $p = 2$ is the most common in finite element analysis, and we will use the notation $H^k(\Omega) = W^{2,k}(\Omega)$ to emphasize that this is a Hilbert space. Therefore, for $k = 1$, we have the space:

$$W^{2,1}(\Omega) = H^1(\Omega) = \left\{ u \in L^2(\Omega) \mid D^1 u \in L^2(\Omega) \right\}. \tag{2.32}$$

with inner product and norm given by:

$$(u, v)_{H^1(\Omega)} = (u, v)_{L^2} + (\nabla u, \nabla v)_{L^2},$$
$$\|u\|^2_{H^1} = \|u\|^2_{L^2} + \|\nabla u\|^2_{L^2}.$$

The space $H^1_0(\Omega)$.

Speaking of the boundary $\partial\Omega$ is a delicate matter. This has to do with the fact that the Lebesgue integral is defined except perhaps on a set of measure zero, and $\partial\Omega$, evidently, has measure zero. This means that it is insignificant to speak of restricting a Sobolev function to the boundary of the domain, as we can even alter the values of the function on the boundary without altering its belonging to the space.

For this purpose, we must develop the concept of a **trace**, which roughly consists of making an approximation from within the domain and evaluating the approximation on the boundary, meaning that every $w \in W^{p,1}(\Omega)$ can be restricted to $v \in L^p(\partial\Omega)$. Conversely, every $v \in L^p(\partial\Omega)$ can be extended to $w \in W_1^p(\Omega)$.

Definition 21 (Trace Operator)**.** Let $\Omega \subset \mathbb{R}^n$ be a bounded domain with Lipschitz boundary $\partial\Omega$. The *trace operator*

$$\gamma \colon H^1(\Omega) \longrightarrow L^2(\partial\Omega)$$

is defined as the unique bounded linear operator that extends the classical restriction mapping, so that for every function $u \in H^1(\Omega)$ that is continuous up to the boundary,

$$\gamma u = u|_{\partial\Omega}.$$

Moreover, the trace operator is bounded in the sense that there exists a constant $C > 0$ (depending only on Ω) such that

$$\|\gamma u\|_{L^2(\partial\Omega)} \leq C \, \|u\|_{H^1(\Omega)} \quad \forall u \in H^1(\Omega).$$

This result, which is a consequence of the trace theorem (see, e.g., [11]), ensures that even if functions in $H^1(\Omega)$ are defined only in a weak sense, one can still meaningfully assign boundary values. The trace operator is well-defined and satisfies

$$\|\gamma v\|_{L^p(\partial\Omega)} \leq C \|v\|_{L^p(\Omega)}^{1-\frac{1}{p}} \|v\|_{L^p(\Omega)}^{\frac{1}{p}}. \tag{2.33}$$

The trace operator extends the usual restriction mapping of the continuous functions.

Using the trace operator, we can define the **Hilbert space with constraints**.

Definition 22.

$$W_0^{2,1}(\Omega) = H_0^1(\Omega) = \left\{ v \in H^1(\Omega) \mid \gamma v = 0 \right\} \tag{2.34}$$

which consists of all functions in $H^1(\Omega)$ that are zero in the trace operator sense on $\partial\Omega$. Commonly, the trace operator is omitted, and the restriction is simply written as $v|_{\partial\Omega}$.

The dual space of $H_0^1(\Omega)$ is denoted by $H^{-1}(\Omega)$. We can identify L^2 and its dual, but we should not identify H_0^1 and its dual as explained in Remark 2. We have then the inclusions

$$H_0^1 \subset L^2 \subset H^{-1},$$

where these injections are continuous and dense.

It is a standard result that the space $L^2(\Omega)$ is separable. In particular, the set of simple functions (or equivalently, the set of continuous functions with compact support, which is dense in $L^2(\Omega)$) can be shown to be countable in an appropriate sense. For a complete discussion, see [11].

Definition 23. The partial derivatives of a function $u : \mathbb{R}^n \to \mathbb{R}$ with respect to a variable $x_i \in \Omega \subseteq \mathbb{R}^n$ are commonly denoted as $\frac{\partial u}{\partial x_i}$, but also the notation u_{x_i} is used, or using the **"del" or "nabla" operator** ∇

$$\nabla u := \left(\frac{\partial u}{\partial x_1}, \dots, \frac{\partial u}{\partial x_n} \right).$$

The nabla operator can also be applied to a scalar function which yields the gradient of the function, i.e., $\nabla f = \text{grad } f$, or it can also be used with the dot product with a vector function in cartesian coordinates, which gives the divergence of the function, i.e., $\nabla \cdot \hat{f} = \text{div } \hat{f}$

Another operator related to the nabla operator is the Laplace operator Δ, which is generally defined in terms of the nabla operator as

$$\Delta u := \nabla \cdot \nabla u = \sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2}.$$

Theorem 9 (Green's first identity)**.** Let $u, v \in C^2(\Omega)$, then

$$\int_\Omega \nabla u \cdot \nabla v \, dx = -\int_\Omega v \Delta u \, dx + \int_{\partial\Omega} v \nabla u \cdot n \, dx \tag{2.35}$$

Where n is the outward normal vector to the domain.

Proof. First recall the following identity which resembles the Leibniz rule

$$\nabla \cdot (fg) = f \nabla g + \nabla f g$$

As well as Gauss's divergence theorem which states

$$\int_\Omega \nabla \cdot f \, dx = \int_{\partial\Omega} f \cdot n \, dx,$$

where n is the outward normal vector to the domain.
Then we get that

$$\int_{\partial\Omega} v \nabla u \cdot n \, dx = \int_\Omega \nabla(v \nabla u) \, dx = \int_\Omega v(\nabla \cdot \nabla u) \, dx + \int_\Omega \nabla v \cdot \nabla u \, dx.$$

Then, by the definition of $\Delta u := \nabla \cdot \nabla u$ we get

$$\int_\Omega \nabla u \cdot \nabla v \, dx = -\int_\Omega v \Delta u \, dx + \int_{\partial\Omega} v \nabla u \cdot n \, dx.$$

□

Proposition 10 (Poincaré's Inequality)**.** Let Ω be a bounded domain. Then there exists a constant C_Ω such that for each function $u \in H_0^1(\Omega)$,

$$\|u\|_{L^2} \leq C_\Omega \|\nabla u\|_{L^2}. \tag{2.36}$$

Proof. We are going to carefully choose a 'test function' φ such that $-\Delta\varphi = 1$ in Ω and with $\sup_{x\in\Omega} |\nabla\varphi(x)| \leq C$, for an arbitrary $C \in \mathbb{R}$. Such functions exists if the boundary of Ω is sufficiently smooth.
By means of the Green's identity, we have

$$\begin{aligned}\int_\Omega u^2 \ dx &= -\int_\Omega u^2 \Delta\varphi \ dx \\ &= -\int_{\partial\Omega} u^2 \nabla\varphi \cdot n \ ds + \int_\Omega 2u\nabla u \cdot \nabla\varphi \ dx \\ &= \int_\Omega 2u\nabla u \cdot \nabla\varphi \ dx\end{aligned}$$

since $u = 0$ on ∂u.
Furthermore, using the Cauchy-Schwartz inequality we have

$$\|u\|_{L^2}^2 = \int_\Omega u^2 \ dx = \int_\Omega 2u\nabla u \cdot \nabla\varphi \ dx \leq 2 \max_{x\in\Omega}\{|\nabla\varphi(x)\,, \|u\|_{L^2}\} \, \|\nabla u\|_{L^2} \tag{2.37}$$

Finally, dividing by $\|u\|_{L^2}$ on both sides, we get the desired result. □

Notice that by Poincare's inequality we have for all $u \in H_0^1(\Omega)$

$$\|\nabla u\|_{L^2}^2 \leq \|u\|_{H^1(\Omega)}^2 = \|u\|_{L^2}^2 + \|\nabla u\|_{L^2}^2 \leq C_\Omega \|\nabla u\|_{L^2}^2 \tag{2.38}$$

Hence we can define as well $\|u\|_{H_0^1} := \|\nabla u\|_{L^2}$ which is an equivalent norm to $\|u\|_{H^1(\Omega)}$ on $H_0^1(\Omega)$. Further it is induced by the scalar product

$$(u, v)_{H_0^1(\Omega)} = \int_\Omega \nabla u \cdot \nabla v \quad \forall u, v \in H_0^1(\Omega)$$

2.3 Base for $H_0^1(\Omega)$

We are interested in describe the construction of a basis for the Sobolev space $H_0^1(\Omega)$, this is fundamental both for theoretical analysis and for the practical implementation of numerical methods. The goal is to show that $H_0^1(\Omega)$, being a separable Hilbert space,admits an orthonormal basis. In particular, by considering the eigenvalue problem associated with the negative Laplacian under homogeneous Dirichlet conditions, one obtains a sequence of eigenfunctions that is complete in $L^2(\Omega)$. By normalizing these eigenfunctions with respect to the H_0^1-norm, we derive an orthonormal basis for $H_0^1(\Omega)$. This basis is subsequently useful in developing numerical schemes such as the finite element method and variational approaches. Moreover, the spectral decomposition that yields this Hilbert basis will be implemented in formulating robust variational physics-informed neural networks, where the error estimate are naturally expressed in terms of the coefficients in this basis.

Definition 24. Let $(E_n)_{n\geq 1}$ be a sequence of closed sub-spaces of H. One says that H is the **Hilbert sum** of the $(E_n)_{n\geq 1}$ and one writes $H = \bigoplus_n E_n$ if

i) the spaces E_n are mutually orthogonal, i.e.

$$(u, v) = 0 \quad \forall u \in E_n,\ \forall v \in E_m,\ m \neq n,$$

ii) The linear space spanned by $\bigcup_{n=1}^{\infty} E_n$ is dense in H.

Definition 25. A sequence $(e_n)_{n\geq 1}$ in H, Hilbert space, is said to be an **orthonormal basis** of H (or a Hilbert basis, or simply a basis when there is no confusion) if it satisfies the following properties:

i) $\|e_n\|_H = 1$, and $(e_n, e_m) = 0$ for all $n \neq m \in \mathbb{N}$.

ii) The linear space spanned by the $e_n's$ is dense in H.

Moreover let $(e_n)_{n\in\mathbb{N}}$ be a orthonormal basis. Then for every $u \in H$, we have

$$u = \sum_{k=1}^{\infty} (u, e_k)e_k, \quad \text{i.e.}$$

$$u = \lim_{n\to\infty} \sum_{k=1}^{n} (u, e_k)e_k \quad \text{and}$$

$$\|u\|_H^2 = \sum_{k=1}^{\infty} |(u, e_k)|^2.$$

Definition 26. We define a **compact operator** on a Banach space E to be an operator T such that for any bounded sequence $\{f_n\} \in E$, the sequence $\{\|Tf_n\|_E\}$ has a convergent sub-sequence. By $\mathcal{K}(E)$ we denote all the compact operators from E to E.

For a complete treatment of compact operators, refer to *Davies* [19], pages 75 and 76.

Definition 27. We denote by $\mathcal{L}(H) = \{T : H \to H \mid T \text{ is a linear operator}\}$ A bounded operator $Y \in \mathcal{L}(H)$ is said to be **self-adjoint** if $T^* = T$, i.e.,

$$(Tu, v) = (u, Tv), \quad \forall u,\ v \in H. \tag{2.39}$$

Definition 28. Let $T \in \mathcal{L}(E)$ where E is a normed vector space.
The **resolvent set,** denoted by $\rho(T)$, is defined by

$$\rho(T) = \{\lambda \in \mathbb{C} \mid (T - \lambda I) \text{ is bijective from } E \text{ onto } E\} \tag{2.40}$$

The **point spectrum,** denoted by $\sigma(T)$, is the complement of the resolvent set, i.e., $\sigma(T) = \mathbb{C} \setminus \rho(T)$.
A real number λ is said to be an **eigenvalue** of T if

$$\ker(T - \lambda I) \neq \{0\} \tag{2.41}$$

$\ker(T - \lambda I)$ is the corresponding **eigenspace.** The set of all eigenvalues is denoted by $EV(T)$.

Lemma 11. Let $T \in \mathcal{L}$ be a self-adjoint operator such that $\sigma(T) = \{0\}$. Then $T = 0$

This lemma can be found in *Brezis* [11], Corollary 6.10.

Theorem 12. Let H be a separable Hilbert space and let T be a compact self-adjoint operator. Then there exists a Hilbert basis composed of eigenvalues of T.

Proof. Let $(\lambda_n)_{n\geq 1}$ be a sequence of all (distinct) nonzero eigenvalues of T. Set

$$\lambda_0 = 0, \quad E_0 = \ker(T), \text{ and } E_n = \ker(T - \lambda_n I).$$

The claim is that H is the Hilbert sum of the $E_n's$, $n = 0, 1, 2 \ldots$
First of all, let us see that the spaces E_n are mutually orthogonal.
Take $u \in E_m$ and $v \in E_n$ with $m \neq n$, then

$$Tu = \lambda_m u \text{ and } Tv = \lambda_n v,$$

so that

$$(Tu, v) = \lambda_m(u, v) = (u, Tv) = \lambda_n(u, v) \tag{2.42}$$

by the assumption that T is a self-adjoint operator. Since λ_m, λ_n are both non-zero different real numbers, we conclude that

$$(u, v) = 0$$

Now, let us see that the vector space spanned by the spaces E_n is dense in E. let us denote by $F = \text{Span}\{E_n\}_{n\in\mathbb{N}}$.
Clearly, $T(F) \subset F$. Therefore $T(F^\perp) \subset F^\perp$; Indeed, given $u \in F\perp$

$$(Tu, v) = (u, Tv) = 0 \quad \forall v \in F,$$

so that $Tu \in F^\perp$. The operator T restricted to $F^\perp$ is denoted by T_0. This is a self-adjoint compact operator on $F^\perp$. We claim that $\sigma(T_0) = \{0\}$. let us suppose it is not; suppose that some $\lambda \neq 0$ belongs to $\sigma(T_0)$. Since $\lambda \in EV(T_0)$, there's some $u \in F^\perp$, $u \neq 0$, sucha that $T_0 u = \lambda u$. Therefore, λ is one of the eigenvalues of T, say $\lambda = \lambda_n$ for some $n \geq 1$. Thus $u \in E_n \subset F$. Which is a contradiction since $F \cap F^\perp = \{0\}$.
Applying lemma 11, we deduce that $T_0 = \{0\}$, which means that T is zero in $F^\perp$. It follows that $F^\perp \subset \ker(T)$. On the other hand, $\ker(T) \subset F$ and consequently $F^\perp \subset F$. Which implies that $F^\perp = \{0\}$, and so F is dense in H.
Finally we choose a Hilbert basis in each space E_n and with them we construct a Hilbert basis by taking the union of these bases. □

Theorem 13 (*Brezis* [11], Theorem 9.31)**.** There exists a Hilbert basis $(\varphi_n)_{n\geq 1}$ of $L^2(\Omega)$ and a sequence $(\lambda_n)_{n\geq 1}$ of real numbers such that $\lambda_n > 0$ and $\lambda_n \to +\infty$. Also

$$\varphi_n \in H_0^1(\Omega),$$
$$-\Delta\varphi_n = \lambda_n \varphi_n \text{ in } \Omega.$$

The $\lambda_n's$ are called the eigenvalues of $-\Delta$ (with Dirichlet boundary condition) and the $\varphi_n's$ are called the associate eigenfunctions.

Proof. Given $f \in L^2(\Omega)$, let $Tf := u$ be the unique solution $u \in H_0^1(\Omega)$ to the variational problem

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega fv \, dx \quad \forall v \in H_0^1(\Omega)$$

Define

$$a(u, v) := \int_\Omega \nabla u \cdot \nabla v \, dx,$$

and

$$l(v) := \int_\Omega fv \, dx,$$

therefore

$$\begin{aligned} T : L^2(\Omega) &\longrightarrow L^2(\Omega) \\ f &\longmapsto Tf := u, \end{aligned}$$

and in general

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega fv \, dx \quad \forall v \in H_0^1(\Omega).$$

Since by hypothesis $Tf := u$

$$\int_\Omega \nabla Tf \cdot \nabla v \, dx = \int_\Omega fv \, dx \quad \forall v \in H_0^1(\Omega),$$

therefore

$$(Tf, v)_{H^1(\Omega)} = (f, v)_{L^2}.$$

Notice that T is well defined by virtue of the Lax-Milgram theorem (7.1).
Moreover, we affirm that T is a *self-adjoint* operator, indeed, consider f_1 and $f_2 \in L^2(\Omega)$, then:

$$(Tf_1, Tf_2)_{H_0^1} = (\nabla Tf_1, \nabla Tf_2)_{L^2} = a(Tf1, Tf_2) = l_{f_1}(Tf_2) = (f_1, Tf_2)_{L^2}.$$

Similarly

$$(Tf_1, Tf_2)_{H_0^1} = (\nabla Tf_1, \nabla Tf_2)_{L^2} = a(Tf_1, Tf_2) = a(Tf_2, Tf_1) = l_{f_2}(Tf_1) = (Tf_1, f_2)_{L^2},$$

from which we get

$$(Tf_1, f_2)_{L^2} = (Tf_1, Tf_2)_{H_0^1} = (f_1, Tf_2)_{L^2}.$$

Now, let us turn our attention to the fact that T turns out to be an *compact* operator. Consider $u \in H_0^1(\Omega)$ to be any solution of the variational problem, then by Poincaré's inequality:

$$\begin{aligned} \|u\|_{H_0^1}^2 &= \int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega fv \, dx \\ &\leq \|u\|_{L^2} \|f\|_{L^2} \leq C_\Omega \|\nabla u\|_{L^2} \|f\|_{L^2} = C_\Omega \|u\|_{H_0^1} \|f\|_{L^2}, \end{aligned}$$

thus

$$\|u\|_{H_0^1} \leq C_\Omega \|f\|_{L^2}, \tag{2.43}$$

consequently

$$\|Tf_n\|_{H_0^1} \leq C_\Omega \|f\|_{L^2}. \tag{2.44}$$

Given a bounded sequence $(f_n)_{n\in\mathbb{N}} \subset H_0^1(\Omega)$ the point is to prove that $(\|Tf_n\|_{H_0^1(\Omega)})$ has a convergent sub-sequence. Which is clear by the Bolzano Weierstrass theorem ([11]), since $\|f_n\|_{L^2} \leq C \quad \forall n \in \mathbb{N}$ and by 2.44

$$\|Tf_n\|_{H_0^1} \leq C_\Omega \|f\|_{L^2} \leq C_\Omega C = \hat{C}$$

So we have the compactness of T.

On the other hand, $\ker(T) = \{0\}$ because a and l_f are both positive definite as they come from inner products on $L^2(\Omega)$. Therefore by Theorem 12, $L^2(\Omega)$ admits a Hilbert basis (φ_n) consisting of eigenfunctions of T associated to eigenvalues $(\mu_n)_{n=1}^\infty$ with $\mu_n := \frac{1}{\lambda_n} > 0 \quad \forall n$ and $\mu_n \to \infty$ (For more detail, check *Davies*, [19], Theorem 4.2.2).

The key insight is that T is the inverse of the Laplacian operator $-\Delta$, i.e. $T = (-\Delta)^{-1}$.

It maps $\lambda_n\varphi_n \in H^1(\Omega)$ to φ_n because by hypothesis and (9):

$$-\Delta\varphi_n = \lambda_n\varphi_n$$

so then

$$\int_\Omega -\Delta\varphi_n \, v dx = \int_\Omega \lambda_n\varphi_n \, v \, dx,$$

therefore:

$$\int_\Omega \nabla\varphi_n \cdot \nabla v \, dx = \int_\Omega \lambda_n\varphi_n \, v \, dx,$$

which is:

$$\varphi_n = T(\lambda_n\varphi_n)$$

from which we can affirm,

$$T(\varphi_n(x)) = \mu_n\varphi_n,$$

then

$$\lambda_n^{-1} \int_\Omega \nabla\varphi_n \cdot \nabla v \, dx = \int_\Omega \varphi_n v \, dx.$$

This implies that

$$\lambda_n^{-1}(\varphi_n(x), v)_{H^1(\Omega)} = (\varphi_n(x), v)_{L^2} = (T\varphi_n(x), v)_{H^1(\Omega)} \tag{2.45}$$

Thus we have that $\varphi_n \in H_0^1(\Omega)$ and since $T\varphi_n = \mu_n\varphi_n$

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega \nabla T\varphi_n \cdot \nabla v \, dx = \int_\Omega \nabla \mu_n\varphi_n \cdot \nabla v \, dx = \int_\Omega \varphi_n v \, dx \quad \forall v \in H_0^1(\Omega),$$

$$\int_\Omega \nabla\varphi_n \cdot \nabla v \, dx = \lambda_n \int_\Omega \varphi_n v \, dx \quad \forall v \in H_0^1(\Omega).$$

In other words, $\varphi_n(x)$ is a weak solution of the variational problem with $\lambda_n = \frac{1}{\mu_n}$.

Finally observe that, taking $v = \varphi_n(x)$, yields:

$$\|\varphi_n(x)\|^2_{H^1} = (\varphi_n, \varphi_n)_{H^1} = \lambda_n(\varphi_n, \varphi_n)_{L^2}$$

We know that $(\varphi_n)_{n=1}^{\infty}$ is an orthonormal basis for $L^2(\Omega)$ therefore

$$(\varphi_n, \varphi_n)_{L^2} = 1$$

Then:

$$\|\varphi_n(x)\|^2_{H^1} = \lambda_n$$

□

Under this assumptions of Theorem 13, the sequence

$$(\varphi_n/\sqrt{\lambda_n})_{n=1}^{\infty}$$

is a Hilbert basis of $H_0^1(\Omega)$ when we equip this space with the scalar product $(u, v)_{H^1\Omega} = \nabla u \cdot \nabla v$,

The functions $(\varphi_n/\sqrt{\lambda_n + 1})_{n=1}^{\infty}$ are a Hilbert basis for $H_0^1(\Omega)$ equipped with the scalar product $\int_\Omega (\nabla u \cdot \nabla v + uv)$.

The orthonormality follows from 13 by taking $v = e_k$. To see that indeed (φ_n) is dense in $H_0^1(\Omega)$, for this take $f \in H_0^1(\Omega)$ be such that $(f, \varphi_n)_{H_0^1} = 0$ for all n. We have to prove that $f = 0$. From 13 we have that $\lambda_n \int \varphi_n f = 0$ for all n and in consequence $f = 0$ because (φ_n) form a base of $L^2(\Omega)$.

Being established in theorem (12) the base for $L^2(\Omega)$ we can construct a basis for the dual space $H^{-1}(\Omega)$.

One of the main advantages of having at hand a Hilbert basis is that the orthonormality let us calculate norms in Hilbert spaces.

Theorem 14 (Generalized Parseval's Identity)**.** Given H, a Hilbert space with an orthonormal basis $\{\varphi_k\}_{k\geq 1}$. For any $x \in H$,

$$\|x\|^2 = \sum_{k=1}^{\infty} |(x, \varphi_k)|^2 .$$

Proof. Let H be a Hilbert space with an orthonormal basis $\{\varphi_k\}_{k\geq 1}$. For any $x \in H$, define the partial sum

$$x_N = \sum_{k=1}^{N} (x, \varphi_k)\, \varphi_k.$$

Since the φ_k are orthonormal, by Bessel's inequality, we have

$$\|x_N\|^2 = \sum_{k=1}^{N} |(x, \varphi_k)|^2 \leq \|x\|^2.$$

Because $\{\varphi_k\}$ is a basis, the span of $\{\varphi_k\}$ is dense in H; hence, as $N \to \infty$ the partial sums x_N converge in norm to x, that is,

$$\lim_{N\to\infty} \|x - x_N\| = 0.$$

Now, using the orthogonal decomposition (or the Pythagorean theorem), we have

$$\|x\|^2 = \|x_N\|^2 + \|x - x_N\|^2.$$

Taking the limit as $N \to \infty$ and noting that $\|x - x_N\| \to 0$, we obtain

$$\|x\|^2 = \lim_{N\to\infty} \|x_N\|^2 = \lim_{N\to\infty} \sum_{k=1}^{N} |(x, \varphi_k)|^2 = \sum_{k=1}^{\infty} |(x, \varphi_k)|^2 .$$

□

In general, consider a real, separable Hilbert space H with inner product $(\cdot,\cdot)_H : H \times H \to \mathbb{R}$, and $\{\varphi_n(x)\}_{n=1}^{\infty}$ a countable, orthogonal basis of H, with $\|\varphi_n(x)\|_H^2 := \lambda_n$. And consider $R \in H^*$

By the Riesz representation theorem (5), there exists a unique solution $u_R \in H$ of

$$(u_R, v)_H = R(v), \qquad \forall R \in H,$$

that satisfies

$$\|u_R\|_H = \|R\|_{H^*}.$$

Consequently, the mapping $f \mapsto u_R$ is an isometry. As $(\varphi_n)_{n=1}^{\infty}$ is an orthogonal basis of H, we then have by Parseval's identity (6.13):

$$\|R\|_{H^*} = \|u_R\|_H = \left(\sum_{n=1}^{\infty} \frac{(eu_R, \varphi_n)_H^2}{\|\varphi_n\|_H^2} \right)^{\frac{1}{2}} = \left(\sum_{n=1}^{\infty} \lambda_n^{-1} R(\varphi_n)^2 \right)^{\frac{1}{2}}. \tag{2.46}$$

For a computational approach, we need to approximate the dual norm of elements $R \in H^*$ using a truncated version of this series expansion. To do so, we need an appropriate orthogonal basis $(\varphi_n)_{n=1}^{\infty}$ of H.

Let us restrict ourselves to problems where the space of test functions is $H_0^1(\Omega)$. For this we take the orthogonal basis of $H_0^1(\Omega)$ to be the weak solution of:

$$\lambda_k \int_\Omega v(x)\varphi_n(x)\,dx = \int_\Omega \nabla\varphi_n(x) \cdot \nabla v(x) + \varphi_n(x)v_n(x)\,dx \qquad \forall v \in H_0^1(\Omega),$$
$$\|\varphi_n\|_{L^2} = 1.$$

In strong form, we may write the eigenvalues problem as

$$-\Delta\,\varphi_n(x) = \lambda_n \varphi_n(x),$$

where φ_n are the eigenvectors of the operator $1-\Delta$ with corresponding eigenvalues λ_n, and normalized to have unit norm $L^2(\Omega)$. The key properties that we derived in the Section 2.2 are that the set $(\varphi_n)_{n=1}^{\infty}$ forms an orthogonal basis of $H_0^1(\Omega)$ and an orthonormal basis for $L^2(\Omega)$. Also, we showed that $\|\varphi_n\|_{H^1(\Omega)}^2 = \lambda_n$ where λ_n is the eigenvalue of $-\Delta$ corresponding to φ_n and that $\lambda_n \geq 1$ and form a non-decreasing and unbounded sequence.

As before, we notice that f and u_f are related via $(1-\Delta)u_f = f$, and we may then state that

$$\|f\|_{H^{-1}} = \|(-\Delta)^{-1} f\|_{H^1},$$

where the series expression (2.46) allow us to evaluate the H^1-norm in a straightforward manner.

We apply the previous theory to the space $H_0^1([0,1])$ and the Laplace operator $-\Delta$ with homogeneous Dirichlet boundary conditions.

2.3.1 One dimensional setting

Let us focus our attention on the domain $\Omega = (0,1) \subseteq \mathbb{R}$. By Theorem 12, since $-\Delta$ is compact and self-adjoint, there exists a Hilbert basis $\{\varphi_n\}$ of $L^2([0,1])$ consisting of eigenfunctions of T with corresponding eigenvalues $\{\mu_n\}$, where $\mu_n > 0$ and $\mu_n \to 0$.

For each φ_n, we have

$$T\varphi_n = \mu_n \varphi_n.$$

Applying T to φ_n and using the variational formulation,

$$\int_0^1 \nabla(\mu_n \varphi_n) \cdot \nabla v \, dx = \int_0^1 \varphi_n v \, dx \quad \forall v \in H_0^1([0,1]).$$

Simplifying,

$$\mu_n \int_0^1 \nabla \varphi_n \cdot \nabla v \, dx = \int_0^1 \varphi_n v \, dx.$$

This implies that φ_n satisfies the eigenvalue problem

$$-\Delta \varphi_n = \lambda_n \varphi_n \quad \text{in } (0,1), \quad \varphi_n(0) = \varphi_n(1) = 0,$$

where $\lambda_n = \frac{1}{\mu_n}$.

In one dimension, the Laplacian Δ reduces to the second derivative:

$$-\varphi_n''(x) = \lambda_n \varphi_n(x) \tag{2.47}$$

this is a second-order linear ordinary differential equation:

$$\varphi_n''(x) + \lambda_n \varphi_n(x) = 0.$$

The characteristic equation for (2.47) is

$$r^2 + \lambda_n = 0 \qquad \text{with solution} \quad r = \pm i\sqrt{\lambda_n}.$$

The general solution is

$$\varphi_n(x) = A\cos(\sqrt{\lambda_n}\,x) + B\sin(\sqrt{\lambda_n}\,x).$$

To determine the constants A and B, we make use of the boundary conditions $\varphi_n(0) = \varphi_n(1) = 0$ to obtain that $A = 0$ and:

$$\varphi_n(x) = B\ \sin(\sqrt{\lambda_n}\,x).$$

From the fact that $\varphi_n(1) = 0$ we can see then:

$$B\ \sin(\sqrt{\lambda_n}\,1) = 0,$$

since we seek a non-trivial solution, $B \neq 0$ and

$$\sin(\sqrt{\lambda_n}) = 0 \quad \to \sqrt{\lambda_n} = n\pi.$$

Thus the eigenvalues are

$$\lambda_n = (n\pi)^2, \qquad n \in \mathbb{N}$$

To determine the eigenfunctions, we can replace $\sqrt{\lambda_n} = n\pi$ into the simplified general solution of the linear ordinary problem:

$$\varphi_n(x) = B\ \sin(n\pi\,x),$$

since we need this basic function to be normalized, it is

$$\begin{aligned} \|\varphi_n\|_{L^2} &= 1, \\ \int_0^1 |\varphi_n(x)|^2\,dx &= 1, \\ B^2\int_0^1 \sin^2(n\pi x)\,dx &= 1. \end{aligned}$$

As

$$\int_0^1 \sin^2(n\pi\,x)\,dx = \frac{1}{2},$$

then, $B^2\,\frac{1}{2} = 1$ with $B = \sqrt{2}$. Thus, the normalized eigenfunctions are:

$$\varphi_n(x) = \sqrt{2}\ \sin(n\pi\,x).$$

In the original problem, the operator T is the inverse of the Laplacian $(-\Delta)^{-1}$, therefore:

$$T\,\varphi_n = \mu_n\varphi_n, \quad \text{and} \quad \mu_n = \frac{1}{\lambda_n} = \frac{1}{(n\pi)^2}.$$

From which as $n \to \infty$ and $\mu_n \to 0$ ensure that T is compact, the self-adjointness comes naturally from the symmetry of the Laplacian in the variational formulation.

So then, in the interval $(0,1)$, the eigenfunctions and eigenvalues are explicitly given by

$$\varphi_n(x) = \sqrt{2}\sin(n\pi x), \quad \text{and} \quad \lambda_n = (n\pi)^2, \quad n = 1, 2, 3, \ldots.$$

The set $\{\varphi_n\}$ forms an orthonormal basis for $L^2([0,1])$. Since each $\varphi_n \in H_0^1([0,1])$ and the eigenvalues $\lambda_n \to \infty$, the functions $\{\varphi_n\}$ also form a basis for $H_0^1([0,1])$. Any function $u \in H_0^1([0,1])$ can be expressed as

$$u(x) = \sum_{n=1}^{\infty} c_n \varphi_n(x),$$

with convergence in the $H_0^1([0,1])$ norm, where the coefficients c_n are given by

$$c_n = \int_0^1 u(x)\varphi_n(x)\, dx.$$

These functions can be stated in a more general setting. Now, to construct a basis function in an interval $[a,b] \subseteq \mathbb{R}$, we should define:

$$\varphi_n(x) := \sqrt{\frac{2}{b-a}}\, \sin\left(n\frac{\pi(x-a)}{b-a}\right) \tag{2.48}$$

with the derivative given by:

$$d_x\, \varphi_n(x) = \sqrt{\frac{2}{b-a}}\, \frac{\pi\, n}{b-a}\, \cos\left(n\frac{\pi(x-a)}{b-a}\right), \tag{2.49}$$

which are not only orthogonal but also normalized in the space $L^2([a,b])$. A straightforward calculation shows that:

$$\int_a^b \varphi_n^2(x)\, dx = 1.$$

Now, for the space $H^1(\Omega)$ we can notice that:

$$\|\varphi_n(x)\|_{H^1}^2 = \|\varphi_n(x)\|_{L^2} + \|\varphi_n(x)\|_{H_0^1}^2,$$

but

$$\|\varphi_n(x)\|_{L^2}^2 = 1$$
$$\|\varphi_n(x)\|_{H_0^1}^2 = \|d_x\, \varphi_n(x)\|_{L^2}^2 = \int_a^b (d_x\, \varphi_n(x))^2\, dx = \frac{\pi^2\, n^2}{(b-a)^2}$$

Therefore:

$$\|\varphi_n(x)\|_{H^1}^2 = 1 + \frac{\pi^2\, n^2}{(b-a)^2},$$

so in order to normalize in $H^1(\Omega)$ we must divide by:

$$\sqrt{1+\frac{\pi^2\,n^2}{(b-a)^2}}.$$

In the case where we are building a base for $H_0^1([a,b])$, notice that by the definition of the norm, we have already shown:

$$\|\varphi_n(x)\|_{H_0^1}^2 - = \|d_x\,\varphi_n(x)\|_{L^2}^2 = \int_a^b (d_x\,\varphi_n(x))^2\,dx = \frac{\pi^2\,n^2}{(b-a)^2}.$$

To normalize, one divides by

$$\frac{\pi\,n}{b-a}. \tag{2.50}$$

The basic elements are summarized in the following table:

Summary on Basis functions	
Space	**Basic Element**
$L^2([a,b])$	$\sqrt{\frac{2}{b-a}}\,\sin\left(n\frac{\pi(x-a)}{b-a}\right)$
$H^1([a,b])$	$\sqrt{\frac{2(b-a)}{1+\pi^2n^2}}\,\sin\left(n\frac{\pi(x-a)}{b-a}\right)$
$H_0^1([a,b])$	$\sqrt{\frac{2(b-a)}{\pi^2n^2}}\,\sin\left(n\frac{\pi(x-a)}{b-a}\right)$

Chapter 3

Partial Differential Equations

In this chapter, our primary objective is to formulate the variational residual associated with a partial differential equation and to describe a framework for approximating its H^{-1}-norm. Given a well-posed variational formulation of a PDE, one can define a residual operator that quantifies the discrepancy between a candidate solution and the exact weak solution. The measure for this discrepancy can be measured as the dual norm of the residual, which is defined on $H^{-1}(\Omega)$, the dual of the test space $H_0^1(\Omega)$.

The central idea is to approximate the H^{-1}- norm of the residual by expanding it in the terms of an orthonormal basis of $H_0^1(\Omega)$. By making use of the Riesz representation theorem (5), we obtain a unique representative in $H_0^1(\Omega)$ whose norm is equivalent to the H^{-1}-norm of the residual. In a practical setting, since these spaces are of infinite dimension, one has to truncate the infinite expansion to a finite number of basis elements, giving back a discrete error estimator.

3.1 Partial Differential Equations

A differential equation describes the relationship between a function and its derivatives. When this function depends on several independent variables, we encounter the realm of partial differential equations (PDEs). In contrast, a **ordinary differential equation**, (ODE) involves a function of a single variable. A **partial differential equation** (PDE) involves an unknown function $F : \Omega \to \mathbb{R}$ of multiple variables, incorporating their partial derivatives.

Similarly to ordinary derivatives, partial derivatives are defined through limits. Formally, for an open set $\Omega \subset \mathbb{R}^d$ and a function $F : \Omega \to \mathbb{R}$, the partial derivative of F at $x = (x_1, \cdots, x_d) \in \Omega$ with respect to x_i is:

$$\frac{\partial}{\partial x_i} F(x) = \lim_{\Delta x \to 0} \frac{F(x_1, \cdots, x_i + \Delta x, \cdots, x_d) - F(x)}{\Delta x}. \tag{3.1}$$

A function F is said to be continuously differentiable, or $C^1(\Omega)$, if all its partial derivatives exist and are continuous within a neighborhood of each point $x \in \Omega$.

Recall the **multi-index** notation which is an array $\alpha = (\alpha_1, \ldots, \alpha_n)$ of non-negative integers, with its order defined as

$$|\alpha| = k = \sum_{i=1}^{n} \alpha_i.$$

Using multi-indices, higher-order partial derivatives can be compactly represented as

$$D^\alpha u(x) := \frac{\partial^k u(x)}{\partial x_1^{\alpha_1} \cdots \partial x_n^{\alpha_n}}.$$

Symbolically, a typical PDE can be expressed in a standardized form. For an open set $\Omega \subset \mathbb{R}^d$, and a function $F : \mathbb{R}^{d^k} \times \mathbb{R}^{d^{k-1}} \times \cdots \times \mathbb{R}^d \times \mathbb{R} \times \Omega \to \mathbb{R}$, we define:

Definition 29. An equation of the form

$$F(D^k u(x), D^{k-1} u(x), \ldots, Du(x), u(x), x) = 0, \quad x \in \Omega, \tag{3.2}$$

where $u : \Omega \to \mathbb{R}$ is the unknown function, is called a k-th order partial differential equation.

3.1.1 Classification of PDE's

Solving a PDE involves finding all functions u that satisfy (3.2), often subject to additional boundary conditions on parts of the boundary $\partial\Omega$. When exact solutions are elusive, the focus shifts to establishing the existence and properties of these solutions.

Second-order PDEs are classified as elliptic, parabolic, or hyperbolic. This classification informs us about the nature of the physical phenomena they model:

- **Elliptic equations** typically describe steady states, often representing equilibrium conditions in physical systems.
- **Parabolic equations** are used to model time-dependent processes that evolve towards a steady state.
- **Hyperbolic equations** capture wave propagation and other transport phenomena.

The degree of a PDE is defined by the highest order of its derivatives in the equation. For a scalar second-order linear PDE in d variables $x = (x_1, \ldots, x_d)$, we have:

$$C \cdot D^2 u + b \cdot Du + au = f \quad \text{in } \Omega, \tag{3.3}$$

or equivalently:

$$-\sum_{i,j=1}^{d} c_{ij} \frac{\partial}{\partial x_i} \left(\frac{\partial u}{\partial x_j} \right) + \sum_{i=1}^{d} b_i \frac{\partial u}{\partial x_i} + au = f \quad \text{in } \Omega, \tag{3.4}$$

where $a(x) \in \mathbb{R}$, $b(x) \in \mathbb{R}^d$, and $C(x) \in \mathbb{R}^{d\times d}$ are the coefficients of the equation. When these coefficients do not depend on x, the PDE is said to have constant coefficients. To ensure the existence of classical solutions, regularity conditions must be met: $u \in C^2(\Omega)$, $c_{ij} \in C^1(\Omega)$, $b_i \in C^1(\Omega)$ and $a \in C^0(\Omega)$. These conditions can be relaxed in the weak formulation of the PDE.

Definition 30 (Classification of PDEs)**.** Consider a second-order linear PDE of the form (3.3) with a symmetric coefficient matrix $C(x)$. The equation is classified as:

1. **Elliptic** at $x \in \Omega$ if $C(x)$ is positive definite, i.e., for all $v \neq 0 \in \mathbb{R}^d$, $v^tCv > 0$.
2. **Parabolic** at $x \in \Omega$ if $C(x)$ is positive semidefinite ($v^tCv \geq 0$ for all $v \in \mathbb{R}^d$), not positive definite, and the rank of $C(x), b(x), a(x)$ equals d.
3. **Hyperbolic** at $x \in \Omega$ if $C(x)$ has one negative and $n-1$ positive eigenvalues.

An equation is called elliptic, parabolic, or hyperbolic on the open set Ω if it maintains that classification throughout Ω.

Remark 3. In practice, we distinguish between time-dependent and time independent PDEs. Elliptic equations are time-independent, describing steady-state situations.

The concept of well-posedness, due to J. Hadamard (1865-1963), is related to the requirements that can be expected from solving a partial differential equation.

Definition 31 (Hadamard's Well-posedness)**.** A given problem for a partial differential equation is said to be well-posed if:

1. A solution exists,
2. the solution is unique,

otherwise it is ill-posed.

If a problem is well-posed, obtaining a numerical approximation of the exact solution seems possible as long as the data to the problem are approximated suitable. some problems are ill-posed because of their nature, despite the initial and boundary conditions being correctly defined.

The notion of continuous dependence on the data indicates that the solution should not have to change much if the data are slightly perturbed. This requirement is important because in applications, the boundary data are usually obtained through measurement and may be noisy and small measurement errors should not affect dramatically the solution.

3.2 Weak solution and regularity

The concept of a well-posed problem in the context of partial differential equations (PDEs) lacks specificity regarding the nature of the unique solution. Specifically, it does not clarify whether the solution u should be infinitely differentiable. While having an smooth solution might be ideal, it is often an unrealistic expectation. A more practical approach is to require that a solution to a PDE of order k be at least k times continuously differentiable. This ensures that all derivatives in the equation exist and are continuous, defining what is known as a **classical** solution to the PDE.

However, not all PDEs admit classical solutions. For many equations, classical solutions may not exist. Such equations often govern fluid dynamics problems where shock waves develop and propagate, introducing discontinuities in the solution. In these cases, it is physically appropriate to allow for solutions that are not continuously differentiable or even continuous. Therefore, we seek **generalized** or **weak** solutions, which relax the requirements for differentiability.

Given this context, it becomes important to separate the concepts of **existence** and **regularity** of solutions. One can define a **weak solution** for a given PDE, which may lack regularity, and then focus on proving its existence, uniqueness, and continuous dependence on the data. The existence of weak solutions often relies on simpler estimates, while their regularity might depend on more intricate ones. In some exceptional cases, it is possible to find explicit solutions in terms of boundary values, but such instances are rare.

Typically, proving the existence of solutions involves indirect methods, such as deriving a contradiction from the assumption of nonexistence. However, nonconstructive techniques used in existence theorems are often not practical for computing approximations of the solution. To address this, we can seek methods that facilitate the computation of approximations. This approach involves initially proving an existence theorem within an abstract mathematical framework and subsequently developing a stable numerical scheme based on this theoretical result to approximate the solution effectively.

3.3 Poisson Equation

The Poisson equation is a partial differential equation of significant importance in mathematical physics, named after the French mathematician Siméon Denis Poisson (1781 - 1840) [33]. It is a generalization of Laplace's equation and was first introduced in the context of potential theory. Poisson's work in the early 19th century focused on electrostatics and gravitational fields, leading to the formulation of this equation.

Poisson's equation in its general form is written as:

$$-\Delta u = -\nabla^2 u = f(r), \tag{3.5}$$

where Δ or ∇^2 is the Laplace Operator and ∇ is the Nabla operator., which in $\mathbb{R}^3$ is given

by:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}, \tag{3.6}$$

u is the scalar **potential filed** and $f(r)$ is a known function representing the **source term**.

The Laplace equation is a special case of the Poisson equation where the source term $f(r)$ is zero:

$$\Delta u = 0. \tag{3.7}$$

There is in general many functions that satisfies each equation, thus to obtain a unique solution, it is necessary to impose some constrains to the equation, these constrains are called **boundary conditions**, which specifies the solution at the boundary of the domain, when these are imposed, the problem is then called a **boundary value problem**.
There are three types of boundary that occur frequently, these are: **Dirichlet conditions, Neumann conditions and Robin conditions.**

Dirichelt Boundary conditions: The Dirichlet boundary condition is a type of boundary condition names after Peter Gustav Lejeune Dirichlet (1805-1859) [17]. This condition specifies the value that the unknown function needs to take along the boundary of the domain. Given, for example, the Laplace equation, the boundary value problem with the Dirichlet boundary condition is written as:

$$\begin{aligned} \Delta u(x) &= 0 \qquad \forall x \in \Omega, \\ u(x) &= g(x) \quad \forall x \in \partial\Omega. \end{aligned} \tag{3.8}$$

Neumann Boundary Condition: The Neumann boundary condition is a type of boundary condition named after Carl Neumann (1832 - 1925). When imposed to a PDE, it specifies the values that the derivative of a solution is going to take on the boundary of the domain. Given for example the Laplace equation, the boundary value problem with the Neumann boundary condition is written as:

$$\begin{aligned} \Delta u(x) &= 0, \qquad \forall x \in \Omega, \\ \frac{\partial u(x)}{\partial n} &= h(x). \quad \forall x \in \partial\Omega. \end{aligned} \tag{3.9}$$

where n represents the unit normal vector to the boundary surface Ω, if $\Omega \subset \mathbb{R}^3$. In the case of an ordinary differential equation, it is $\Omega \subset \mathbb{R}^1$, the derivative normal to the boundary coincides with the global derivative φ'.

Robin Boundary Condition: The Robin boundary condition is a type of boundary condition named after Victor Gustave Robin (1855-1897). It consists of a linear combination of the values of the scalar field and its derivatives on the boundary. In the Laplace equation, we might impose Robin boundary conditions and the boundary value problem is written as:

$$\begin{aligned} \Delta u(x) &= 0 \qquad \forall x \in \Omega, \\ au(x) + b\frac{\partial u(x)}{\partial n} &= j(x) \quad \forall x \in \partial\Omega. \end{aligned} \tag{3.10}$$

where a and b are real parameters.

Cauchy Boundary Conditions: The mixed boundary conditions consists of applying different types of boundary conditions in different parts of the domain. The mixed boundary condition differs from the Robin condition because the latter consist of different types of boundary conditions applied to the same region of the boundary, while the mixed condition implies different types of boundary conditions applied to different parts of the boundary.

3.4 The weak Formulation of Elliptic Problems

Our objective is to introduce the concept of weak formulation, which facilitates the derivation of existence and uniqueness results for solutions and is well-suited for numerical approximation.

Consider the following model problem: Let $\Omega \subset \mathbb{R}^n$ be an open bounded domain. Given a real-valued function $f \in L^2(\Omega)$, we seek a function u defined in Ω that satisfies

$$P \quad \begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega. \end{cases} \tag{3.11}$$

Assuming the solution is sufficiently smooth, particularly the solution u belongs to $H^2(\Omega)$, we can multiply both sides of equation (3.11) by a test function $v \in H_0^1(\Omega)$ and integrate over Ω to obtain:

$$-\int_\Omega (\Delta u)v \ dx = \int_\Omega fv \ dx. \tag{3.12}$$

Applying Green's Identity, we derive that

$$-\int_\Omega v(\Delta u)\, dx = \int_\Omega \nabla u \cdot \nabla v \, dx - \int_{\partial\Omega} v(\nabla u) \cdot \hat{n} \, dx, \tag{3.13}$$

but, since $v \in H_0^1(\Omega)$, the second part of the right hand side vanishes, because $v|_{\partial\Omega} = 0$, therefore:

$$\boxed{\int_\Omega \nabla u \cdot \nabla v \ dx = \int_\Omega fv \ dx \quad \forall v \in H_0^1(\Omega).} \tag{3.14}$$

The boundary condition and the definition of the trace operator lead us to conclude that the solution u of problem (3.14) must satisfy:

$$u \in H_0^1(\Omega). \tag{3.15}$$

3.4.1 Abstract Variational Problem

The previous example fits into a broader abstract framework that effectively addresses a wide range of boundary value elliptic problems. To this end, we consider the following components:

i) A Hilbert space V, equipped with a norm $\|\cdot\|_V$,

ii) A bilinear form $a(\cdot,\cdot)$ continuous and coercive defined on $V \times V$, meaning there exists a constants $M > 0$ and $\alpha > 0$ such that:

$$\begin{aligned} \forall u,\, v \in V, \quad & a(u,v) \leq M\|u\|_V\|v\|_V, \\ \forall v \in V, \quad & a(v,v) \geq \alpha\|v\|_V^2, \end{aligned}$$

iii) A linear functional l that is continuous on V, i.e., an element of the topological dual space $V^* = \mathcal{L}(V, \mathbb{R})$, endowed with the dual norm.

We then can rephrase the problem P as the general variational problem:

$$\mathcal{P}: \quad \begin{cases} \text{Given } l \in V^*, \text{ find } u \in V \text{ solving the problem,} \\ \forall v \in V, \quad a(u,v) = l(v). \end{cases} \tag{3.16}$$

For our specific situation:

$$a(u,v) = l(v) \quad \forall v \in H_0^1(\Omega), \tag{3.17}$$

where $a : H_0^1(\Omega) \times H_0^1(\Omega) \to \mathbb{R}$ is the bilinear form given by

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \; dx, \qquad u, v \in H_0^1(\Omega), \tag{3.18}$$

and $l : H_0^1(\Omega) \to \mathbb{R}$ is the linear functional defined by

$$l(v) = \int_\Omega f v \; dx \qquad v \in H_0^1(\Omega). \tag{3.19}$$

We observe that $l(\cdot)$ is a continuous linear functional, as evidenced by the Poincaré inequality:

$$l(v) = (f,v)_{L^2} \leq \|f\|_{L^2}\|v\|_{L^2} \leq C_{\Omega,p}\|f\|_{L^2}\|\nabla v\|_{L^2} = C\|v\|_{H_0^1(\Omega)}, \tag{3.20}$$

where $C = C_{\Omega,p}\|f\|_{L^2}$. Thus, l is continuous and an element of the dual space of $H_0^1(\Omega)$.

The continuity of $a(\cdot,\cdot)$ follows from the Cauchy-Schwarz inequality:

$$a(u,v) = (\nabla u, \nabla v)_{L^2} \leq \|\nabla u\|_{L^2}\|\nabla v\|_{L^2} = \|u\|_{H_0^1}\|v\|_{H_0^1}. \tag{3.21}$$

Moreover, $a(\cdot,\cdot)$ is coercive by means of the Poincaré inequality:

$$a(u,u) = (\nabla u, \nabla u)_{L^2} = \|\nabla u\|_{L^2}^2 \geq m\|u\|_{H_0^1}, \tag{3.22}$$

with $m = 1$.

Thus, we satisfy the conditions of the Lax-Milgram theorem, ensuring that the variational problem $a(u,v) = l(v)$ has a unique solution $u \in H_0^1(\Omega)$ for all $v \in H_0^1(\Omega)$.

This states the well-posedness of our problem by definition:

Definition 32. The problem $\mathcal{P}$ is said to be **well-posed** if there exists a unique solution and the following linear stability property holds:

$$\exists C > 0, \forall l \in V^*, \quad \|u\|_V \leq C\|l\|_{V^*} \tag{3.23}$$

3.4.2 Residual Operator and the Inf-Sup Condition

Many partial differential equations (PDEs) can be expressed in a **weak form** via an operator

$$\mathcal{R} : U \longrightarrow V^*, \tag{3.24}$$

where U is a space of trial functions, and V^* is the dual space of a test-function space V. Formally, the PDE is stated by requiring that the residual of any admissible candidate solution vanish when paired with every test function $v \in V$:

$$(\mathcal{R}(u),\, v)_{V^* \times V} = 0 \quad \forall\, v \in V. \tag{3.25}$$

For simplicity, let us assume $\mathcal{R}$ represents a linear, inhomogeneous PDE, so we can write

$$(\mathcal{R}(u),\, v)_{V^* \times V} \;=\; a(u, v) \;-\; l(v), \tag{3.26}$$

where $l \in L^2(\Omega)$ and $a : U \times V \to \mathbb{R}$ is a bilinear form. Under this viewpoint, the weak statement $\|\mathcal{R}(u)\| = 0$ for a suitable norm on V^* is equivalent to the existence of an exact solution $u^* \in U$ of the PDE in weak form (3.25).

A common choice is to use the **dual norm** on V^*, induced by the norm on V:

$$\|f\|_{V^*} \;=\; \sup_{v \in V \setminus \{0\}} \frac{|f(v)|}{\|v\|_V}. \tag{3.27}$$

This norm is particularly advantageous because, under suitable assumptions, there is a close relationship between $\|\mathcal{R}(u)\|_{V^*}$ and the error $\|u - u^*\|_U$.

We assume that $a(\cdot, \cdot)$ is continuous, i.e.,

$$|a(u, v)| \;\leq\; M\, \|u\|_U\, \|v\|_V, \tag{3.28}$$

for some constant $M > 0$, and also satisfies an **inf-sup stability condition**

$$\inf_{u \in U \setminus \{0\}} \sup_{v \in V \setminus \{0\}} \frac{|a(u, v)|}{\|u\|_U\, \|v\|_V} \;\geq\; \gamma \quad (0 < \gamma \leq M). \tag{3.29}$$

We define the linear map

$$A : U \longrightarrow V^*, \tag{3.30}$$

$$u \longmapsto \big\{\, v \mapsto a(u, v) \big\}. \tag{3.31}$$

The boundedness of a ensures that A is well-defined and continuous; the inf-sup condition guarantees injectivity and, under further assumptions, surjectivity. Consequently, if A is onto V^*, there exists a unique solution u^* for the PDE in (3.25) given any $f \in V^*$. For more details on the bijectivity argument, see the Babuška–Brezzi inf-sup theorem. We present a more general version of this theorem:

Theorem 15 (Babuška–Brezzi for PDEs)**.** Let V and M be two Hilbert spaces associated with a PDE problem, where

$$V \;=\; \{\,\text{trial (or solution) functions}\,\}, \quad M \;=\; \{\,\text{functions enforcing constraints or auxiliary variables}\,\}.$$

Suppose we have two continuous bilinear forms

$$a(\cdot,\cdot) : V \times V \;\longrightarrow\; \mathbb{R} \quad \text{and} \quad b(\cdot,\cdot) : V \times M \;\longrightarrow\; \mathbb{R},$$

and two continuous linear functionals

$$\ell : V \;\longrightarrow\; \mathbb{R}, \quad \chi : M \;\longrightarrow\; \mathbb{R}.$$

We consider the following variational problem:

$$\begin{cases} a(u,v) \;+\; b(v,\lambda) \;=\; \ell(v), & \forall\, v \in V, \\ b(u,\mu) \;=\; \chi(\mu), & \forall\, \mu \in M. \end{cases} \tag{3.32}$$

Suppose the following conditions hold:

1. **(Coercivity on a subspace):** There exists a subspace

$$U_0 \;=\; \{\, v \in V : b(v,\mu) = 0 \;\; \forall\, \mu \in M\},$$

such that $a(\cdot,\cdot)$ is coercive on U_0, i.e., there is a constant $\alpha > 0$ with

$$a(v,v) \;\geq\; \alpha\, \|v\|_V^2, \quad \forall\, v \in U_0.$$

2. **(Inf-sup condition):** The bilinear form $b(\cdot,\cdot)$ satisfies

$$\inf_{u \in V\setminus\{0\}} \sup_{\mu \in M\setminus\{0\}} \frac{|b(u,\mu)|}{\|u\|_V\, \|\mu\|_M} \;\geq\; \gamma, \quad \gamma > 0.$$

3. **(Continuity):** The bilinear forms $a(\cdot,\cdot)$ and $b(\cdot,\cdot)$, along with the linear functionals ℓ and χ, are continuous with respect to the norms on V and M.

Then the variational problem (3.32) admits a **unique solution** $(u,\lambda) \in V \times M$. Moreover, the solution map

$$(\,\ell, \chi\,) \;\longmapsto\; (u,\lambda)$$

is a continuous linear operator from $V^* \times M^*$ onto $V \times M$. In particular,

- **Existence:** A solution (u,λ) exists for every right-hand side (ℓ,χ).
- **Uniqueness:** The solution is unique under the above constraints.

- **Stability:** Small changes in ℓ or χ lead to small changes in the solution (u, λ).

When $V = U$, it is, the test and trial spaces of the variational formulation coincide, and the bilinear form $a(\cdot, \cdot)$ is coercive on V, the inf-sup condition is automatically satisfied. Specifically, if

$$a(v, v) \;\geq\; \alpha \, \|v\|_V^2, \quad \alpha > 0,$$

then for $u \neq 0$, choosing $v = u$ shows

$$\sup_{v \neq 0} \frac{a(u, v)}{\|u\|_V \, \|v\|_V} \;\geq\; \frac{a(u, u)}{\|u\|_V^2} \;\geq\; \alpha,$$

so the inf-sup constant can be taken as $\gamma = \alpha$, the coercivity constant.

3.5 Error estimate using the dual norm of the residual

Suppose $u^* \in U$ is the exact solution of (3.25), i.e., $\mathcal{R}(u^*) = 0$ in V^*. Then

$$a(u^*, v) = l(v),$$

For any approximate solution $u \in U$, the weak residual becomes

$$(\mathcal{R}(u), v)_{V^* \times V} \;=\; a(u, v) \;-\; a(u^*, v) \;=\; a(u - u^*, \, v),$$

for all $v \in V$. Then,

$$\|\mathcal{R}(u)\|_{V^*} \;=\; \sup_{v \neq 0} \frac{\big|a(u - u^*, \, v)\big|}{\|v\|_V} \;\leq\; M \, \|u - u^*\|_U, \tag{3.33}$$

using (3.28). On the other hand, from the inf-sup condition (3.29),

$$\|\mathcal{R}(u)\|_{V^*} \;=\; \sup_{v \neq 0} \frac{\big|a(u - u^*, \, v)\big|}{\|v\|_V} \;\geq\; \gamma \, \|u - u^*\|_U.$$

Hence,

$$\frac{1}{M} \, \|\mathcal{R}(u)\|_{V^*} \;\leq\; \|u - u^*\|_U \;\leq\; \frac{1}{\gamma} \, \|\mathcal{R}(u)\|_{V^*}. \tag{3.34}$$

This shows that the error in the solution is equivalent, up to constants, to the dual norm of the residual. In practical terms, minimizing $\|\mathcal{R}(u)\|_{V^*}$ is a valid strategy for approximating the true solution in the U-norm.

In this context, the residual of the PDE serves as a measure of how well a function u satisfies the governing equation. In general, the PDE operator can be described by a map $\mathcal{R} : U \to V$, where U and V are Banach spaces, usually $U = H_0^1(\Omega)$ or $L^2(\Omega)$. The PDE then takes the form:

$$\mathcal{R}(u) = 0. \tag{3.35}$$

For instance, consider Poisson's equation (3.11), $-\Delta u = f$ on a domain Ω with homogeneous Dirichlet boundary conditions, where $f \in L^2(\Omega)$. This problem can be formulated in two ways: the strong formulation and the weak formulation.

In the **strong formulation**, the problem is described by the operator:

$$\begin{aligned} \mathcal{R}^s : H^2(\Omega) \cap H_0^1(\Omega) &\longrightarrow L^2(\Omega) \\ u &\longmapsto \Delta u - f. \end{aligned} \tag{3.36}$$

However, when the function u lacks sufficient regularity, it is more convenient to use the **weak formulation**

$$\begin{aligned} \mathcal{R}^w : H_0^1(\Omega) &\longrightarrow H^{-1}(\Omega) \\ (\mathcal{R}^w(u), v)_{H^{-1}\times H^1} &= \int_\Omega \nabla u \cdot \nabla v - f v \, dx. \end{aligned} \tag{3.37}$$

3.5.1 The Weak Residual for the Poisson Equation

Consider the Poisson equation (3.11),

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases}$$

where $\Omega \subset \mathbb{R}^n$ is a bounded Lipschitz domain. Its variational formulation asks: find $u \in H_0^1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v \, dx \;=\; \int_\Omega f \, v \, dx, \quad \forall\, v \in H_0^1(\Omega).$$

In the broader inf-sup framework, one typically has two Hilbert spaces V (trial/solution) and M (auxiliary/constraint) with a bilinear form $b(\cdot,\cdot) : V \times M \to \mathbb{R}$ enforcing constraints. The Poisson setting, however, has no extra constraints that require a second space. The single bilinear form

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v \, dx,$$

is both continuous and *coercive* on $V = H_0^1(\Omega)$. Hence, one directly applies the Lax–Milgram theorem (7.1) to conclude existence and uniqueness of the solution.

- Since M is trivial, the inf-sup condition of Babuška–Brezzi for $b(\cdot,\cdot)$ does not come into play.
- The positivity (coercivity) of $a(\cdot,\cdot)$ on $H_0^1(\Omega)$ is enough to guarantee a unique weak solution.

- As a result, the specialized theorem reduces to the *classical* Lax-Milgram Theorem approach, ensuring that a single Hilbert space and a single bilinear form suffice for well-posedness.

The **Weak Residual of Elliptic Problem** is a function $\mathcal{R}^w$ which maps a candidate solution $u \in H_0^1(\Omega)$ to a element $\mathcal{R}^w(u,\cdot)$ of the dual space $H^{-1}(\Omega)$ such that:

$$\begin{aligned}
&\mathcal{R}^w : H_0^1(\Omega) \longrightarrow H^{-1} \\
&\qquad\quad u \mapsto \mathcal{R}^w(u,) \qquad \text{such that,} \\
&\mathcal{R}^w(u,v) = (\mathcal{R}^w(u), v)_{H^{-1}\times H_0^1(\Omega)} = \int_\Omega \nabla u \cdot \nabla v \, dx - \int_\Omega f v \, dx, \qquad \forall v \in H_0^1(\Omega),
\end{aligned}$$

which can be stated as an abstract variational problem as we did in 3.4.1.

Let $a : H_0^1(\Omega) \times H_0^1(\Omega) \to \mathbb{R}$ be the bilinear form defined by

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \, dx, \quad u, v \in H_0^1(\Omega), \tag{3.38}$$

and let $l : H_0^1(\Omega) \to \mathbb{R}$ be the linear functional defined by

$$l(v) = \int_\Omega f v \, dx, \quad v \in H_0^1(\Omega). \tag{3.39}$$

This formulation satisfies the inf-sup stability condition (see 3.29), since the bilinear form $a(\cdot,\cdot)$ is coercive and continuous on $H_0^1(\Omega)$.

Then, we can rewrite the weak residual as:

$$\mathcal{R}^w(u,v) = a(u,v) - l(v). \tag{3.40}$$

With this residual we can estimate the error of the approximation without know beforehand the exact solution of the PDE. We can state the variational formulation as a minimization problem:

$$\text{Find } u \in H_0^1(\Omega) \text{ such that } \mathcal{R}^w(u,v) = 0 \ \forall v \in H_0^1(\Omega). \tag{3.41}$$

From 3.34 we get that given an candidate solution $u \in H_0^1(\Omega)$ to the variational problem 3.17, we can quantify the discrepancy between the exact solution and the candidate by calculating the dual norm of the residual, via

$$\frac{1}{M} \|\mathcal{R}^w(u,\cdot)\|_{H^{-1}} \ \leq \ \|u - u^*\|_{H_0^1} \ \leq \ \frac{1}{\gamma} \|\mathcal{R}^w(u,\cdot)\|_{H^{-1}}.$$

Which means that one can have a estimation of the error only by knowing the nom of the residual operator. The challenge remain however on how to calculate such dual norm, which

in general is given by

$$\|R\|_{H^{-1}} = \sup_{0\neq v\in H_0^1(\Omega)} \frac{(\mathcal{R}(u), v)_{H^{-1}(\Omega)\times H_0^1(\Omega)}}{\|v\|_{H_0^1}} = \sup_{0\neq v\in H_0^1(\Omega)} \frac{\mathcal{R}^w(u,v)}{\|v\|_{H_0^1}}.$$

To render this norm computable, we proceed as follows. Let $\{\varphi_k\}_{k=1}^{\infty}$ be an orthonormal basis for $H_0^1(\Omega)$ with respect to the inner product

$$(u,v)_{H_0^1} = \int_\Omega \nabla u \cdot \nabla v \, dx.$$

By the Riesz representation theorem (5), there exists a unique $\psi \in H_0^1(\Omega)$ such that

$$(\psi, v)_{H_0^1} = \mathcal{R}^w(u,v) \quad \forall\, v \in H_0^1(\Omega).$$

Expanding ψ in the basis,

$$\psi = \sum_{k=1}^{\infty} c_k \, \varphi_k, \quad \text{with } c_k = (\psi, \varphi_k)_{H_0^1} = \mathcal{R}^w(u, \varphi_k),$$

Parseval's identity then implies

$$\|\psi\|_{H_0^1}^2 = \sum_{k=1}^{\infty} |c_k|^2 = \sum_{k=1}^{\infty} |\mathcal{R}^w(u,\varphi_k)|^2 .$$

Since the norm $\|\psi\|_{H_0^1}$ is equivalent to the H^{-1} norm of the residual, we have

$$\|\mathcal{R}^w(u,\cdot)\|_{H^{-1}} = \|\psi\|_{H_0^1} = \left(\sum_{k=1}^{\infty} |\mathcal{R}^w(u,\varphi_k)|^2 \right)^{\frac{1}{2}}.$$

In practice, this infinite sum is approximated by truncating the series after N terms:

$$\|\mathcal{R}^w(u,\cdot)\|_{H^{-1}} \approx \left(\sum_{k=1}^{N} |\mathcal{R}^w(u,\varphi_k)|^2 \right)^{\frac{1}{2}}.$$

This discretization provides a computable error indicator that can be used, for instance, as a loss function (A.1.2) in numerical schemes or in the training of physics-informed neural networks.

Chapter 4

Physics-Informed Neural Networks

The numerical solution of PDEs is widely used for the solution of physical systems, from heat diffusion to fluid dynamics. Traditional approaches, such as the finite element method (FEM) (A.2.2), have long dominated this field. In FEM, one discretize the domain into a mesh, approximates solutions using piecewise polynomial basis functions, and enforces physical laws through variational formulations.

In recent years, neural networks (for reference, you can read Section A.1), accompanied by the **universal approximation theorem** [21] offer a flexible framework for representing solutions to PDEs. However, purely data-driven approaches struggle in physical systems where data is sparse, noisy, or confined to limited regions of the domain. **Physics-informed neural networks** (PINNs) integrate the physical laws of these systems, encoded as PDEs, into the neural network's training process, making easier to integrate data driven learning with a more fundamental modeling. Like FEM, PINNs aim to approximate solutions of PDEs while taking into account governing physical laws, but they differ in their approach.

Finite Element Methods rely on predefined polynomial basis functions over a discretized mesh. They enforce PDEs in their weak form, which reduces the order of derivatives and enables piecewise approximations. FEM depends purely on the PDE and boundary conditions and typically requires re-meshing to refine solutions locally. In contrast, Physics-Informed Neural Networks (PINNs) use neural networks to implicitly learn adaptive basis functions. This approach bypasses mesh generation and facilitates the handling of irregular domains. Moreover, PINNs operate directly on the strong form of the PDE and leverage automatic differentiation [5] to compute exact derivatives of the network output. They can also incorporate sparse experimental data directly into the loss function (A.1.2) and achieve adaptability through gradient-based optimization, dynamically prioritizing regions where the PDE residual or data mismatch is large.

4.1 Physics-Informed Neural Networks

As discussed in the appendix section, training neural networks often relies on large datasets to fine-tune the model parameters effectively. However, in the context of physical systems, obtaining sufficient data can be particularly challenging. Data may be incomplete, biased, or noisy due to limitations in measurement techniques, making it difficult for traditional machine learning approaches to converge to meaningful solutions.

Using neural networks to find the solution of PDEs can be traces back to the works of Dissanayake and Phan-Thien in 1994 and latter to the aportations of Lagaris, Likas and Fotiadis in 1998. Nevertheless they lacked a framework in which develop a simple implementation using home computers. To address this, Maziar Raissi, Paris Perdikaris, and George Karniadakis introduced a framework in 2017 that integrates prior knowledge of the physical laws governing system dynamics directly into the neural network architecture [24]. This approach leverages the fundamental principles underlying the system—such as conservation laws and symmetries—as a form of regularization. By embedding these physical constraints, the neural network is guided toward a more realistic solution space, filtering out predictions that violate the inherent physics of the problem [22, 23, 35, 40]

The method, termed *Physics-Informed Neural Networks* (PINNs), combines data-driven learning with the governing equations of the system. PINNs are trained not only on data but also on the mathematical laws that describe the system's behavior, such as partial differential equations (PDEs). This integration enables the model to generalize well, even in scenarios where data is scarce or noisy. A key enabler of this approach is **automatic differentiation**, which allows for the efficient computation of derivatives with respect to both the inputs and parameters of the network—crucial for ensuring that the network satisfies the physical constraints throughout the learning process.

In this section, we focus on using PINNs for the solution of partial differential equations (PDEs). For illustrative purposes, we consider the Poisson equation (3.11) in one dimension:

$$\begin{aligned} -u_{xx} &= f \quad \text{in } (0,1), \\ u &= 0 \quad \text{on } \{0,1\}, \end{aligned} \tag{4.1}$$

and develop a physics-informed neural network to approximate the solution as the output of a deep neural network.

The Neural Network Approach

In this approach, we use fully connected feedforward neural networks, so the following discussion focuses on this class of networks (A.1.2). The goal is to approximate a function with a scalar value, and we will denote the neural network by $u_{NN} : \mathbb{R}^n \to \mathbb{R}$, which is defined as follows:

For any input vector $\vec{x} \in \mathbb{R}^n$, the output $u_{NN}(\vec{x})$ is computed through a sequence of

operations:

$$\begin{aligned} h_1 &= \sigma(W_1\vec{x} + b_1), \\ h_{l+1} &= \sigma(W_{l+1}h_l + b_{l+1}), \quad l = 1, \dots, k, \\ u_{NN}(\vec{x}) &= W_{k+1}h_k + b_{k+1}. \end{aligned} \tag{4.2}$$

Here, $k+1$ represents the depth of the network (i.e., the number of layers). The matrices $W_l \in \mathbb{R}^{d_{l+1}\times d_l}$where d_l denotes the number of neurons in layer l, and vectors $b_l \in \mathbb{R}^{d_{l+1}}$ store the weights and biases of the network. The function σ is a non-linear activation function, applied component-wise, meaning for any vector $\vec{y} \in \mathbb{R}^n$, we have $\sigma(\vec{y}) = [\sigma(y_1), \sigma(y_2), \dots, \sigma(y_n)]$.

The regularity of u_{NN} depends on the choice of the activation function σ on the last layers. For example, using ReLU (Rectified Linear Unit) as the activation function results in $u_{NN} \in C^0(\mathbb{R})$, while using RePU (Rectified Power Unit) allows that $u_{NN} \in C^r(\Omega)$ for some finite r. In our case, we use the hyperbolic tangent (tanh) as the activation function, which makes $u_{NN} \in C^\infty(\Omega)$, fully exploiting the regularity of the solution.

All weights and biases of the network can be collected in a single vector of parameters $\theta \in \mathbb{R}^P$, where P represents the total number of parameters. This structure induces a mapping

$$\begin{aligned} \mathcal{F}^{NN} : \mathbb{R}^P &\to C^\infty(\Omega) \\ \theta &\mapsto \mathcal{F}^{NN}(\theta) = u_{NN} \\ u_{NN} &= u_{NN}(\vec{x}, \theta). \end{aligned}$$

It is useful to define the set

$$U_{NN} \subset U, \quad U_{NN} = \mathcal{F}^{NN}(\mathbb{R}^P), \tag{4.3}$$

where U contains all the functions that can be generated by the neural network with the given architecture.

To speak about *physics-informed* neural networks, one should say something about the loss function (A.1.2) one should take in order to train the neural network.

Consider for example,

$$\|\Delta u - f\|_{L^2} := \|\mathcal{R}^s(u)\|_{L^2} := \int_\Omega |\mathcal{R}^s(u)|^2 \, dx,$$

that is, the strong residual of the strong formulation of the Poisson equation with Dirichlet boundary condition (3.11) and in such way, to measure the discrepancy between any candidate solution $u^* \in H^2(\Omega)$ and the exact solution $u \in H^2(\Omega)$ since, if the problem is well posed, this solution is unique and is the minimum of $\|\mathcal{R}^s(u)\|_{L^2}$.

Thus, solving the minimization problem for PINNs becomes finding a suitable $\mathcal{L}(u)$, a loss function (A.1.2) from which one can solve:

$$\text{Find } \theta \in \mathbb{R}^P \text{ such that } \theta = \arg\min_{\theta \in U_{NN}} \mathcal{L}(u_{NN}(\vec{x}, \theta)). \tag{4.4}$$

In this formulation, the neural network parameters are optimized to minimize the residual of the PDE, ensuring that the network approximates the true solution of the physical system.

The Construction of the Loss Function

Physics-informed neural networks (PINNs) operate on the assumption that the unknown function u, which solves a given partial differential equation (PDE), can be approximated by a neural network, via the **universal approximation theorem** [21]. The partial derivatives required in the PDE are computed via **automatic differentiation** [5] on the neural network. Formally, a PDE can be expressed as:

$$F(D^k u(x), D^{k-1}u(x), \ldots, Du(x), u(x), x) = 0, \quad x \in \Omega, \tag{4.5}$$

where D^i represents the multi-index notation for partial derivatives, and $x = (x, t)$ captures both spatial and time-like variables.

In the context of PINNs, automatic differentiation [5] provides an efficient way to compute these derivatives. To frame the problem, let $\bar{D}$ denote the partial derivatives computed through automatic differentiation on the neural network. We aim to approximate the solution of the PDE by minimizing the difference between the PDE's true residual and the network's approximation:

$$\max_{x\in\Omega} \left| F(D^k u(x), \ldots, Du(x), u(x), x) - F(\bar{D}^k u_{NN}(x), \ldots, \bar{D}u_{NN}(x), u_{NN}(x), x) \right| < \varepsilon. \tag{4.6}$$

For this to hold, the unknown function $u(x)$ must be well approximated by the neural network $u_{NN}(x)$ and its derivatives, yielding:

$$\sup_{x\in\Omega} |u(x) - u_{NN}(x)| < \varepsilon. \tag{4.7}$$

By the Universal Approximation Theorem [21], a neural network of sufficient size and depth can satisfy this condition. Furthermore, for each partial derivative, we require:

$$\sup_{x\in\Omega} |D^j u(x) - \bar{D}^j u_{nn}(x)| < \varepsilon_j, \tag{4.8}$$

for some $\varepsilon_j > 0$ and $j = 1, \ldots, k$, assuming the neural network is continuously differentiable for the required derivatives of the PDE. This explains the importance of using differentiable activation functions.

4.2 PINNs

For well-posed PDEs with sufficient regularity, $\mathcal{R}^s(u) \in L^2(\Omega)$. Therefore, the residual is minimized using the L^2-norm, given by:

$$\|\mathcal{R}^s(u)\|_{L^2}^2 = \int_\Omega \mathcal{R}^s(u)^2 \, d\Omega. \tag{4.9}$$

The loss function for the PDE is defined by

$$\mathcal{L}(u_{NN}) := \|\mathcal{R}^s(u_{NN})\|_{L^2}^2$$

That is, the objective in PINNs is to approximate the solution by minimizing the **strong residual**. Specifically, we seek to find u such that:

$$u = \underset{u \in L^2(\Omega)}{\arg\min} \|\mathcal{R}^s(u)\|_{L^2}^2 \tag{4.10}$$

To compute this norm we use a discretization of the domain Ω, which leads to an approximation of the norm. Different discretization methods affect the convergence rate and the minimization of errors.

A classical approach is to adopt an random sampling of the domain and utilize Montecarlo integration to approximate $\|\mathcal{R}^s(u)\|_{L^2}$. In the PINNs approach, this sample is incorporated into approximating the L^2-norm of the residual as follows:

$$\mathcal{L}_{\mathcal{R}^s}(u_{NN}) := \|\mathcal{R}^s(u_{NN})\|_{L^2}^2 \approx \frac{1}{N_S} \sum_{i=1}^{N_S} |\mathcal{R}^s(u_{NN}(x^i, \theta))|^2 \tag{4.11}$$

where $\{x^i\}_{i=1}^{N_S} \subset \Omega$ are randomly sampled points.

In addition, the PDE must satisfy boundary condition that must be enforced during the training process. Let d^i represent the N_B points that describe the Dirichlet boundary and initial conditions of the problem. The loss for these points is calculated via the mean squared error with respect of the boundary points x_B^j:

$$\mathcal{L}_\Gamma(u_{NN}) = \frac{1}{N_B} \sum_{j=1}^{N_B} |u(x_B^j, \theta) - d^j|^2. \tag{4.12}$$

By combining equations 4.11 and 4.12, we obtain the total loss function for the PDE:

$$\mathcal{L}(u_{NN})_{\text{total}} = \mathcal{L}(u_{NN})_{\mathcal{R}^s} + \mathcal{L}(u_{NN})_\Gamma. \tag{4.13}$$

This approach allows the neural network to adjust its parameters based on the physics of the system, guiding it toward solutions that obey the underlying physical laws.

4.3 Numerical experiment

In this section, we present a numerical experiment designed to illustrate both the strengths and some of the limitations of the physics-informed neural network (PINN) approach for solving partial differential equations.

In this toy problem, we consider the Poisson equation (3.11), and take $\Omega = (0, \pi)$, the source term $f(x) = -4\sin(2x)$ and zero Dirichlet boundary conditions. This leads to the exact solution $u_{\text{exact}}(x) = \sin(2x)$.

For the implementation, we use *TensorFlow 12.8* to build a feed-forward, fully-connected neural network. The architecture consists of five hidden layers, each containing 10 neurons, with the hyperbolic tangent (tanh) serving as the activation function throughout. The network parameters are optimized using the Adam optimizer with a learning rate of 10^{-3}. Moreover, we use Monte Carlo integration [30] with 500 collocation points and 10000 epochs of training.

In Figure 4.1, we compare the exact solution with the neural network approximation u_{NN}. Although the PINN approximation is highly accurate in the interior of the domain, the boundary conditions are not satisfied exactly. This slight discrepancy arises because the boundary conditions are enforced via the optimizer rather than being imposed strongly within the PDE formulation.

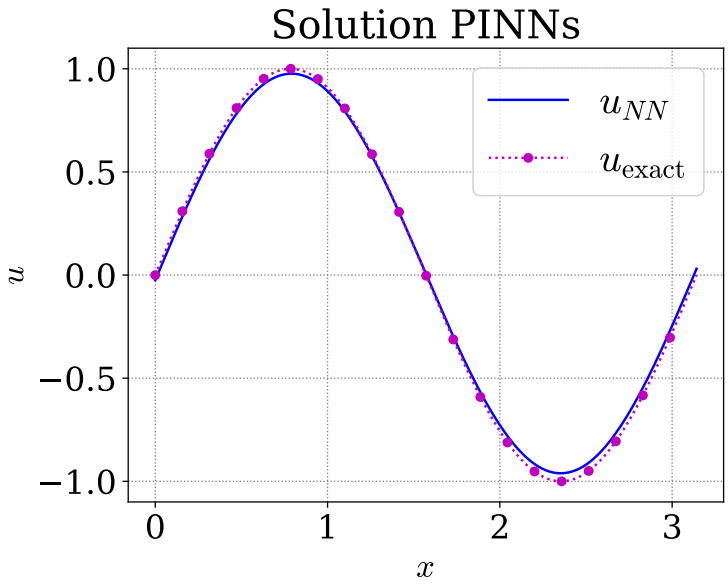


Figure 4.1: Approximation of the neural network (u_{NN}) to the exact solution (u_{exact})

In Figure 4.2 we present the evolution of the loss on a log-log plot. Although the loss decreases by approximately three orders of magnitude over 10 000 iterations of the Adam solver, this reduction does not guarantee that the approximation error decreases at a comparable rate. In fact, after the first 50 iterations, oscillations become apparent in the loss curve; these oscillations can be attributed to the inherent variability of Monte Carlo integration. This behavior is well documented in [24] (see also [36]). In light of these observations, we proceed to discuss more robust techniques for both error estimation and the control of numerical integration errors.

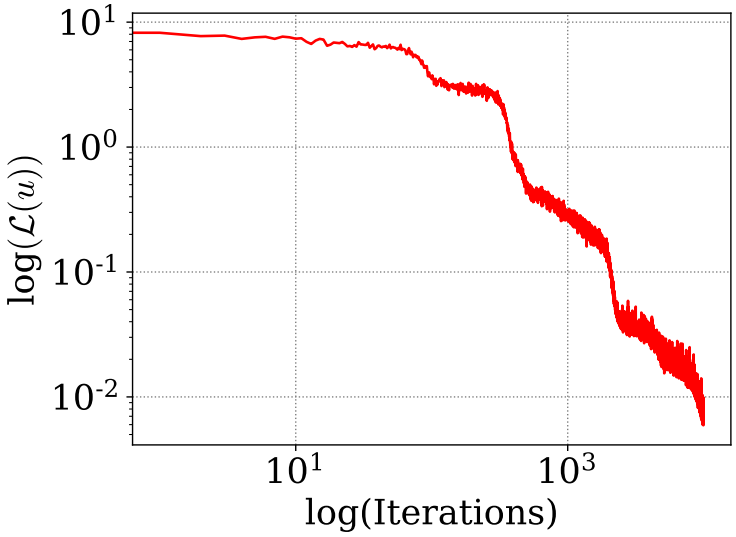


Figure 4.2: Evolution of the loss function (4.13) after 10,000 epochs

Chapter 5

Robust Variational Physics-Informed Neural Networks (RVPINNS)

5.1 Variational Physics-Informed Neural Networks

Physics-informed neural networks (PINNs) have been applied to a wide range of scientific computing problems. However, the loss function in standard PINNs (4.13) is typically derived from the strong form of the PDE residual. For certain problems with lower regularity in the data, a variational form solution may be more suitable.

A natural extension to PINNs is the **Variational PINNs** (VPINNs) framework [27, 26, 28, 29]., which introduces a variational formulation of the underlying PDE using a Petrov-Galerkin framework. In VPINNs, the solution is approximated by a deep neural network, while the test functions are chosen from linear vector spaces. This approach is particularly sensitive to the choice of the test function basis. Given a discrete test space, the stability, robustness, and precision of the loss function of the method are highly dependent on the selection of these test functions [4, 9, 10]

One limitation of VPINNs is that the loss function is not always robust with respect to the true error [37]; it can tend to zero even when the true error does not. It is also essential to select appropriate quadrature rules and the degree of piecewise polynomial test functions to ensure optimal convergence rates during mesh refinements.

To address these challenges, one can draw upon ideas from minimal residual methods (MinRes), a class of numerical methods designed to solve PDEs with guaranteed stability. The essence of MinRes methods is to minimize the dual norm of the PDE residual in its variational form. Since the residual operator resides in the dual of the test space and its dual norm is generally difficult to compute, the Riesz representation theorem becomes useful. Instead of minimizing the residual in the dual norm, we minimize its Riesz representative, which lies in the test space and can be measured in a suitable test norm.

In [37], the authors review the initial works on VPINNs from [27] and propose a more ro-

bust version of variational PINNs. As in VPINNs, they employ a Petrov-Galerkin formulation of the PDE, approximate the solution using neural networks, and select a finite-dimensional test space. The objective is then to minimize the residual in the discrete dual norm. This is achieved by selecting a single test function that represents the weak residual's Riesz representative over the test space. By expanding this Riesz representative on a discrete basis, the resulting loss function incorporates the inverse of the Gram matrix for the selected inner product.

The norm of this test function is both efficient and reliable, as it bounds the true error from below and above. However, the method remains sensitive to the choice of the discrete basis spanning the test space, provided that the numerical integration and the Gram matrix inversion are accurate. Minimizing this proposed loss is equivalent to minimizing the test norm of the Riesz representative of the residual, reflecting the core concept of classic MinRes methods.

The strategy relies in two main ideas. First that the appropriate selection of the inner product in the test space, ensuring the stability of the variational formulation and second, the selection of a single test function, that is the Riesz representation of the weak residual [37].

In particular, if one choose an orthonormal discrete basis with respect to the inner product in the test space, the Gram matrix becomes the identity and we recover the original definition of the loss function in VPINNs. The largest bottleneck of RVPINNs at the moment is that in certain configurations, we need to invert the Gram matrix. However there are cases where the inversion of the Gram matrix is trivial or easy to compute [42].

5.2 Classic Variational Physics-Informed Neural networks

The objective of VPINNs is to approximate the solution by minimizing the **weak residual.** To make a computational approximation using neural networks, we are going to implement a Petrov-Galerkin approach, for this need to discretize the test space $H_0^1(\Omega)$ via an infinite Hilbert basis (see [11]):

$$H_0^1(\Omega) = \text{span}\{\varphi_n\}_{n\geq 1} \tag{5.1}$$

From this basis, we select a finite subset $V_M = \{\varphi_m\}_{n=1}^M$, which will impact the accuracy of the approximation depending on the number and choice of basis elements.

As in the construction of PINNs, let us denote by $u_{NN}(x;\theta)$ the realization of a neural network with trainable parameters $\theta \in \mathbb{R}^P$. According to equation 3.34, we can estimate the error of the approximation using the dual norm of the residual. However, computing such a norm is generally not possible in a computational setting. Therefore, we approximate this norm by using the finite subset V_M of the basis of $H_0^1(\Omega)$ in a Petrov-Galerkin manner. The

problem then consists of finding u_{NN} such that

$$\text{Find } u_{NN} \text{ such that } \mathcal{R}^w(u_{NN}(x;\theta),\varphi_m) = 0 \quad \text{for all } \varphi_m \in V_M. \tag{5.2}$$

Therefore, the minimization problem at hand is

$$\text{Find } \theta \in \mathbb{R}^P \text{ such that } \theta = \arg\min_{\theta\in\mathbb{R}^P} \mathcal{L}_w(u_{NN}(x,\theta)). \tag{5.3}$$

The typical definition of the loss function in this context is

$$\mathcal{L}_{\mathcal{R}^w}(u_{NN}(x;\theta)) = \sum_{m=1}^{M} \left(\mathcal{R}^w(u_{NN}(x;\theta),\varphi_m)\right)^2 + C(u_{NN}(x;\theta)), \tag{5.4}$$

where $C(\cdot)$ is a V_M-independent quadratic functional employed to impose the boundary conditions of the PDE.

However, the classical VPINNs approach can suffer from issues of convergence due to the choice of the basis functions. The method's performance heavily depends on the selection of these test functions, which may lead to instability or slow convergence if not chosen appropriately. For this reason, we adopt the robust variational approach.

5.3 Robust Variational Physics-Informed Neural Networks

To motivate the construction of robust loss functions in variational physics-informed neural networks (VPINNs), we begin by examining how the residual interacts with the test functions in the discrete test space V_M. For a given set of trainable parameters θ, corresponding to a neural network approximation $u_{NN}(x;\theta)$, we can define a function $\bar{\varphi} \in V_M$ as:

$$\bar{\varphi} = \sum_{m=1}^{M} \mathcal{R}^w(u_{NN}(x;\theta),\varphi_m)\,\varphi_m, \tag{5.5}$$

where $\{\varphi_m\}_{m=1}^M$ are the basis functions spanning the finite-dimensional test space V_M, and $\mathcal{R}^w(u_{NN}(x;\theta),\varphi_m)$ denotes the evaluation of the weak residual at u_{NN} with test function φ_m.

Due to the linearity of the weak residual $\mathcal{R}^w$ in its second argument and the properties of the inner product, it follows that:

$$\mathcal{L}_{\mathcal{R}^w}(u_{NN}(x;\theta)) = \sum_{m=1}^{M} \left(\mathcal{R}^w(u_{NN}(x;\theta), \varphi_m)\right)^2 + C(u_{NN}(x;\theta))$$
$$= \mathcal{R}^w(u_{NN}(x;\theta), \bar{\varphi}) + C(u_{NN}(x;\theta)), \tag{5.6}$$

where $C(u_{NN}(x;\theta))$ is a penalty term used to enforce boundary conditions or other constraints. This observation suggests that VPINNs can be generalized by considering the residual evaluated at a particular test function $\bar{\varphi} \in V_M$. Thus, we can define the loss function as:

$$\mathcal{L}_{\mathcal{R}^w}(u_{NN}) = \mathcal{R}^w(u_{NN}, \bar{\varphi}) + C(u_{NN}). \tag{5.7}$$

The key challenge now is to choose the test function $\bar{\varphi}$ such that the VPINN becomes robust. To achieve this, we define $\bar{\varphi}$ as the **Riesz representative** of the residual in V_M with respect to a norm that induces the inf-sup stability condition (see condition 3.29). The strategy is based on leveraging the Riesz representation theorem, which states that for every continuous linear functional on a Hilbert space, there exists a unique element in that space (the Riesz representative) such that the functional can be expressed as an inner product with this element.

In **Robust Variational Physics-Informed Neural Networks** (RVPINNs), they introduce an inner product $(\cdot,\cdot)_V$ on the test space V, whose associated norm $\|\cdot\|_V$ satisfies the inf-sup stability condition. For a given neural network approximation $u_{NN}(x;\theta)$, we compute $\phi := \phi(\theta) \in V_M$ as the solution to the following Galerkin problem:

$$(\phi, \varphi_m)_V = \mathcal{R}^w(u_{NN}(x;\theta), \varphi_m), \quad \text{for } m = 1, \ldots, M. \tag{5.8}$$

Here, ϕ acts as the Riesz representative of the residual $\mathcal{R}^w(u_{NN}(x;\theta), \cdot)$ in the test space V_M. The loss function is then defined as:

$$\mathcal{L}_{\mathcal{R}^w}(u_{NN}(x;\theta)) = \mathcal{R}^w(u_{NN}(x;\theta), \phi) + C(u_{NN}(x;\theta)). \tag{5.9}$$

Due to the linearity of both the weak residual and the inner product in their respective second arguments, we can express ϕ in terms of the basis functions:

$$\phi = \sum_{m=1}^{M} \eta_m(\theta)\varphi_m, \tag{5.10}$$

where $\eta_m(\theta)$ are coefficients to be determined.

Substituting this expression into equation (5.8) leads to a linear system in matrix form:

$$(\phi, \varphi_n)_V = \left(\sum_m \eta_m(\theta)\varphi_m, \varphi_n \right)_V = \sum_m \eta_m(\varphi_m, \varphi_n)_V = \mathcal{R}^w(u_{NN}(x;\theta)) \tag{5.11}$$

which can be stated as a linear system as

$$\mathbf{G}\,\eta(\theta) := R(\theta), \tag{5.12}$$

where: $\eta(\theta) = [\eta_1(\theta), \eta_2(\theta), \ldots, \eta_M(\theta)]^\top$ is the vector of coefficients. $\mathbf{G}$ is the Gram matrix with entries $\mathbf{G}_{nm} = (\varphi_n, \varphi_m)_V$ and $R(\theta)$ is the residual vector with entries $R_n(\theta) = \mathcal{R}^w(u_{NN}(x;\theta), \varphi_n)$. The **Gram Matrix**, is symmetric and positive definite due to the properties of the inner product.

Solving for $\eta(\theta)$, we obtain:

$$\eta(\theta) = \mathbf{G}^{-1} R(\theta). \tag{5.13}$$

Substituting into the loss function, we have:

$$\begin{aligned} \mathcal{R}^w(u_{NN}(x;\theta), \phi) &= \sum_{m=1}^{M} \eta_m(\theta) \mathcal{R}^w(u_{NN}(x;\theta), \varphi_m) \\ &= R(\theta)^\top \eta(\theta) \\ &= R(\theta)^\top \mathbf{G}^{-1} R(\theta). \end{aligned} \tag{5.14}$$

Thus, the loss function can be equivalently written as:

$$\mathcal{L}_{\mathcal{R}^w}(u_{NN}(x;\theta)) = R(\theta)^\top \mathbf{G}^{-1} R(\theta) + C(u_{NN}(x;\theta)). \tag{5.15}$$

We observe that minimizing this loss function is equivalent, up to the constraint $C(u_{NN})$, to minimizing the squared norm of ϕ in V, since:

$$\begin{aligned} \|\phi\|_V^2 &= (\phi, \phi)_V \\ &= \mathcal{R}^w(u_{NN}(x;\theta), \phi) \\ &= R(\theta)^\top \mathbf{G}^{-1} R(\theta). \end{aligned} \tag{5.16}$$

Therefore, minimizing the loss function corresponds to minimizing $\|\phi\|_V^2$, which serves as a robust local error estimator for the approximation error $\|u - u_{NN}\|$.

When the test space V_M is spanned by an orthonormal set $\{\varphi_m\}_{m=1}^M$ with respect to the norm $\|\cdot\|_V$, we have that $|\varphi_m| = 1$ and $(\varphi_m, \varphi_n)_{H_0^1} = 0$, for all $m \neq n$ and the linear space generated by the φ_m's is dense in $H_0^1(\Omega)$; the Gram matrix $\mathbf{G}$ becomes the identity matrix. Consequently, the Riesz representative simplifies to:

$$\phi = \sum_{m=1}^{M} \mathcal{R}^w(u_{NN}(x;\theta), \varphi_m)\varphi_m, \tag{5.17}$$

and the loss function reduces to:

$$\mathcal{L}_{\mathcal{R}^w}(u_{NN}(x;\theta)) = \sum_{m=1}^{M} \left(\mathcal{R}^w(u_{NN}(x;\theta), \varphi_m)\right)^2 + C(u_{NN}(x;\theta)). \tag{5.18}$$

This expression recovers the classical VPINNs loss function discussed in the previous section. However, the RVPINN approach offers a more general and robust framework by incorporating the inner product and Gram matrix, which can handle non-orthonormal bases and different norms.

In this work, we adopt a **Deep Fourier Residual Method** (DFRM), inspired by the approach in [43], where the authors define the loss function as the continuous dual norm of the weak residual. This norm is then approximated by truncating a series expansion using an orthonormal basis for the test space [42]. Specifically, we approximate the norm of the weak residual in the H^{-1} space. This approach turns out to be equivalent to equation (5.18) when considering the standard H^{-1}-norm on V and choosing the basis functions $\{\varphi_m\}$ as orthonormal sinusoidal (spectral) functions with respect to the H^{-1}-norm.

By leveraging the properties of the H^{-1}-norm and spectral basis functions, we can obtain a robust and efficient method for training neural networks to solve PDEs. This is particularly beneficial for problems where classical PINNs or VPINNs may face difficulties due to sensitivity to the choice of test functions or lack of robustness in the loss function.

5.4 Deep Fourier Residual Method

In this section, we delve into the **Deep Fourier Residual Method** (DFRM) [43, 41], which leverages spectral methods and the properties of Fourier series to solve partial differential equations (PDEs) using neural networks. The method can be viewed as a specific instance of the Robust Variational Physics-Informed Neural Networks (RVPINNs) framework discussed earlier.

We consider the Poisson equation (3.11) on the domain $\Omega = (0, \pi)$ with homogeneous Dirichlet boundary conditions:

$$\begin{cases} -u_{xx}(x) = f(x), & x \in \Omega, \\ u(0) = u(\pi) = 0. \end{cases} \tag{5.19}$$

Our goal is to approximate the solution $u(x)$ using a neural network $u_{NN}(x;\theta)$, where θ denotes the trainable parameters. The key idea of the DFRM is to define a loss function based on the continuous dual norm of the weak residual $\mathcal{R}^w(u_{NN})$, which is then approximated using a truncated Fourier series.

The weak formulation of equation (5.19) is given by: find $u \in H_0^1(\Omega)$ such that

$$a(u, v) = l(v), \quad \forall v \in H_0^1(\Omega), \tag{5.20}$$

where

$$a(u,v) = \int_\Omega u_x(x)v_x(x)\,dx,$$
$$l(v) = \int_\Omega f(x)v(x)\,dx.$$

The weak residual $\mathcal{R}^w(u_{NN}, v)$ is defined as

$$\mathcal{R}^w(u_{NN}, v) = a(u_{NN}, v) - l(v). \tag{5.21}$$

To quantify the error between the approximate solution u_{NN} and the true solution u, we consider the dual norm of the residual in $H^{-1}(\Omega)$, defined as

$$\|\mathcal{R}^w(u_{NN})\|_{H^{-1}} = \sup_{v\in H_0^1(\Omega),\, \|v\|_{H_0^1}=1} \mathcal{R}^w(u_{NN}, v). \tag{5.22}$$

Minimizing this dual norm provides an estimate of the error in the $H_0^1(\Omega)$ norm due to the equivalence between the residual's dual norm and the error norm in the context of elliptic PDEs.

The Sobolev space $H_0^1(\Omega)$ has an orthonormal basis consisting of the eigenfunctions of the negative Laplacian operator $-\Delta$ with Dirichlet boundary conditions. Specifically, the set of functions $\{\varphi_n(x)\}_{n=1}^\infty$, where

$$\varphi_n(x) = \sqrt{\frac{2}{\pi}}\sin(nx), \tag{5.23}$$

forms an orthonormal basis for $H_0^1(\Omega)$ under the inner product

$$(u,v)_{H_0^1} = \int_\Omega u_x(x)v_x(x)\,dx. \tag{5.24}$$

These functions also satisfy

$$-\varphi_n''(x) = \lambda_n\varphi_n(x), \quad \lambda_n = n^2, \quad n\in\mathbb{N}. \tag{5.25}$$

Using the Fourier basis, we can expand the residual $\mathcal{R}^w(u_{NN})$ in terms of the basis functions:

$$\mathcal{R}^w(u_{NN}, v) = \sum_{n=1}^\infty c_n(u_{NN})(\varphi_n, v)_{H_0^1}, \tag{5.26}$$

where

$$c_n(u_{NN}) = \mathcal{R}^w(u_{NN}, \varphi_n). \tag{5.27}$$

The dual norm of the residual can then be expressed, by Parseval's Identity as

$$\|\mathcal{R}^w(u_{NN})\|_{H^{-1}} = \left(\sum_{n=1}^\infty |c_n(u_{NN})|^2\right)^{1/2}. \tag{5.28}$$

In practice, we approximate the infinite series by truncating it after N terms:

$$\|\mathcal{R}^w(u_{NN})\|_{H^{-1}} \approx \left(\sum_{n=1}^{N} |c_n(u_{NN})|^2 \right)^{1/2}. \tag{5.29}$$

The loss function for the DFRM is chosen as the squared truncated dual norm of the residual:

$$\mathcal{L}_{\text{DFRM}}(u_{NN}) = \sum_{n=1}^{N} |\mathcal{R}^w(u_{NN}, \varphi_n)|^2 . \tag{5.30}$$

Since the basis functions φ_n are orthonormal in $H_0^1(\Omega)$, the Gram matrix $\mathbf{G}$ in the RVPINN framework becomes the identity matrix, simplifying computations.

In conclusion, the loss function for the DFRM is

$$\mathcal{L}_{\text{DFRM}}(u_{NN}) = \sum_{n=1}^{N} a(u_{NN}, \varphi_n) - l(\varphi_n) \tag{5.31}$$

$$= \sum_{n=1}^{N} \int_\Omega u'_{NN}(x)\varphi'_n(x)\, dx - \int_\Omega f(x)\varphi_n(x)\, dx. \tag{5.32}$$

In order to compute the loss numerically, the integrals appearing in (5.31) must be approximated via numerical quadrature. Consider a set of quadrature points $\{x_i\}_{i=1}^M$ in Ω with associated quadrature weights $\{w_i\}_{i=1}^M$. Then, the integrals are approximated as follows:

$$\int_\Omega u'_{NN}(x)\varphi'_n(x)\, dx \approx \sum_{i=1}^{M} w_i\, u'_{NN}(x_i)\, \varphi'_n(x_i), \tag{5.33}$$

$$\int_\Omega f(x)\varphi_n(x)\, dx \approx \sum_{i=1}^{M} w_i\, f(x_i)\, \varphi_n(x_i). \tag{5.34}$$

Thus, the discretized version of the loss becomes:

$$\mathcal{L}^{\text{disc}}_{\text{DFRM}}(u_{NN}) = \sum_{n=1}^{N} \left(\sum_{i=1}^{M} w_i\, u'_{NN}(x_i)\, \varphi'_n(x_i) - \sum_{i=1}^{M} w_i\, f(x_i)\, \varphi_n(x_i) \right)^2 . \tag{5.35}$$

This formulation allows the loss to be computed directly from pointwise evaluations of the neural network derivative, the source term f, and the test functions $\{\varphi_n\}$ and their derivatives at the quadrature points.

In the computational implementation, we use automatic differentiation to compute $u'_{NN}(x)$ and evaluate the integrals numerically over the domain Ω. The use of the Fourier basis allows us to exploit the orthogonality properties to simplify the calculation of the loss function.

The truncation parameter N controls the accuracy of the approximation of the dual norm. Increasing N leads to a more accurate estimation of the residual norm but also increases computational cost.

Remark 4. To enforce homogeneous Dirichlet boundary conditions on the candidate solutions, we introduce a non-trainable cut-off function $\xi : \bar{\Omega} \to \mathbb{R}$. If $\tilde{u}$ denotes the output of the fully connected feed-forward neural network, then the final solution is defined as $u_{NN} = \xi\tilde{u}$. The function ξ is chosen to satisfy $\xi|_{\Gamma_D} = 0$ on the Dirichlet boundary and $\xi > 0$ on $\bar{\Omega} \setminus \Gamma_D$. For more details, we refer to [42].

5.4.1 Numerical Experiment

As before, here we present a numerical experiment to illustrate the behavior of the DFRM approach for solving partial differential equations.

We consider the toy problem presented in Section 4.3, where $\Omega = (0, \pi)$, the source term $f(x) = -4\sin(2x)$, and zero Dirichlet boundary conditions. This leads to the exact solution $u_{\text{exact}}(x) = \sin(2x)$.

Now, the architecture consists of five hidden layers, each containing 10 neurons, with the hyperbolic tangent (tanh) serving as the activation function throughout. The network parameters are optimized using the Adam optimizer with a learning rate of 10^{-3}. Moreover, we take $N_{\text{test}} = 10$; this is the first ten Fourier modes (basis functions), and we use randomly distributed points and a mid-point integration rule with 200 integration points for training.

Figure 5.1 shows the comparison between the exact solution and the neural network approximation u_{NN}. The quality of these results highlights the relevance of imposing the boundary conditions strongly on the NN model.

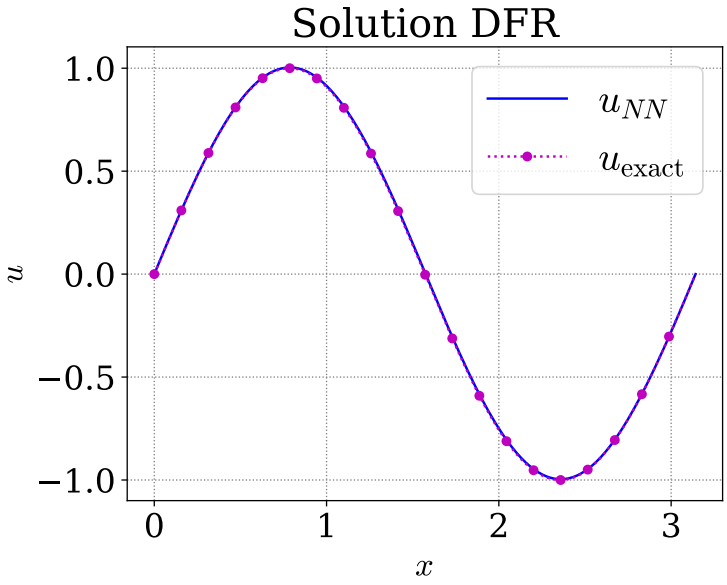


Figure 5.1: Approximation of the neural network (u_{NN}) to the exact solution (u_{exact})

In Figure 5.2, we display the evolution of the loss function computed with respect to the H_0^1-norm of the exact solution, which highlights the practical relevance of the equivalence

established in (3.34). Additionally, the same figure presents a validation curve obtained using ten times as many integration points as those used for computing the training loss. This validation curve is included to ensure that the numerical errors inherent in the integration process do not affect the learning curve of the model. A comparison between the loss, the validation curve, and the relative H_0^1-norm of the error clearly demonstrates that minimizing the loss function indeed leads to a corresponding reduction in the true error. This observation reinforces the robustness of the method in effectively controlling the error.

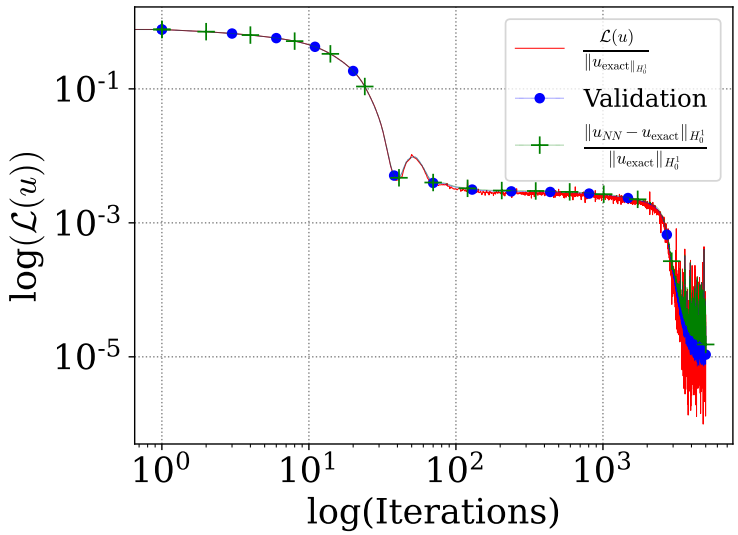


Figure 5.2: Evolution of the loss function (5.4), the validation data, and the relative error of the approximation

Chapter 6

Application of the method to the diffusion-advection equation and to the heat equation.

6.1 Motivation: The diffusion-advection equation

In [37], the authors propose a RVPINNs approach to solve the advection-diffusion problem. We present a detailed presentation of the main results. This helped us as a motivation to try to tackle the heat equation with Dirichlet boundary condition.

Our objective is to show that the advection-diffusion equation can be approximated using a robust variational physics-informed neural network (RVPINNS) approach. Consider the following steady-state advection-diffusion equation on a bounded Lipchitz polyhedron $\Omega \subset \mathbb{R}^n$ (with boundary $\partial\Omega$):

$$-\nabla \cdot \big(\varepsilon \nabla u\big) \; + \; \beta \cdot \nabla u \; = \; f, \quad \text{in } \Omega, \tag{6.1}$$

subject to homogeneous Dirichlet boundary conditions $u = 0$ on $\partial\Omega$. The coefficient $\varepsilon > 0$ represents the diffusive coefficient, the vector field $\beta \in [L^\infty(\Omega)]^n$ is the (divergence-free) advective velocity, and $f \in L^2(\Omega)$ is the source term.

Variational Formulation

To derive the variational form of (6.1), we multiply the PDE by a test function $v \in H_0^1(\Omega)$ and integrate over Ω:

$$\int_\Omega \Big(-\nabla \cdot (\varepsilon \nabla u)\Big)\, v \, dx \; + \; \int_\Omega \big(\beta \cdot \nabla u\big)\, v \, dx \; = \; \int_\Omega f \, v \, dx. \tag{6.2}$$

Diffusion Term. The first term in (6.2) corresponds to diffusive transport:

$$\int_\Omega -\nabla \cdot (\varepsilon \nabla u)\, v\, dx.$$

Applying the divergence theorem yields

$$\int_\Omega -\nabla \cdot (\varepsilon \nabla u)\, v\, dx \;=\; \int_\Omega \varepsilon \nabla u \cdot \nabla v\, dx \;-\; \int_{\partial\Omega} \varepsilon\, (\nabla u \cdot n)\, v\, dS.$$

Since $v \in H_0^1(\Omega)$ enforces $v = 0$ on $\partial\Omega$, the boundary integral vanishes, and we obtain

$$\int_\Omega -\nabla \cdot (\varepsilon \nabla u)\, v\, dx \;=\; \int_\Omega \varepsilon \nabla u \cdot \nabla v\, dx.$$

Advection Term. The second term,

$$\int_\Omega (\beta \cdot \nabla u)\, v\, dx,$$

reflects advective transport. Combining both terms, the variational form of (6.1) becomes

$$\int_\Omega \varepsilon \nabla u \cdot \nabla v\, dx \;+\; \int_\Omega (\beta \cdot \nabla u)\, v\, dx \;=\; \int_\Omega f\, v\, dx,$$

for all $v \in H_0^1(\Omega)$. We denote $H_0^1(\Omega)$ as our trial and test space (denoted by U and V, respectively, in an abstract sense).

Hence, the problem admits the continuous variational formulation: find $u \in H_0^1(\Omega)$ such that

$$a(u, v) \;=\; l(v), \quad \forall\, v \in H_0^1(\Omega),$$

where the bilinear form $a(\cdot, \cdot)$ and linear functional $l(\cdot)$ are given by

$$\begin{aligned} a(u, v) &:= \int_\Omega \varepsilon \nabla u \cdot \nabla v\, dx \;+\; \int_\Omega (\beta \cdot \nabla u)\, v\, dx, \\ l(v) &:= \int_\Omega f\, v\, dx. \end{aligned} \tag{6.3}$$

We consider the norm

$$\|\cdot\|_{H_0^1} \;:=\; \big(\nabla\cdot, \nabla\cdot\big)_{L^2}^{1/2},$$

on $H_0^1(\Omega)$.

Continuity of the Bilinear Form

To apply the Lax–Milgram theorem (7.1), we need $a(\cdot, \cdot)$ to be continuous and coercive on $H_0^1(\Omega)$. For continuity, by the triangle and Cauchy–Schwarz inequalities,

$$\begin{aligned} |a(u, v)| &\le \Big|\int_\Omega \varepsilon \nabla u \cdot \nabla v\, dx\Big| \;+\; \Big|\int_\Omega (\beta \cdot \nabla u)\, v\, dx\Big| \\ &\le \varepsilon \|\nabla u\|_{L^2} \|\nabla v\|_{L^2} \;+\; \|\beta\|_\infty \|\nabla u\|_{L^2}\, C_\Omega\, \|\nabla v\|_{L^2} \\ &\le \Big(\varepsilon + C_\Omega\, \|\beta\|_\infty\Big)\, \|u\|_{H_0^1} \|v\|_{H_0^1}, \end{aligned}$$

where C_Ω stems from the Poincaré inequality, ensuring $\|v\|_{L^2} \leq C_\Omega \|\nabla v\|_{L^2}$. Hence, $a(\cdot,\cdot)$ is continuous on $H_0^1(\Omega)$.

Coercivity of the Bilinear Form

To demonstrate coercivity, we note that

$$a(v,v) \;=\; \int_\Omega \varepsilon\,|\nabla v|^2\,dx \;+\; \int_\Omega (\beta\cdot\nabla v)\,v\,dx.$$

Since $v \in H_0^1(\Omega)$ vanishes on the boundary $\partial\Omega$, and $\nabla\cdot\beta = 0$, the second term simplifies to zero by integration by parts and the divergence-free condition on β. Thus,

$$a(v,v) \;=\; \varepsilon\|\nabla v\|_{L^2}^2 \;\geq\; \varepsilon\|v\|_{H_0^1}^2.$$

Hence, $a(\cdot,\cdot)$ is coercive with coercivity constant $\varepsilon > 0$.

Stability and Equivalence of Residual and Error

Having verified continuity and coercivity, the Babuška–Brezzi (or Lax–Milgram) theorem guarantees that the variational problem is well-posed. Moreover, the coercivity condition implies that the norm of the error $\|u-u^*\|_{H_0^1}$ is equivalent (up to constants) to the dual norm of the residual $\|\mathcal{R}(u)\|_{H^{-1}}$. In particular:

$$\frac{1}{M}\,\|\mathcal{R}(u)\|_{H^{-1}} \;\leq\; \|u-u^*\|_{H_0^1} \;\leq\; \frac{1}{\gamma}\,\|\mathcal{R}(u)\|_{H^{-1}},$$

where $M > 0$ and $\gamma > 0$ are constants determined by the continuity and the coercivity condition.

6.2 The Parabolic Problem: Variational Formulation

In this section, we consider the heat equation: Let $\Omega \subset \mathbb{R}^n$ a open bounded domain with Lipchitz boundary $\partial\Omega$,

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} \;-\; \Delta u(x,t) \;&=\; f(x,t), \quad \forall x\in\Omega, \quad t>0 \\ u(x,0) &= g_0(x), \quad \forall x\in\Omega, \\ u(x,t) &= g_1(t), \quad \forall x\in\partial\Omega, \quad t>0 \end{aligned} \tag{6.4}$$

where u is the unknown function, $f \in L^2(\Omega)$ and $g_i(x)$ correspond to the boundary and initial conditions. For the sake of simplicity, we take $g_1 = 0$.

Our approach is to discretize the time variable, leading to a sequence of steady-like problems.

Let Δt be the time-step size, and define $t^n = n\,\Delta t$. Denote

$$u^n(x) := u(x, t^n), \quad f^n(x) := f(x, t^n).$$

An approximation of the time derivative $\frac{\partial u(x,t)}{\partial t}$ by a finite difference yields

$$\frac{u^n(x) - u^{n-1}(x)}{\Delta t}.$$

Thus, the heat equation (6.4) becomes

$$\frac{u^n(x) - u^{n-1}(x)}{\Delta t} \;-\; \Delta\, u^n(x) \;=\; f^n(x),$$

which can be rearranged to

$$u^n(x) \;-\; \Delta t\, \Delta u^n(x) \;=\; u^{n-1}(x) \;+\; \Delta t\, f^n(x). \tag{6.5}$$

Remark 5. The backward Euler method, also known as the implicit Euler method, is a first-order method used for solving ordinary differential equations. Its primary advantage lies in its unconditional stability, making it particularly suitable for stiff differential equations. As described in [12] chapter 5, the method discretizes the derivative implicitly and requires solving a nonlinear equation at each time step, but its stability ensures reliable solutions even for large time steps. The backward Euler method is often a preferred choice in problems where long-term stability is crucial, such as in parabolic and stiff systems.

Notice that the right-hand side, $u^{n-1}(x) + \Delta t\, f^n(x)$, can be viewed as a new source term

$$g(x) := u^{n-1}(x) + \Delta t\, f^n(x),$$

since u^{n-1} is known from the previous time-step when solving for u^n.

We focus on a variational formulation of (6.5). Let $H_0^1(\Omega)$ be the test space, and multiply (6.5) by a test function $v \in H_0^1(\Omega)$. Then,

$$u^n\, v \;-\; \Delta t\, \Delta u^n\, v \;=\; g\, v$$

Integrating over Ω, we get

$$\int_\Omega u^n\, v\, dx \;-\; \Delta t \int_\Omega \Delta u^n\, v\, dx \;=\; \int_\Omega g\, v\, dx. \tag{6.6}$$

The diffusion term can be simplified via integration by parts (Green's First Identity). Specifically,

$$-\,\Delta t \int_\Omega \Delta u^n\, v\, dx \;=\; \Delta t \int_\Omega \nabla u^n \cdot \nabla v\, dx \;-\; \Delta t \int_{\partial\Omega} \big(\nabla u^n \cdot \mathbf{n}\big)\, v\, ds.$$

Since $v \in H_0^1(\Omega)$ vanishes on $\partial\Omega$, the boundary term is zero. Hence, (6.6) becomes

$$\int_\Omega u^n\, v\, dx \;+\; \Delta t \int_\Omega \nabla u^n \cdot \nabla v\, dx \;=\; \int_\Omega g\, v\, dx.$$

Thus, the variational formulation is: find $u^n \in H_0^1(\Omega)$ such that

$$a(u^n,\, v) \;=\; l(v), \quad \forall\, v \in H_0^1(\Omega),$$

where the bilinear form $a(\cdot,\cdot)$ and linear functional $l(\cdot)$ are given by

$$\begin{aligned} a(u,v) &:= \int_\Omega u\, v\, dx \;+\; \int_\Omega \varepsilon\, \nabla u \cdot \nabla v\, dx, \\ l(v) &:= \int_\Omega g\, v\, dx. \end{aligned} \tag{6.7}$$

Continuity of the Bilinear Form

Define the norm

$$\|w\|_{H_0^1} := \|\nabla w\|_{L^2}.$$

We verify that $a(\cdot,\cdot)$ is continuous on $H_0^1(\Omega)$. By the triangle and Cauchy–Schwarz inequalities,

$$|a(u,v)| \;\le\; \varepsilon\, \|\nabla u\|_{L^2}\, \|\nabla v\|_{L^2} \;+\; \|u\|_{L^2}\, \|v\|_{L^2}.$$

Using Poincaré's inequality, $\|u\|_{L^2} \le C_\Omega\, \|\nabla u\|_{L^2}$, we get

$$|a(u,v)| \;\le\; \varepsilon\, \|\nabla u\|_{L^2}\, \|\nabla v\|_{L^2} \;+\; C_\Omega^2\, \|\nabla u\|_{L^2}\, \|\nabla v\|_{L^2} \;=\; (\varepsilon + C_\Omega^2)\, \|u\|_{H_0^1}\, \|v\|_{H_0^1}.$$

Hence, $a(\cdot,\cdot)$ is continuous with $M = \varepsilon + C_\Omega^2$.

Coercivity of the Bilinear Form

To show coercivity, notice that

$$a(u,u) \;=\; \int_\Omega u^2\, dx \;+\; \varepsilon \int_\Omega |\nabla u|^2\, dx.$$

Since $\int_\Omega u^2\, dx \ge 0$, we have

$$a(u,u) \;\ge\; \varepsilon\, \|\nabla u\|_{L^2}^2 \;=\; \varepsilon\, \|u\|_{H_0^1}^2.$$

Thus $a(\cdot,\cdot)$ is coercive on $H_0^1(\Omega)$ with coercivity constant $\alpha = \varepsilon$.

When the trial and test spaces coincide in a single Hilbert space, coercivity alone ensures the inf-sup condition is automatically satisfied. Concretely,

$$\inf_{u\neq 0} \sup_{v\neq 0} \frac{|a(u,v)|}{\|u\|_{H_0^1}\, \|v\|_{H_0^1}} \;\ge\; \alpha \;=\; \varepsilon.$$

Hence, $a(\cdot,\cdot)$ meets both continuity and coercivity, implying a unique solution to the discrete time-step problem via the Lax–Milgram or Babuška–Brezzi theorem (15) .

For each time step n, the parabolic problem (6.5) reduces to a steady-like variational problem in $H_0^1(\Omega)$ with the bilinear form $a(\cdot,\cdot)$ from (6.7) and a right-hand side $l(\cdot)$. Coercivity and continuity guarantee the well-posedness of each time-step solution, and one obtains the approximate evolution by iterating in time.

6.3 Construction of the loss function

In the 1-dimensional setting we are interested in solving the following partial differential equation on $\Omega = (0, \pi)$, commonly referred to as the heat equation with Dirichlet boundary conditions:

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} - \frac{\partial^2 u(x,t)}{\partial x^2} &= f(x,t), \quad \forall x \in [0,\pi], \\ u(0,t) &= u(\pi,t) = 0, \qquad \forall t \in [0,T], \quad T > 0 \\ u(x,0) &= g_0(x), \quad \forall x \in [0,\pi]. \end{aligned} \tag{6.8}$$

We discretize the interval $[0, T]$ by setting $\Delta t = T/N_{\text{time}}$ for a given N_{time}. Then $t^n := \Delta t\, n$, and by replacing

$$\frac{\partial u(x,t)}{\partial t} \approx \frac{u(x,t^n) - u(x,t^{n-1})}{\Delta t},$$

we can rewrite equation (6.8) as:

$$u(x,t^n) - \Delta t \frac{\partial^2 u(x,t^n)}{\partial x^2} = u(x,t^{n-1}) + \Delta t\, f(x,t^n) \tag{6.9}$$

Since we want to implement a RVPINN to get an approximation of the solution of this problem, let us take again a variational approach, by considering a test space $H_0^1(\Omega)$ where Ω is an interval $(a, b) \subset \mathbb{R}$. By multiplying by $v \in H_0^1(\Omega)$ and integrating over all Ω, by denoting $u^n(\cdot) := u(\cdot, t^n)$ we can write (6.9) as

$$\int_\Omega u^n(x)\, v(x)\, dx - \Delta t \int_\Omega \frac{\partial^2 u^n(x)}{\partial x^x} v(x)\, dx = \int_\Omega u^{n-1}(x)\, v(x) + \Delta t\, f^n(x)\, v(x)\, dx \tag{6.10}$$

By using integration by parts in the second term

$$\int_\Omega \frac{\partial^2 u^n(x)}{\partial x^2}\, v(x)\, dx = \left[\frac{\partial u^n(x)}{\partial x}\, v(x)\right]_{\partial\Omega} - \int_\Omega \frac{\partial u^n(x)}{\partial x}\, \frac{\partial v(x)}{\partial x}\, dx.$$

And since $v \in H_0^1(\Omega)$, then

$$\left[\frac{\partial u^n(x)}{\partial x}\, v(x)\right]_{\partial\Omega} = 0,$$

and then (6.10) becomes:

$$\int_\Omega u^n(x)\, v(x)\, dx + \Delta t \int_\Omega \frac{\partial u^n(x)}{\partial x} \frac{\partial v(x)}{\partial x}\, \partial x = \int_\Omega u^{n-1}(x)\, v(x) + \Delta t\, f^n(x)\, v(x)\, dx \tag{6.11}$$

In this way, we can then define the weak residual of the variational heat equation in one dimension as

$$\begin{aligned} \mathcal{R}_n^w(u^n), v) := &\int_\Omega u^n(x)\, v(x)\, dx \; + \; \Delta t \int_\Omega \frac{\partial u^n(x)}{\partial x} \frac{\partial v(x)}{\partial x}\, dx \\ &- \int_\Omega u^{n-1}(x)\, v(x) - \Delta t\, f^n(x)\, v(x)\, dx \qquad \forall v \in H_0^1(\Omega) \end{aligned} \tag{6.12}$$

At this point, to obtain an approximation of the residual, we need to start to discretize even more the problem and then by selecting a base for the test space $H_0^1(\Omega)$, $\{\varphi_k\}_{k=1}^{\infty}$ and, recall (2.48), (2.49) and (2.50), from which:

$$\varphi_k(x) := \sqrt{\frac{2}{\pi}}\,\frac{\sin(kx)}{k} \qquad \forall k \in \mathbb{N},$$
$$d_x\,\varphi_k(x) = \sqrt{\frac{2}{\pi}}\,\cos(kx) \qquad \forall k \in \mathbb{N},$$

forms an orthonormal basis for $H_0^1(\Omega)$.

We recall from (3.34) that we want to estimate the error

$$\|u^n - u^*(\cdot, t^n)\|_{H_0^1}.$$

where u^* denotes the exact solution of the parabolic problem.

For this aim, we need to approximate the dual norm of the residual in $H^{-1}(\Omega)$, i.e.,

$$\|\mathcal{R}_n^w(u, v)\|_{H^{-1}}$$

To calculate the H^{-1}-norm, we use the Riesz Representation Theorem 5 and ensure that there is a unique $\psi^n \in H_0^1(\Omega)$ such that for all $n \in \mathbb{N}$:

$$\mathcal{R}_n^w(u^n, v) = (\psi^n, v)_{H_0^1}, \qquad \forall v \in H_0^1(\Omega)$$

and

$$\|\mathcal{R}_n^w(u^n, v)\|_{H^{-1}} = \|\psi^n\|_{H_0^1}$$

when we expand ψ^n in the orthonormal basis $\{\varphi_k\}$ and truncate the approximation using the first N_{test} basis functions, we obtain

$$\psi^n \approx \sum_{k=1}^{N_{\text{test}}} c_k^n \varphi_k$$

then,

$$\|\psi^n\|_{H_0^1}^2 \approx \left(\sum_{k=1}^{N_{\text{test}}} c_k^n \varphi_k, \sum_{k=1}^{N_{\text{test}}} c_k^n \varphi_k \right)_{H_0^1}$$

and, using the orthonormality of the basis functions $(\varphi_j, \varphi_k)_{H_0^1} = \delta_{j,k}$, so, we obtain

$$\|\psi^n\|_{H_0^1}^2 \approx \sum_{k=1}^{N_{\text{test}}} \sum_{j=1}^{N_{\text{test}}} c_j^n c_k^n (\varphi_j, \varphi_k)_{H_0^1} = \sum_{k=1}^{N_{\text{test}}} |c_k^n|^2.$$

Furthermore,

$$(\psi^n, \varphi_k)_{H_0^1} \approx \sum_{j=1}^{N_{\text{test}}} c_j^n (\varphi_j, \varphi_k)_{H_0^1} = \sum_{j=1}^{N_{\text{test}}} c_j^n \delta_{jk} = c_k^n.$$

And, we conclude that

$$\|\mathcal{R}_n^w(u^n, v)\|_{H^{-1}}^2 = \|\psi^n\|_{H_0^1}^2 \approx \sum_{k=1}^{N_{\text{test}}} |c_k^n|^2 = \sum_{k=1}^{N_{\text{test}}} |(\psi^n, \varphi_k)_{H_0^1}|^2 = \sum_{k=1}^{N_{\text{test}}} |\mathcal{R}_n^w(u^n, \varphi_k)|^2 \tag{6.13}$$

which is often referred to as **Parseval's identity**.

Finally, our goal is then to define a general loss, this is: to minimize the weighted sum of the residuals over all the time steps of the discretization, leaving us with:

$$\mathcal{L}(u) = \Delta t \sum_{n=1}^{N_{\text{time}}} \|\mathcal{R}_n^w(u^n, \phi_k)\|_{H^{-1}}^2 \approx \Delta t \sum_{n=1}^{N_{\text{time}}} \sum_{k=1}^{N_{\text{test}}} |\mathcal{R}_n^w(u^n, \varphi_k)|^2 \tag{6.14}$$

This loss function can be expressed explicitly as:

$$\begin{aligned} \mathcal{L}(u(x,t)) = \Delta t \sum_{n=1}^{N_{\text{time}}} \sum_{k=1}^{N_{\text{test}}} \Bigg(& \int_\Omega u^n(x)\, \varphi_k(x)\, dx \, + \, \Delta t \int_\Omega \frac{\partial u^n(x)}{\partial x} \frac{\partial \varphi_k(x)}{\partial x}\, dx \\ & - \int_\Omega u^{n-1}(x)\, \varphi_k(x)\, dx \, - \, \Delta t \int_\Omega f^n(x)\, \varphi_k(x)\, dx \Bigg)^2 . \end{aligned} \tag{6.15}$$

To compute the loss function, we implement a quadrature rule to approximate the value of the integrals.

Numerical integration

Let us consider parabolic problems on 1-dimension with $\Omega = [0, \pi]$. In this interval, we use the midpoint rule for approximating the integrals. So given a discretization of the interval, this is a sample of points $\{\hat{x}_i\}_{i=0}^{N_S+1}$ where N_{S} is the number of sampled points. Denote by:

$$\begin{aligned} x_i &= \frac{\hat{x}_{i-1} + \hat{x}_i}{2}, \\ w_i &= \hat{x}_i - \hat{x}_{i-1}. \end{aligned}$$

which we can use to approximate each of the integral terms of the loss function, i.e.,

$$\begin{aligned} \int_0^\pi u^n(x)\varphi_k(x)\, dx &\approx \sum_{i=1}^{N_{\text{S}}} w_i\, u^n(x_i)\, \varphi_k(x_i) \\ \Delta t \int_0^\pi \frac{\partial\, u^n(x)}{\partial x} \frac{\partial\, \varphi_k(x)}{\partial x}\, dx &\approx \sum_{i=1}^{N_{\text{S}}} w_i\, \frac{\partial\, u^n}{\partial x}(x_i) \frac{\partial\, \varphi_k}{\partial x}(x_i) \\ \int_0^\pi u^{n-1}(x)\varphi_k(x) \, - \Delta t f(x)^n \varphi_k(x) dx &\approx \sum_{i=1}^{N_{\text{S}}} w_i\, u(x_i)^{n-1}(x_i)\varphi_k(x_i) \, - \Delta t f(x_i)^n \varphi_k(x_i) \end{aligned} \tag{6.16}$$

then the discretized loss function (6.15) is

$$\mathcal{L}(u) := \Delta t \sum_{n=1}^{N_{\text{time}}} \sum_{k=1}^{N_{\text{test}}} \left(\sum_{i=1}^{N_S} w_i \left[u^n(x_i)\, \varphi_k + \frac{\partial\, u^n}{\partial x}(x_i) \frac{\partial\, \varphi_k}{\partial x}(x_i) \right.\right. \\ \left.\left. - u^{n-1}(x_i)\, \varphi_k(x_i) - \Delta t\, f^n(x_i)\, \varphi_k(x_i) \right] \right)^2 . \tag{6.17}$$

6.4 Numerical experiment

In this section, we present a numerical experiment designed to illustrate the proposed method for solving the parabolic problem (6.8).

We seek a weak solution of the following strong problem:

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} - \frac{\partial^2 u(x,t)}{\partial x^2} &= 0, \quad \forall x \in [0,\pi], \\ u(0,t) &= u(\pi,t) = 0, \qquad \forall t \in [0,1] \\ u(x,0) &= \sin(x), \quad \forall x \in [0,\pi], \end{aligned} \tag{6.18}$$

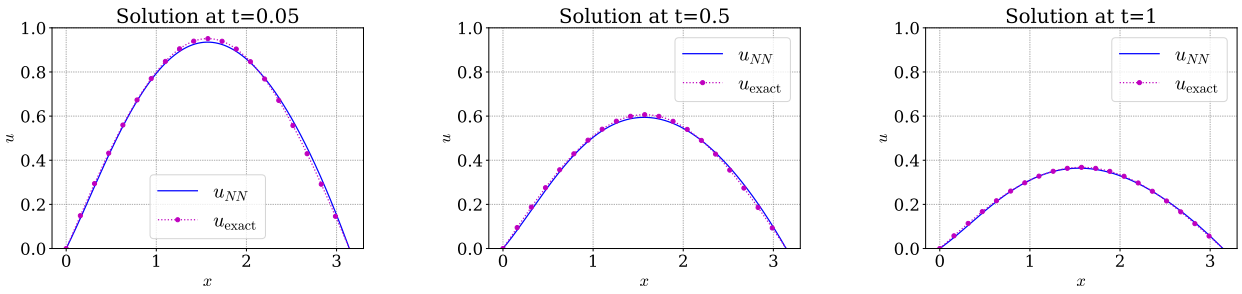
with an exact solution $u_{\text{exact}} = e^{-t}\sin(x)$.

As before, we use *TensorFlow 12.8* to build a feed-forward, fully-connected neural network. The architecture consists of five hidden layers, each containing 10 neurons, with the hyperbolic tangent (tanh) serving as the activation function. The network parameters are optimized using the Adam optimizer with a learning rate of 10^{-3}. In this case, the architecture of the network is modified so that its output includes as many time steps as the time discretization in our formulation. For this experiment, we take 200 time steps, leading to $\Delta t = 0.005$.

In computing the loss, we use a discretization with 20 test functions, i.e., the first 20 Fourier modes approximate the test space at each time step. For spatial integration, we use 250 randomly distributed points and compute the integrals in – using the midpoint integration rule to minimize additional computational costs.

In Figure 6.1, we compare the exact solution u^n with the neural network approximation u^n_{NN} at three different time steps. The homogeneous Dirichlet boundary conditions are fully satisfied at every point, as we apply the cut-off strategy described in the previous Section. This approach enforces the boundary conditions directly by multiplying each network output by the function $\xi = x(x-\pi)$.

In Figure 6.1, we present the evolution of the loss and the error on a log-log plot. To ensure that the integration error is controlled, we include a validation curve where the loss function is computed using five times more integration points. This ensures that the approximation error on the loss is not significantly affected by numerical integration inaccuracies.



(a) Step $n = 10$ (b) Step $n = 100$ (c) Step $n = 200$

Figure 6.1: Neural network solution (u_{NN}^n) vs the exact solution at three time steps

Moreover, there is a clear relationship between the loss and the error trends. Notably, when the loss undergoes its steepest descent, a similar behavior is observed in the error curve. This correlation is further illustrated in Figure 6.3, where we display the loss (relative) vs the relative error. Although the error does not initially decrease during the early stages of the minimization process, a strong alignment between minimizing the loss and reducing the error becomes evident as training progresses. This reinforces the robustness of our approach in capturing the expected solution behavior.

Observe that after 2000 iterations, oscillations in the loss affect the decreasing trend of the error. This issue arises due to an insufficient number of test functions considered in the loss computation and the uncontrolled behavior of the optimization method. A possible solution is to incorporate alternative strategies, such as applying a least-squares approach in the final layer of the neural network. However, this comes at the expense of increased computational costs. For further details on similar approaches, we refer to [3], where the implementation is discussed.

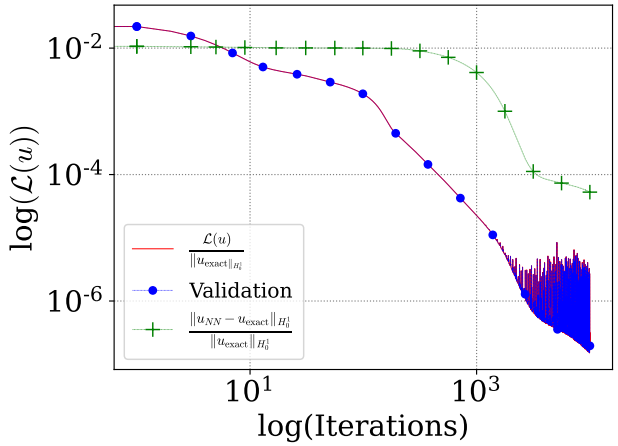


Figure 6.2: Evolution of the loss function (6.17)

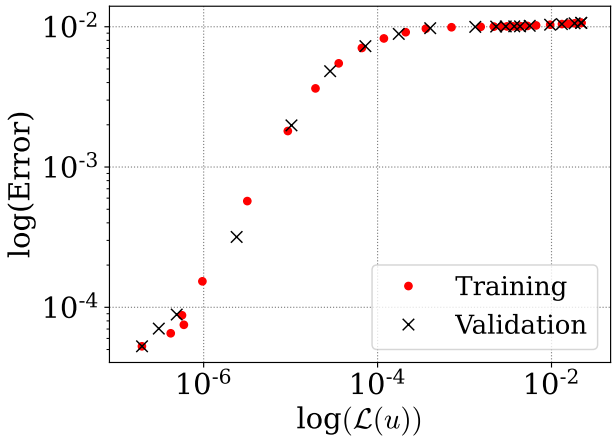


Figure 6.3: Relative error vs the loss evolution.

Chapter 7

Physics-Informed Neural Networks for Industrial-Scale Freezing

Industrial-scale freezing of coffee extracts is crucial to preserving their freshness, aroma, and overall quality. Proper freezing stabilizes the product, maintains essential flavor compounds, and ensures compliance with food safety regulations [7]. Unlike simple substances, coffee extracts are complex solutions containing dissolved solids and water. This complexity causes variations in freezing temperatures, making the freezing process challenging and unpredictable [39].

Following the experimental approach made in [34], a batch of 25 kg of coffee extract was placed inside a cylindrical drum with a diameter of 51 cm and a height of 35 cm, constructed of brass and sealed with an aluminum lid. A plastic liner was placed inside the drum to prevent leakage, and an Emerson TX60D-PAK00 temperature sensor was inserted at the core of the drum to record internal temperatures. Additional sensors were placed at the drum's surface and surrounding areas to monitor heat exchange. The drum was placed in a freezer maintained at -25° C, and the freezing process was monitored for two weeks, as detailed in previous freezing studies [31]. The objective of the model is to provide predictions of the spatial and temporal temperature evolution within the coffee extract, accounting for changes in its thermal properties. To ensure the model's simplicity while capturing essential freezing dynamics, the following assumptions were made:

- The coffee extract remains stationary, with no internal convection or mixing.
- Heat transfer within the extract and drum walls occurs through conduction, while convective heat transfer is considered at the outer surfaces exposed to the freezer air.
- The initial temperature of the extract and drum is uniform, and no mass inflows or outflows occur during the batch freezing proces.

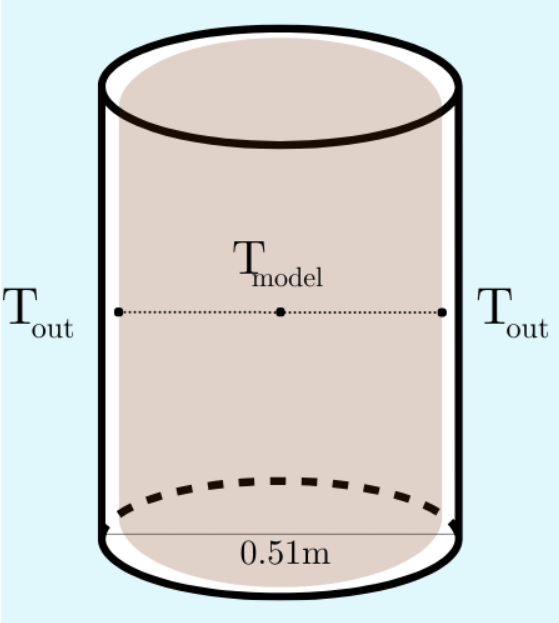


Figure 7.1: Cylindrical drum with a diameter of 0.51 m. A temperature sensor in the core of the drum's surface records the internal temperature. Additional sensors measure the drum's surface temperature and heat exchange with the surrounding freezer at $-25^{\circ}C$.

To capture the essential thermal behavior during freezing, we focus on the horizontal centerline of the cylinder, where experimental temperature measurements, denoted by T_{model}, are obtained. These measurements correspond to the middle point of the cylinder. Moreover, we assume that the coffee extract initially has a uniform temperature throughout the cylinder T_{init}. It is clear that, using a one-dimensional horizontal cross-section through the center to describe the overall thermal behavior of the system.

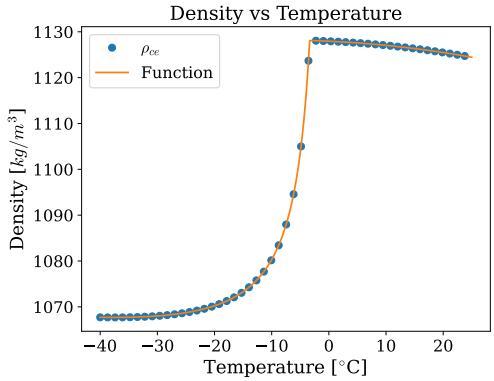
We model the centerline as the one-dimensional interval $[0, 0.51] \subset \mathbb{R}$. Outside the boundaries of this interval, the external temperature is T_{out}. Due to the lack of direct temperature measurements at the actual wall, we employ a linear interpolation to estimate the wall temperature, providing the necessary boundary conditions for the model T_{bound}.

The conservation of energy principle is applied to the coffee extract system, accounting for heat transfer and energy balance within the batch. Since no mass enters or leaves the system during freezing, the internal energy balance is described as:

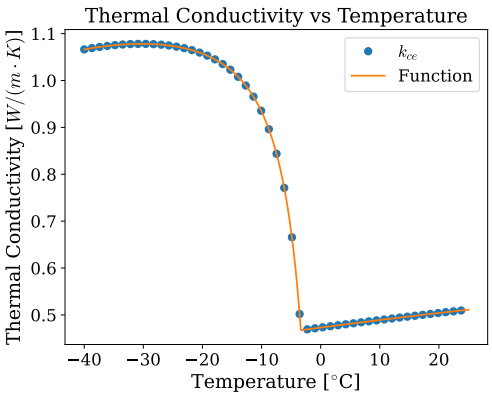
$$\begin{aligned} \frac{\partial(\rho(T)c(T)T)}{\partial t} - \nabla \cdot (k(T)\nabla T) = 0, \qquad & x \in (0, 0.51) \text{ and } t \geq 0, \\ T(0.51, t) = T(0, t) = T_{\text{bound}}, \quad & t \geq 0 \\ T(x, 0) = T_{\text{init}}, \qquad & x \in [0, 0.51]. \end{aligned} \tag{7.1}$$

In (7.5), t is the time parameter, measured in seconds [s], x represents the spatial coordinate measured in meters [m], T is the temperature measured in degrees Celsius [$^{\circ}C$], ρ is the density of the extract, measured in kilograms per squared meter [kg/m^3], the heat

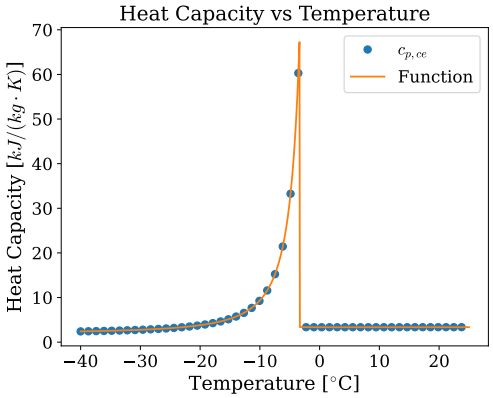
capacity c is measured in Joules per kilogram per degree Celsius [J/kg°C] and the Thermal conductivity k is measured in Watts per meter per degree Celsius [W/m°C]



(a) Density.



(b) Thermal conductivity.



(c) Heat capacity.

Figure 7.2: Thermophysical properties of the coffee extract as a function of temperature. The blue dots represent the experimental data, while the orange curves describe the fitted model used for the simulation. When the coffee extract reaches zero °C, the phase change drastically changes the physical properties.

7.0.1 Non-dimensional formulation

The parametric formulation introduces numerical difficulties, particularly as the temperature approached the freezing point. Small variations in temperature became difficult to distinguish, effectively masking the movement of the temperature front. To mitigate these issues, we rescaled the parameters into dimensionless quantities. By doing so, we preserve the relevant dynamics of the system while avoiding the disappearance of small temperature gradients.

We first introduce the dimensionless temperature by letting

$$\theta := \frac{T}{T_0},$$

where T_0 is the initial temperature of the system. Equation (7.5) then reads

$$\frac{\partial(\rho(T)\,c(T)\,\theta)}{\partial t} - \nabla\cdot(k(T)\nabla\theta) = 0. \tag{7.2}$$

Next, we define the dimensionless spatial coordinate and time variable by

$$s := \frac{x}{L} \quad \text{and} \quad \tau := \frac{t}{t_0},$$

where L is the characteristic length of the domain and t_0 is a reference time (e.g., the final time of the experiment).

To rewrite the derivatives in terms of s and τ, we apply the chain rule. Since

$$s = \frac{x}{L} \quad \Longrightarrow \quad \frac{\partial s}{\partial x} = \frac{1}{L},$$

we obtain

$$\frac{\partial T}{\partial x} = \frac{1}{L}\frac{\partial T}{\partial s} \quad \text{and} \quad \frac{\partial^2 T}{\partial x^2} = \frac{1}{L^2}\frac{\partial^2 T}{\partial s^2}.$$

Similarly, with $\tau = \frac{t}{t_0}$ it follows that

$$\frac{\partial T}{\partial t} = \frac{1}{t_0}\frac{\partial T}{\partial \tau}.$$

Substituting these expressions into (7.2) yields

$$\frac{1}{t_0}\frac{\partial}{\partial \tau}\big(\rho(T)\,c(T)\,\theta\big) - \frac{1}{L^2}\frac{\partial}{\partial s}\Big(k(T)\,\frac{\partial \theta}{\partial s}\Big) = 0. \tag{7.3}$$

Multiplying (7.3) by L^2 and dividing by $\min_T\{k(T)(T)\} := k_0$ gives

$$\frac{\partial}{\partial \tau}\left(\frac{L^2\,\rho(T)\,c(T)}{t_0\,k_0}\,\theta\right) - \frac{\partial}{\partial s}\left(\frac{k(T)}{k_0}\frac{\partial \theta}{\partial s}\right) = 0.$$

Defining

$$\alpha(\theta) := \frac{L^2\,\rho(T)\,c(T)}{t_0\,k_0} \quad \text{and} \quad \kappa(\theta) := \frac{k(T)}{k_0},$$

the non-dimensional energy conservation equation becomes

$$\frac{\partial}{\partial \tau}\Big(\alpha(\theta)\,\theta\Big) - \frac{\partial}{\partial s}\left(\kappa(\theta)\frac{\partial \theta}{\partial s}\right) = 0 \qquad s,\,\tau \in [0,1]. \tag{7.4}$$

7.1 Variational physics-informed neural network approach

In this section we detail the variational formulation of the heat equation in its non-dimensional form. In previous sections we introduced a non-dimensional formulation to improve numerical conditioning, where small temperature variations are difficult to resolve. Here, we build on that formulation to derive the weak form and associated loss function for a roboust variational physics-informed neural network (RVPINN) approach.

Consider the following heat problem:

$$\frac{\partial}{\partial \tau}\Big(\alpha(\theta)\,\theta\Big) - \frac{\partial}{\partial s}\left(\kappa(\theta)\frac{\partial \theta}{\partial s}\right) = 0, \qquad s \in \Omega_s = (0,1) \text{ and } \tau \in (0,1), \tag{7.5}$$

$$\theta(0,\tau) = \theta(1,\tau) = \theta_{\text{bound}}, \qquad \tau \geq 0 \tag{7.6}$$

$$\theta(s,0) = \theta_{\text{init}}, \qquad s \in [0,1]. \tag{7.7}$$

where θ is the non-dimensional temperature, s is the non-dimensional spatial parameter, τ is the non-dimensional time parameter, for characteristic length $L_{\text{ref}} = 0.5715$ meters, initial temperature $T_{\text{ref}} = 20$ degress celcius and $\text{time}_{\text{ref}} = 2$ hours.

Time discretization

We discretize the (dimensionless) time interval $[0,1]$ by setting $\Delta\tau = 1/N_{\text{time}}$ for a given $N_{\text{time}} \in \mathbb{N}$, and denote $\tau^n := n\,\Delta\tau$.

To approximate the time derivative, we employ a backward finite-difference scheme as follows:

$$\frac{\partial}{\partial \tau}(\alpha(\theta)\,\theta)\bigg|_{\tau^n} \approx \frac{\alpha\big(\theta^n\big)\,\theta^n \;-\; \alpha\big(\theta^{n-1}\big)\,\theta^{n-1}}{\Delta\tau}, \tag{7.8}$$

and to simplify the discretization we take

$$\alpha(\theta^n) \approx \alpha(\theta^{n-1}).$$

That is, we approximate

$$\frac{\partial}{\partial \tau}(\alpha(\theta)\,\theta)\bigg|_{\tau^n} \approx \alpha\big(\theta^{n-1}\big)\,\frac{\theta^n - \theta^{n-1}}{\Delta\tau}. \tag{7.9}$$

Replacing the time discretization on the energy conservation equation yields:

$$\alpha\big(\theta^{n-1}\big)\,\frac{\theta^n - \theta^{n-1}}{\Delta\tau} \;-\; \frac{\partial}{\partial s}\Big(\kappa\big(\theta^n\big)\,\frac{\partial \theta^n}{\partial s}\Big) \;=\; 0, \tag{7.10}$$

which can be simplified to:

$$\alpha\big(\theta^{n-1}\big)\,\theta^n \;-\; \alpha\big(\theta^{n-1}\big)\,\theta^{n-1} \;=\; \Delta\tau\,\kappa\big(\theta^{n-1}\big)\,\frac{d^2\theta^n}{ds^2}. \tag{7.11}$$

Variational formulation

Multiplying (7.11) by a test function $v \in H_0^1(\Omega_s)$ and integrating over the spatial domain Ω_s, we obtain

$$\int_{\Omega_s} \alpha(\theta^{n-1})\,\theta^n(s)\,v(s)\,ds - \Delta\tau \int_{\Omega_s} \kappa(\theta^{n-1})\,\frac{d^2\theta^n(s)}{ds^2}\,v(s)\,ds = \int_{\Omega_s} \alpha(\theta^{n-1})\,\theta^{n-1}(s)\,v(s)\,ds. \tag{7.12}$$

Since the test functions belong to $H_0^1(\Omega_s)$, integration by parts (with zero boundary contributions) allows us to rewrite the second term as

$$-\int_{\Omega_s} \kappa(\theta^{n-1})\,\frac{d^2\theta^n}{ds^2}\,v\,ds = \int_{\Omega_s} \kappa(\theta^{n-1})\,\frac{d\theta^n}{ds}\,\frac{dv}{ds}\,ds.$$

Thus, the weak formulation becomes

$$\int_{\Omega_s} \alpha(\theta^{n-1})\,\theta^n\,v\,ds + \Delta\tau \int_{\Omega_s} \kappa(\theta^{n-1})\,\frac{d\theta^n}{ds}\,\frac{dv}{ds}\,ds = \int_{\Omega_s} \alpha(\theta^{n-1})\,\theta^{n-1}\,v\,ds. \tag{7.13}$$

for all $v \in H_0^1(\Omega_s)$

Following the procedure described in Section 3.5.1, we approximate the weak residual by expanding it on an orthonormal basis $\{\varphi_k\}_{k=1}^{N_{\text{test}}}$ of $H_0^1(\Omega_s)$:

$$\mathcal{R}_n^w(\theta,\varphi_k) := \int_{\Omega_s} \alpha(\theta^{n-1})\,\theta^n\,\varphi_k\,ds + \Delta\tau \int_{\Omega_s} \kappa(\theta^{n-1})\,\frac{d\theta^n}{ds}\,\frac{d\varphi_k}{ds}\,ds - \int_{\Omega_s} \alpha(\theta^{n-1})\,\theta^{n-1}\,\varphi_k\,ds.$$

Thus, the overall loss function, defined as the time-discrete sum of the squared dual residual norms, is given by

$$\begin{aligned}\mathcal{L}(\theta) = \Delta\tau \sum_{n=1}^{N_{\text{time}}} \sum_{k=1}^{N_{\text{test}}} \Bigg(\sum_{i=1}^{N_S} w_i \Big(&\alpha(\theta^{n-1}(s_i))\,\theta^n(s_i)\,\varphi_k(s_i) \\ &+ \kappa(\theta^{n-1}(s_i))\,\frac{d\theta^n}{ds}(s_i)\,\frac{d\varphi_k}{ds}(s_i) \\ &- \alpha(\theta^{n-1}(s_i))\,\theta^{n-1}(s_i)\,\varphi_k(s_i)\Big)\Bigg)^2.\end{aligned} \tag{7.14}$$

where $\{s_i\}_{i=1}^{N_S}$ and weights $\{w_i\}_{i=1}^{N_S}$ are obtained via an appropriate quadrature rule (e.g., the midpoint rule) over the non-dimensional spatial domain.

Equation (7.14) represents the discretized loss function for the non-dimensional variational formulation of the heat equation. Minimizing this loss with respect to the neural network parameters will enforce that the approximate solution satisfies the weak form of the non-dimensional heat equation across all time steps.

7.2 Numerical experiment

In this experiment, we implement a feed-forward, fully-connected neural network in *TensorFlow 12.8*, consisting of four hidden layers with ten neurons each. We employ the hyperbolic tangent (tanh) activation function throughout these hidden layers. The output layer is designed with one neuron per time step plus an additional neuron for the initial condition, whose output is then passed through a cutoff layer to enforce the boundary condition. We optimize the network parameters using the Adam optimizer with a learning rate of $10^{-2.5}$.

A total of $N_{\text{time}} = 1000$ time steps are taken, each of size $\Delta t = 0.001$. Additionally, we incorporate measured data for both the thermophysical properties and the temperature measurements at the cylinder's core, as well as the freezer's external temperature. Under the chosen non-dimensional setting, the domain length is $L = 0.5715$. To prescribe boundary conditions, we perform a spline interpolation between the core and external temperature data. Training proceeds for up to 20,000 epochs, but an early-stop mechanism halts the process if the loss does not decrease by at least 0.1 over 1000 consecutive epochs, which ended after near 16000 epochs.

To form the residual-based loss function, we project onto 20 test functions, i.e., the first 20 Fourier modes, at each time step. Spatial integration is carried out over 150 randomly distributed points using a midpoint quadrature rule. This balances computational efficiency with sufficient integration accuracy for the loss evaluation.

Figure 7.3 illustrates the evolution of the solution over five time steps. The experimental Dirichlet boundary conditions are fully satisfied at all time steps, and the temperature evolution at the midpoint closely aligns with the expected values from the experiments which can be observed in **??**, along with the experimental and simulated temperature for the walls of the cylinder.

Figure 7.4 presents the evolution of the loss function and the relative error during training, both displayed on a log-log scale. In Figure 7.4c we show the evolution of the relative error in which we compare the temperature given by the NN model and the experimental temperature both measured at the midpoint of the tank. The error decreases consistently, demonstrating the convergence of the model towards the expected solution, achieving good accuracy.

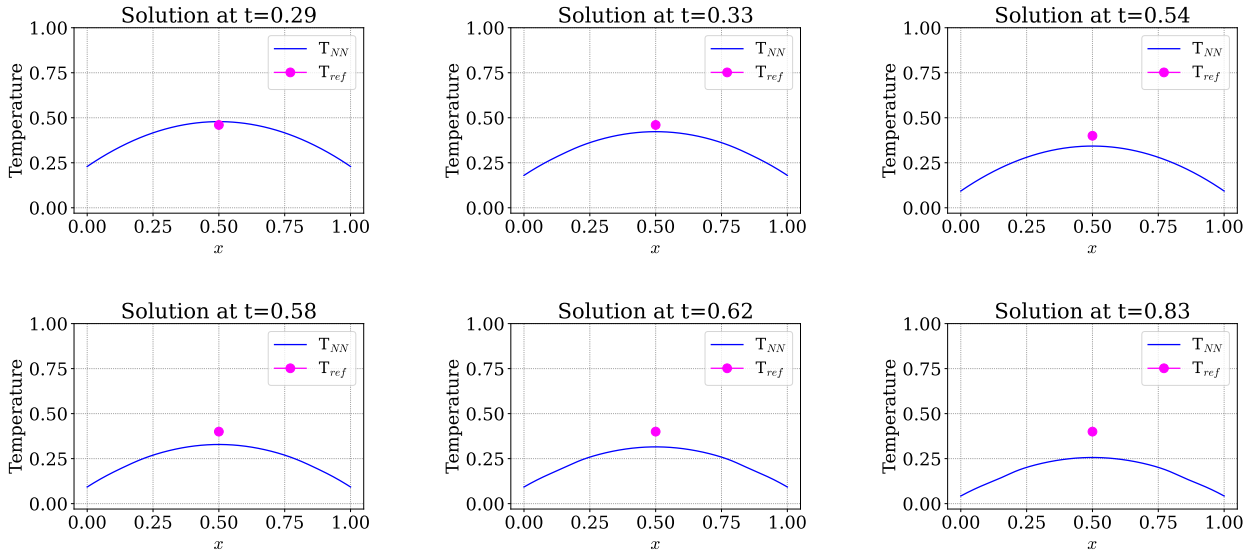
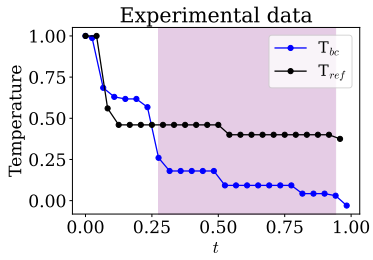
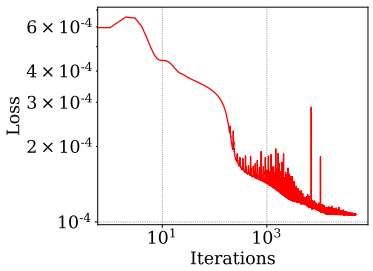


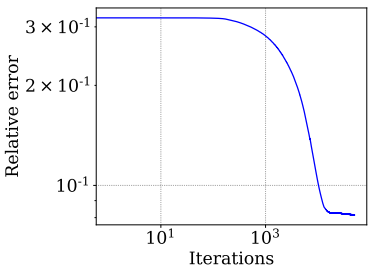
Figure 7.3: Temperature profile of the central horizontal cut of the tank at different time steps.



(a) Measured central temperature and estimated boundary temperature.



(b) Evolution of the loss function (log-log scale)



(c) Evolution of the relative error (log-log scale)

Figure 7.4: Experimental data and the evolution of the loss and the relative error (measured in the middle point of the tank) during training.

Chapter 8

Conclusions

We developed a comprehensive mathematical framework that integrates variational formulations and neural network approximations to solve parabolic partial differential equations; this approach makes a bridge between rigorous numerical analysis grounded in the Hilbert spaces theory and classical finite elements methods, with modern deep learning techniques, specifically physics-informed neural networks and their extension into robust variational PINNs.

This was reached trough a theoretical foundation in the analysis of variational formulations, including the construction of Hilbert bases in Sobolev spaces and error estimates based on the dual norm of such spaces. Building on this formulation we employed the weak formulation for both elliptic and parabolic problems, which then were coupled with neural network architectures. By leveraging both robust variational PINNs (RVPINNs) and the Deep Fourier Residual Method (DFRM), this method improve stability and convergence by minimizing the dual norm of the residual for parabolic problems. Numerical experiments, including the solution of the heat equation and an industrial-scale freezing problem, demonstrated that the method accurately approximate the true solution while ensuring that boundary conditions are met even when the problem includes non-linear parameters and phase changes vary the thermophysical properties of the model.

The developed method provides a mathematically rigorous pathway for incorporating physical laws directly into the training of neural networks. This is particularly significant for complex industrial processes where traditional numerical methods may struggle with high dimensionality or irregular data. By combining variational principles with deep learning, this approach enhances model stability, enables rigorous error control, and opens new paths for solving high-dimensional or non-linear PDEs. In contrast with classical methods such as FEM, which require mesh generation and rely on local approximations and traditional quadrature rules, the robust variational framework eliminates the need for mesh generation, a significant advantage when dealing with complex geometries and high-dimensional problems. Moreover, while FEM benefits from well-established error estimates and convergence properties, the RVPINN approach leverages gradient descent and automatic differentiation to perform a global approximation, offering greater flexibility and adaptability, especially

when experimental data are sparse or noisy, albeit at the cost of carefully selecting test functions and integration schemes to ensure stability.

Despite the promising results, several limitations remain. The performance of the robust variational formulation is sensitive to the selection of test functions and the employed numerical integration schemes. In some cases, the inversion of the Gram matrix—required for computing the Riesz representative, introduces additional computational costs. Although a careful choice of basis functions for the underlying Hilbert space can obviate the need for matrix inversion, such an approach demands careful decision-making in the construction and selection of an appropriate basis. Moreover, since the neural network optimization relies on gradient descent methods, the training process may experience oscillations or slow convergence in certain configurations.

Future research should explore strategies to mitigate these limitation. Potential directions include developing adaptative test functions strategies and exploring alternative bases that further enhance stability. Investigating more sophisticated quadrature rules or adaptative integration techniques to reduce numerical errors. Experimenting with advanced optimization algorithms to improve convergence behavior and extending the framework to cover more complex and multi-dimensional industrial applications.

Appendix A

Appendix A

A.1 Neural Networks

Humans and computers shine in different areas, while computers can effortlessly solve complex mathematical calculations, humans are excellent interpreting language riddles and recognizing objects inside an image. Recent advances in deep learning have allowed machines to surpass humans in tasks that were unimaginable years ago.

Neural networks are a fundamental component of modern artificial intelligence, modeled after the human brain's structure and function. They consist of interconnected nodes, or neurons, which process and transmit information through layers. These networks are capable of learning complex patterns and making predictions based on large datasets. The architecture and training methods of neural networks have been continuously refined, leading to significant advancements in fields such as image recognition, natural language processing, and autonomous systems.

The evolution of computing hardware has been crucial to the advancement of neural networks. Specialized processors, such as Graphics Processing Units (GPUs), have significantly reduced the time required to train complex models. This increase in computational power has allowed researchers to experiment with and improve neural network architectures more effectively.

To understand the structure of a multi-layered neural network, we must first consider the structure of a single neuron. Neurons and neural networks are inspired by their biological counterparts. Conceptually, a neuron, whether biological or artificial, can be seen as a computational unit that receives a series of inputs and determines whether to produce an output. If triggered, it sends an output signal to one or more other neurons. These interconnected neurons work together to process information that is too complex for a single neuron to handle.

A.1.1 The perceptron

The simplest neural network is known as the perceptron. This neural network contains a single input layer and an output node. The architecture of a single neuron is shown in Figure A.1. The training data has m inputs denoted by the vector $\vec{x}$ and a single output y. The input layer contains m nodes that transmit the n features $x_1, \ldots, x_m$ in the vector $\vec{x}$. The input layer connects to the output node through edges with weights $w_1, \ldots, w_m$ in the vector $\vec{w}$.

An artificial neuron $\mathcal{N}$ can be modeled as a mapping from an m-dimensional input feature vector with a bias term $b \in \mathbb{R}$ to a real number y, $\mathcal{N} : \mathbb{R}^{n+1} \to \mathbb{R}$. This is typically done by multiplying each value in the input feature vector with a corresponding weight w_i for $i = 1, \ldots, m$, summing these up, and then applying an activation function σ to produce the result. This can be expressed as:

$$y = \sigma\left(bw_0 + \sum_{i=1}^{m} w_i x_i\right) \tag{A.1}$$

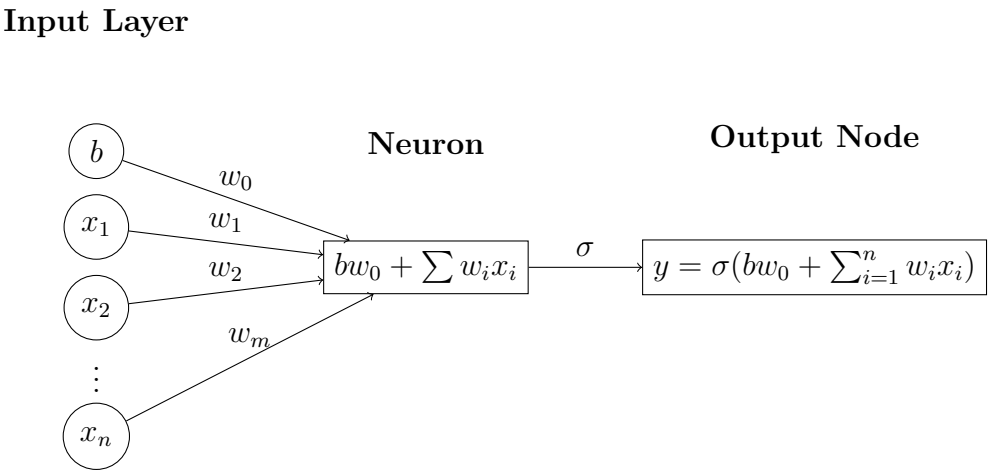


Figure A.1: Illustration of a single neuron model with input, weights, bias, and activation function.

Neural networks operate by computing mappings from input nodes to output nodes. Each node contains a variable that either results from a function computation or is fixed externally (for input nodes). Structurally, neural networks are directed acyclic graphs with edges parameterized by weights. The computed functions at each node are influenced by these weights and the variables of the nodes at the tails of these edges. The overall function computed by a network results from cascading function computations at individual nodes. By appropriately setting the edge weights, almost any function can be computed from the input to the output.

In a single-layer network, inputs are directly connected to output nodes, which compute a function based on their inputs and the edge weights. In contrast, multi-layer networks have

neurons arranged in layers, with hidden layers separating the input and output layers, as shown in Figure A.2.

The primary objective of a neural network is to learn a function that maps inputs to outputs using training examples. The network constructs this function by adjusting the edge weights to ensure that the input-to-output computations match the observed data. This data-driven adjustment of weights is essential to all neural network learning processes.

For simplicity, consider the case where there are m inputs and a single binary output drawn from $\{-1, 1\}$. Each training instance is of the form $(\vec{x}, y)$, where $\vec{x} = (x_1, \ldots, x_m)$ contains m feature variables, and $y \in \{-1, 1\}$ contains the observed value of the binary class variable. The "observed value" is given as part of the training data, and our goal is to predict the class variable for cases where it is not observed. The variable y is referred to as the **target variable**, representing the key properties the machine learning algorithm aims to predict. In other words, we want to learn the function $f(\cdot)$, where:

$$y = f_{\vec{w}}(\vec{x}) \tag{A.2}$$

Here, $\vec{x}$ corresponds to the feature vector, and $\vec{w}$ is the weight vector that must be computed in a data-driven manner to construct the predicted function. The function $f(\vec{x})$ is subscripted with a weight vector to indicate its dependence on $\vec{w}$. Most machine learning problems reduce to the following optimization problem in one form or another:

$$\text{Minimize over } \vec{w} \text{ the mismatch between } y \text{ and } f_{\vec{w}}(\vec{x})$$

This mismatch between the predicted and observed values is penalized using a **loss function**

The input layer does not perform any computation by itself. The linear function $\vec{x} \cdot \vec{w} = \sum w_i x_i$ is computed at the output node. The sign of this real value is then used to predict the dependent variable $\vec{x}$. Therefore, the prediction $\hat{y}$ is computed as follows:

$$\hat{y} = f(\vec{x}) = \text{sign}\vec{x} \cdot \vec{w} = \text{sign}\sum w_i x_i \tag{A.3}$$

The sign function maps a real value to either $+1$ or -1, which is suitable for binary classification. The hat on top of the variable y indicates that it is a predicted value rather than the original observed value y. The neural network creates a basic form of the function $f(\cdot)$ to be learned, although the weights $\vec{w}$ are unknown and need to be inferred in a data-driven manner from the training data. In most cases, a **loss function** is used to quantify a non-zero cost when the observed value y of the class variable differs from the predicted value $\hat{y}$. The weights of the neural network are learned to minimize the aggregate loss over all training instances.

Initially, the weights in the column vector $\vec{w}$ are unknown and may be set randomly. As a result, the predicted output $\hat{y}$ for a given input $\vec{x}$ will also be random and may not match the observed value y. The goal of the learning process is to use these errors to adjust the weights so that the neural network's predictions become more accurate.

A neural network begins with a basic form of the function $f(\vec{x})$, allowing for some ambiguities in the parametrization. By learning the parameters, also referred to as the weights,

the network effectively learns the function it computes. This principle is common in machine learning, where the basic structure of a function relating two or more variables is defined, but some parameters are left unspecified. These parameters are then learned in a data-driven manner to ensure that the functional relationships between the variables are consistent with the observed data. This process involves formulating an optimization problem.

In this example, the sign function serves as an **activation function**, and we will see many other examples of this type of function later in this chapter. The value of a neural network in a node is sometimes referred to as an **activation.**

The **training** process involves feeding each input data instance $\vec{x}$ into the network one by one or in small batches to create the prediction $\hat{y}$. When this prediction differs from the observed class $y \in \{-1, 1\}$, the weights are updated using the error value $y - \hat{y}$ to reduce the likelihood of this error recurring. Specifically, when the data point $\vec{x}$ is fed into the network, the weight vector $\vec{w}$ is updated as follows:

$$\vec{w} \leftarrow \vec{w} + \alpha(y - \hat{y})\vec{x} \tag{A.4}$$

The heuristic of this update is that it always increases the dot product of the weight vector and $\vec{x}$ by a quantity proportional to the error $(y - \hat{y})$, which tends to improve the prediction for that training instance. The parameter α is the **learning rate** of the neural network, although it can be set to 1 in the special case of the perceptron. The perceptron algorithm repeatedly cycles through all the training examples in random order and iteratively adjusts the weights until convergence is reached. A single training data point may be cycled through many times. Each cycle is referred to as an **epoch**.

In many cases, there is an invariant part of the prediction referred to as **the bias**. For example, consider a situation where the feature variables are mean-centered, but the mean of the binary class prediction from $\{-1, 1\}$ is not 0. This tends to occur when the binary class distribution is highly unbalanced. In such cases, the mentioned approach is insufficient for prediction since the different training points sum to zero, whereas the class variables do not. Adding up all the predictions over the training points results in a significant discrepancy between the observed data and the predicted data. When the differences are large, this model will perform poorly regardless of the training data.

We need to incorporate an additional bias variable b that captures the invariant part of the prediction:

$$\hat{y} = \text{sign}\vec{w} \cdot \vec{x} + b = \text{sign}\sum w_i x_i + b \tag{A.5}$$

This bias can be incorporated as the weight of an edge by using a **bias neuron**, as shown in Figure A.1. This is achieved by adding a neuron that always transmits the value 1 to the output node.

Most machine learning algorithms, including neural networks, can be framed as loss optimization problems, where **gradient descent** updates minimize the loss. The key idea is to adjust the network's weights to reduce the difference between the predicted and observed values. This is done by calculating the gradient of the loss function with respect to the weights and updating the weights in the direction that reduces the loss. This process ensures that the network learns from the training data and improves its predictions over time.

Consider a training instance $(\vec{x}_i, y_i)$, where $\vec{x}_i$ is a row vector containing the features and y_i is the observed class variable. The perceptron criterion is a loss function that penalizes the training instance when the sign of $\vec{w} \cdot \vec{x}_i$ differs from y_i:

$$L_i = \max\{-y_i(\vec{w} \cdot \vec{x}_i), 0\} \tag{A.6}$$

Additionally, large absolute values of $\vec{w} \cdot \vec{x}_i$ that do not match the sign of y_i are penalized to a greater degree. Gradient descent updates $\vec{w}$ in the negative direction of the gradient of L_i (with respect to $\vec{w}$). The gradient of the loss is:

$$\frac{\partial L_i}{\partial \vec{w}} = \left[\frac{\partial L_i}{\partial w_1}, \ldots, \frac{\partial L_i}{\partial w_m}\right] = \begin{cases} -y_i \vec{x}_i & \text{if sign}(\vec{w} \cdot \vec{x}_i) \neq y_i \\ 0 & \text{otherwise} \end{cases} \tag{A.7}$$

The matrix calculus notation $\frac{\partial L_i}{\partial \vec{w}}$ represents the derivative of a scalar with respect to a vector and is an m-dimensional vector. The negative of this vector indicates the direction with the fastest rate of loss reduction, which is why the perceptron update is:

$$\vec{w} \leftarrow \vec{w} - \alpha \frac{\partial L_i}{\partial \vec{w}} = \vec{w} + \alpha y_i \vec{x}_i \quad \text{[For misclassified instances]}$$

The perceptron loss function has some weaknesses; for instance, setting $\vec{w}$ to the zero vector results in an optimal loss value of 0, regardless of the training dataset. However, perceptron updates converge to a meaningful solution in linearly separable training datasets with non-zero initial weights. For linearly separable datasets, there exists a non-zero weight vector $\vec{w}$ such that the sign of $\vec{w} \cdot \vec{x}_i$ matches y_i for every training instance $(\vec{x}_i, y_i)$. Conversely, the perceptron algorithm can behave unpredictably with non-linearly separable data, sometimes yielding poor approximations of class separation.

A.1.2 Multilayer Neural Networks

By understanding how to describe a single neuron, we can combine these into what is called a **single layer** to compute the output. The input is referred to as the **input layer** and consists of the data fed to the neural network. Each input is multiplied by a specific weight corresponding to the connections to the neurons in the **hidden layer**, and a specific bias term is added to the input going to the neuron in the hidden layer (see Fig. A.2).At each neuron, the inputs and bias terms are summed up, and an **activation function** is applied to the sum to produce the input for the next layer. If the next layer is the **output layer**, the inputs to each neuron in the output layer are summed, and depending on the desired type of output, a function might need to be applied.

Multilayer neural networks consist of more than one computational layer. A perceptron includes both an input and an output layer, with the output layer being the only computational layer. The input layer transmits data to the output layer, where all computations are visible to the user. In contrast, when multiple hidden layers are present, the computations

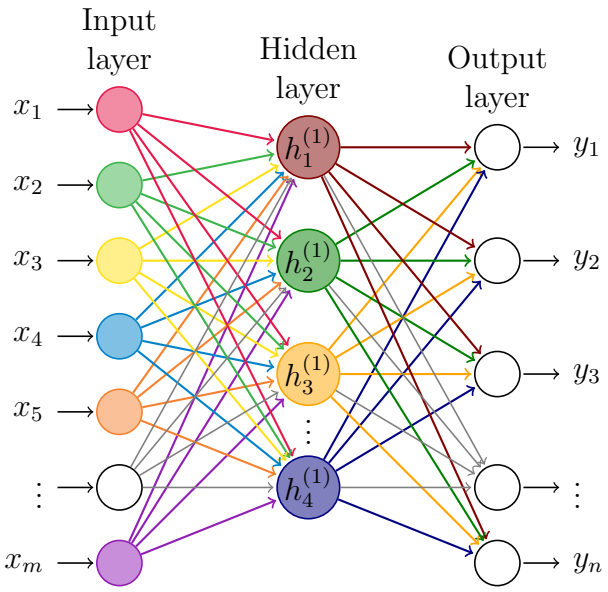


Figure A.2: Architecture of a single hidden layer network

are not directly visible, creating a complex internal structure. This architecture is known as **feed-forward** networks, where each layer feeds into the next in a forward direction from input to output. The default architecture of feed-forward networks assumes full connectivity between nodes in adjacent layers. Therefore, the neural network's architecture is primarily defined by the number of layers and the number and type of nodes in each layer. Note that the input layer is typically not counted as a computational layer because it simply transmits data without performing any computations.

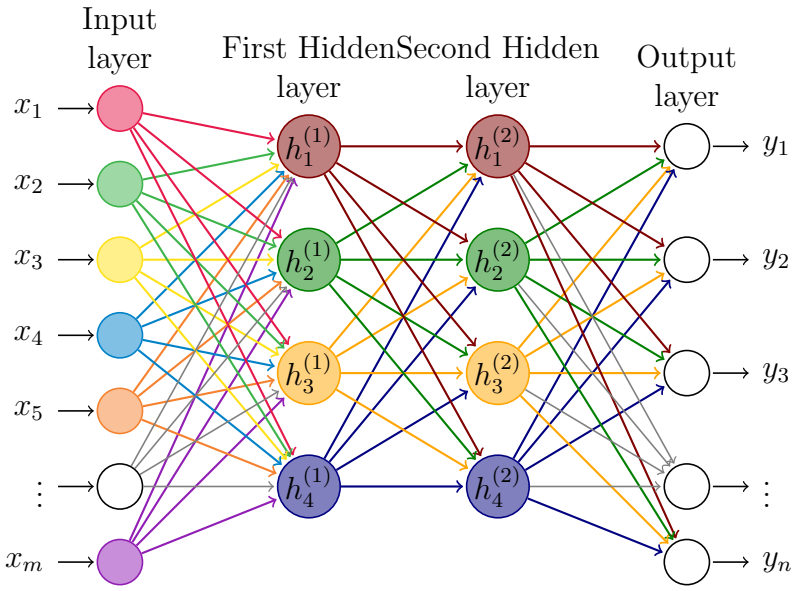


Figure A.3: Multi-Layer NN with two hidden layers

If a neural network contains k layers, in which each layer has $p_1, \ldots, p_k$ units in each layer, then the vector representation of these outputs, denoted by $h_1, \ldots, h_k$ has dimension $p_1, \ldots, p_k$. Therefore the number of units in each layer is referred as the **dimensionality** of that layer. The weights of the connections between the input layer and the first hidden layer are contained in a matrix W_1 with size $p_1 \times m$ given that the input has m features, meanwhile the weights between the r-th hidden layer and the $(r+1)$-th hidden layer are denoted by the $p_{r+1} \times p_r$ matrix W_r. Note that the number of units in the $(r+1)$-th layer defines the number of rows, whereas the number of units in the r-th layer defines the number of columns. If the output layer contains n nodes, then the final matrix W_{k+1} of a $(k+1)$-layered neural network is of size $n \times p_k$. In each step we can consider the addition of the respective bias b_k in each hidden layer as we did with the perceptron.

For example for the 2-layer neural network presented in Fig. A.2, we should have a transformation given by:

$$\vec{y} = W_2\, \sigma(W_1\, \vec{x} + b_1) + b_2 \tag{A.8}$$

In general, considering the bias as a neuron with a constant weight as seen in the perceptron, the transformation is given by:

$$NN = \begin{cases} h_1 = \sigma(W_1\, \vec{x}) & \text{Input to the hidden layer} \\ h_{p+1} = \sigma(W_{p+1}\, h_p) & \text{Hidden to hidden layer} \\ \vec{y} = \sigma(W_{k+1}\, h_k) & \text{Hidden to output layer} \end{cases} \tag{A.9}$$

The length of a neural network, defined by the number of hidden layers plus the output layer, is referred to as the **depth** of the network. As the number of hidden layers increases, the resulting function can become increasingly complex, enabling the network to approximate more complex functions. However, the **Universal Function Approximation Theorem**, proposed by Pinkus in 1999 [32, 16, 21], states that a single layer is sufficient to approximate any continuous function. The limitation of this theorem is that it does not specify the number of neurons required to approximate the function, and the number of neurons necessary for a single hidden layer neural network to approximate a function may not be practical.

Eldan and Shamir demonstrated that there exists a function defined on $\mathbb{R}^d$ that can be expressed by a neural network with only 3 hidden layers, but is very difficult to approximate with a 2-layer neural network unless the number of neurons increases exponentially for any desired level of accuracy. This illustrates that as the number of layers increases, the representation power of the neural network also increases. However, there is a trade-off between the depth of the network and the number of neurons required to achieve a certain level of accuracy.

Activation Functions

Activation functions, denoted as σ, determine the output value of a neuron and condition whether the neuron is "activated" and outputs a non-zero value, hence the name. The choice of activation function is a critical part of neural network design. These introduce non-linearity, which is essential for the network's ability to learn and model complex patterns.

Without this non-linearity, all neural networks would essentially function as linear regression models, regardless of the number of layers.

Activation Function	**Formula**	**Derivative**
Sign Function	$\sigma(v) = \text{sign}(v)$	Not differentiable at $v = 0$
Sigmoid Function	$\sigma(v) = \frac{1}{1+e^{-v}}$	$\sigma'(v) = \sigma(v)(1 - \sigma(v))$
Tanh Function	$\sigma(v) = \tanh(v) = \frac{e^{2v}-1}{e^{2v}+1}$	$\sigma'(v) = 1 - \tanh^2(v)$
ReLU	$\sigma(v) = \max\{v, 0\}$	$\sigma'(v) = \begin{cases} 1 & v > 0 \\ 0 & v \leq 0 \end{cases}$
Leaky ReLU	$\sigma(v) = \begin{cases} v & v > 0 \\ \alpha v & v \leq 0 \end{cases}$	$\sigma'(v) = \begin{cases} 1 & v > 0 \\ \alpha & v \leq 0 \end{cases}$
ELU	$\sigma(v) = \begin{cases} v & v > 0 \\ \alpha(e^v - 1) & v \leq 0 \end{cases}$	$\sigma'(v) = \begin{cases} 1 & v > 0 \\ \alpha e^v & v \leq 0 \end{cases}$
Polynomial ReLU	$\sigma(v) = \begin{cases} v^{n+1} & v > 0 \\ 0 & v \leq 0 \end{cases}$	$\sigma'(v) = \begin{cases} (n+1)v^n & v > 0 \\ 0 & v \leq 0 \end{cases}$
Swish	$\sigma(v) = v \cdot \frac{1}{1+e^{-v}}$	$\sigma'(v) = \sigma(v) + \sigma(v) \cdot (1 - \sigma(v)) \cdot v$
Softmax	$\sigma(\vec{v})_i = \frac{\exp(v_i)}{\sum_{j=1}^{k} \exp(v_j)}$	$\sigma'(\vec{v})_i = \sigma(\vec{v})_i(1 - \sigma(\vec{v})_i)$

Table A.1: Common activation functions and their derivatives

The choice of activation function significantly impacts the neural network's performance and suitability for a given task. For instance, while the sigmoid and tanh functions are suitable for tasks requiring smooth and bounded outputs, ReLU and its variants are preferred for deep networks due to their computational efficiency and ability to mitigate gradient-related issues.

In multi-layer neural networks, a common approach is to use non-linear activation functions like ReLU and tanh in hidden layers to introduce non-linearity while using a linear activation function in the output layer when the target is a real value.

Figure A.4 shows the shape of different activation functions.

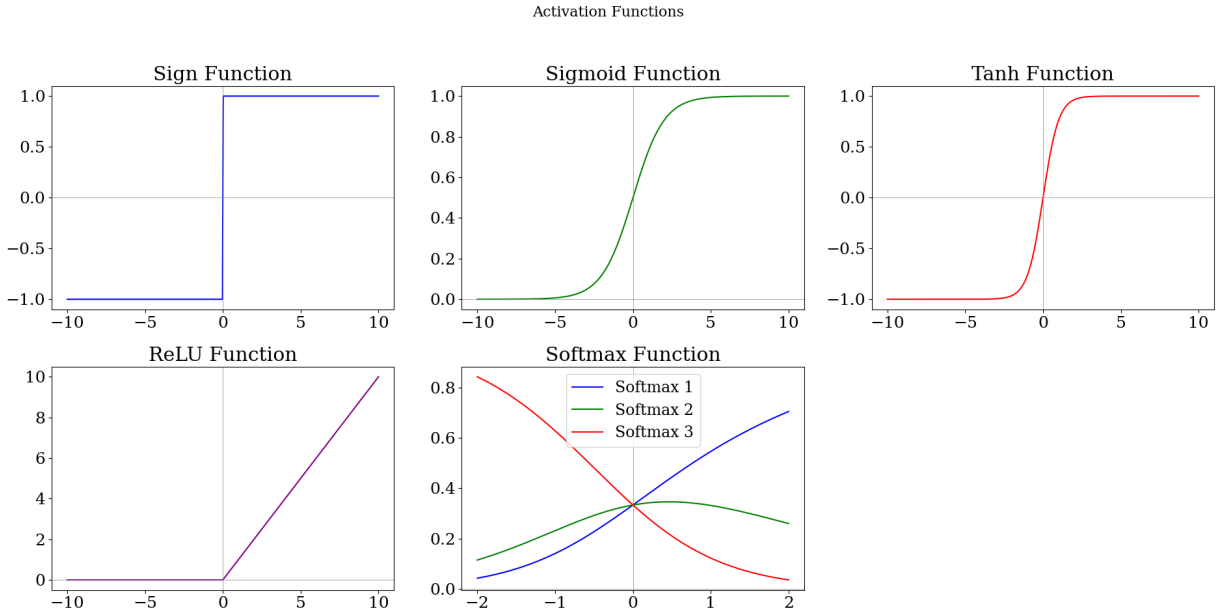


Figure A.4: Various activation functions

Loss Functions

Loss functions, also known as cost or objective functions, play a key role in training neural networks. They measure the gap between the model's predictions and the actual target values, guiding the optimization process to enhance the model's accuracy. Selecting the right loss function is essential and depends on the problem being addressed and the data involved.

A loss function quantifies the difference between the predicted output $\hat{y}$ of a model and the actual target value y. The goal of training a neural network is to minimize this difference by adjusting the model's parameters (weights and biases).

The choice of loss function depends on the nature of the task, such as regression or classification, and on the type of data used. The following table outlines some commonly used loss functions along with their mathematical definitions: Let be y_i: The actual target value for the i-th data point in the dataset, $\hat{y}_i$: The predicted value for the i-th data point, computed by the neural network, n: The total number of data points.

The choice of a loss function should also take into consideration certain properties, which impact its suitability for different tasks:

- **Convexity**: A convex loss function ensures that any local minimum is the global minimum, making optimization more straightforward using gradient-based methods.
- **Differentiability**: A loss function should be differentiable to allow the use of gradient-based optimization techniques. Smooth derivatives are critical for effective learning.

Loss Function	Formula
Mean Squared Error (MSE)	$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
Mean Absolute Error (MAE)	$\mathcal{L}_{\text{MAE}} = \frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$
Binary Cross-Entropy Loss	$\mathcal{L}_{\text{BCE}} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$
Categorical Cross-Entropy Loss	$\mathcal{L}_{\text{CCE}} = -\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij}\log(\hat{y}_{ij})$
Huber Loss	$\mathcal{L}_{\text{Huber}} = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{for } \lvert y_i - \hat{y}_i\rvert \le \delta \\ \delta\lvert y_i - \hat{y}_i\rvert - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$

Table A.2: Common loss functions and their formulas

- **Robustness**: Loss functions should handle outliers gracefully, ensuring that a few extreme values do not overly influence the model.
- **Smoothness**: Smooth loss functions have continuous gradients without abrupt changes, which helps in stable and effective optimization.
- **Sparsity**: Some loss functions encourage sparse outputs, which is particularly useful in high-dimensional data settings where only a few features are significant.
- **Multi-modality**: A multi-modal loss function has multiple minima, useful in tasks that require learning diverse representations.
- **Monotonicity**: A monotonic loss function decreases as the predicted value approaches the actual value, ensuring that the optimization process converges in the right direction.
- **Invariance**: Invariant loss functions remain unchanged under specific transformations (e.g., rotation, scaling, translation), which is valuable in tasks with such transformations in the data.

For regression tasks, Mean Squared Error (MSE) and Mean Absolute Error (MAE) are commonly used. MSE is sensitive to outliers, making it suitable when errors need to be emphasized, while MAE offers robustness in the presence of outliers. Huber loss combines the benefits of MSE and MAE, making it an ideal choice when there is a mix of typical data and outliers.

In classification tasks, binary cross-entropy is used for binary classification problems, while categorical cross-entropy is appropriate for multi-class classification. For imbalanced classes

in multi-class settings, weighting classes differently within the loss function can prevent bias towards the majority class.

While standard loss functions are widely applicable, some tasks may require specialized loss functions tailored to the specific problem at hand. For instance, custom loss functions are often crafted for Physics-Informed Neural Networks (PINNs) and Variational Physics-Informed Neural Networks (VPINNs), which we explore in this work.

Backpropagations and Gradient Descend

Backpropagation is a fundamental algorithm used for training neural networks. It efficiently computes the gradient of the loss function with respect to the weights of the network. This gradient is then used by optimization algorithms like gradient descent to update the weights and minimize the loss

Backpropagation

Backpropagation is the core algorithm for training neural networks. It involves computing the gradient of the loss function with respect to each weight in the network to update the weights and minimize the loss. The backpropagation algorithm is divided into two main passes: the forward pass and the backward pass.

The key idea behind is to apply the chain rule of calculus to compute the gradient of the loss function. This involves a forward pass, where the input data is passed through the network to compute the output, and a backward pass, where the gradient of the loss is propagated back through the network to compute the gradients of the weights.

During the forward pass, the input data is passed through the network layer by layer, and the activations are computed at each layer. The output of each layer is calculated using the weights and activation functions. For a given input $\vec{x}$, the output of the i-th layer is denoted as z_i and is computed as follows:

$$z_i = \sigma(\vec{w}_i \cdot \vec{a}_{i-1} + b_i) \tag{A.10}$$

where $\vec{w}_i$ are the weights, $\vec{a}_{i-1}$ are the activations from the previous layer, b_i is the bias, and σ is the activation function.

The backward pass involves calculating the gradients of the loss function with respect to each weight in the network. This is done by propagating the error backward through the network. The key steps in the backward pass are:

1. Compute the error at the output layer:

$$\delta_L = \nabla_a L \odot \sigma'(z_L) \tag{A.11}$$

where δ_L is the error at the output layer, $\nabla_a L$ is the gradient of the loss with respect to the

activations, $\odot$ denotes the element-wise product, and σ' is the derivative of the activation function.

2. Propagate the error backward through the network:

$$\delta_i = (\vec{w}_{i+1} \cdot \delta_{i+1}) \odot \sigma'(z_i) \tag{A.12}$$

where δ_i is the error at layer i, $\vec{w}_{i+1}$ are the weights of the next layer, and δ_{i+1} is the error at the next layer.

3. Compute the gradients of the weights and biases:

$$\frac{\partial L}{\partial \vec{w}_i} = \delta_i \cdot \vec{a}_{i-1}^T \tag{A.13}$$

$$\frac{\partial L}{\partial b_i} = \delta_i \tag{A.14}$$

4. Update the weights and biases using gradient descent:

$$\vec{w}_i \leftarrow \vec{w}_i - \alpha \frac{\partial L}{\partial \vec{w}_i} \tag{A.15}$$

$$b_i \leftarrow b_i - \alpha \frac{\partial L}{\partial b_i} \tag{A.16}$$

where α is the learning rate.

Gradient Descend

Gradient descent is an optimization algorithm used to minimize the loss function by iteratively updating the weights in the opposite direction of the gradient. The update rule for the weights is given by:

1. **Stochastic Gradient Descent (SGD):** In SGD, the weights are updated for each training example individually, which can lead to faster convergence but more noise in the updates. The update rule is:

$$\vec{w} \leftarrow \vec{w} - \alpha \nabla_{\vec{w}} L(\vec{w}; \vec{x}_i, y_i) \tag{A.17}$$

2. **Mini-Batch Gradient Descent:** In mini-batch gradient descent, the weights are updated based on a small batch of training examples. This reduces the noise in the updates and often leads to better performance. The update rule is:

$$\vec{w} \leftarrow \vec{w} - \alpha \frac{1}{m} \sum_{i=1}^{m} \nabla_{\vec{w}} L(\vec{w}; \vec{x}_i, y_i) \tag{A.18}$$

where m is the batch size.

Learning Rate Decay: Learning rate decay is a technique used to adjust the learning rate over time to ensure more stable and efficient training of neural networks. The learning rate controls the size of the steps taken during the gradient descent optimization process. A high learning rate can lead to rapid progress initially but might cause the model to overshoot the optimal solution. Conversely, a low learning rate might make the training process very slow and get stuck in local minima.

To balance this, learning rate decay gradually reduces the learning rate as training progresses. This allows the model to make large steps initially when it is far from the optimal solution and smaller, more precise steps as it approaches the optimum. Here are two common methods for implementing learning rate decay:

1. Exponential decay:

$$\alpha_t = \alpha_0 \exp(-kt) \tag{A.19}$$

where α_0 is the initial learning rate, k is the decay rate, and t is the epoch number.

2. Inverse decay:

$$\alpha_t = \frac{\alpha_0}{1 + kt} \tag{A.20}$$

Regularization: Regularization techniques are employed in neural networks to prevent overfitting, which occurs when the model learns the training data too well, including its noise and outliers, thus performing poorly on unseen data. Regularization introduces a penalty to the loss function for having excessively large weights, thereby encouraging the model to learn simpler, more general patterns.

One common form of regularization is L2 regularization, also known as weight decay. In L2 regularization, a term proportional to the sum of the squared weights is added to the loss function. This penalty term discourages the weights from becoming too large by imposing a cost for having high weight values. The modified loss function with L2 regularization is:

$$\vec{w} \leftarrow \vec{w} - \alpha(\nabla_{\vec{w}} L + \lambda \vec{w}) \tag{A.21}$$

where λ is the regularization parameter.

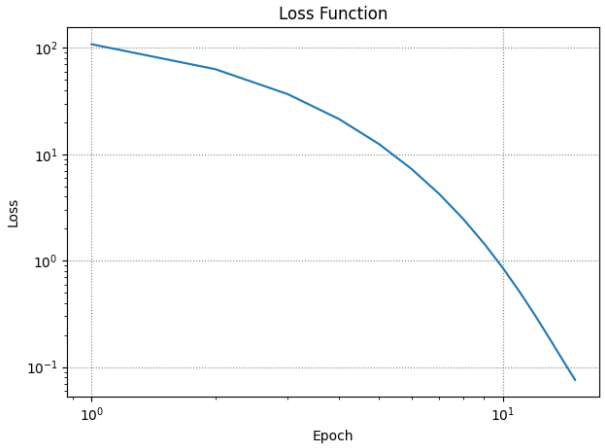
A.1.3 Example

A.2 Appendix B

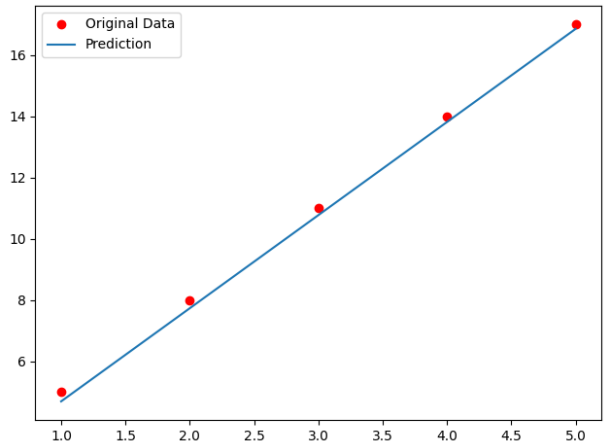
A.2.1 Finite Element Approximation

We consider a variational abstract problem as outlined in $\mathcal{P}$. Specifically, let V be a Hilbert space, $a(\cdot, \cdot)$ a continuous and coercive bilinear form defined on $V \times V$, and $l(\cdot)$ a continuous linear form defined on V. The variational formulation of the problem is as follows:

$$\mathcal{P}\text{: } \textit{Find } u \in V \textit{ such that } a(u, v) = l(v), \textit{ for all } v \in V.$$



(a) Evolution of the loss function over the epochs



(b) Approximation of the neural network to the original data

Figure A.5: Comparison of the loss function evolution and network approximation

Internal Approximation of a Variational Problem

The **internal approximation** of the problem $\mathcal{P}$ involves replacing the infinite-dimensional Hilbert space V with a finite-dimensional subspace, denoted V_h, where we seek the solution u_h. The parameter $h > 0$ relates to the discretization of the domain (either in space or time) and is intended to vanish as part of the theoretical analysis to ensure convergence.

We assume that for every $v \in V$, there exists an element $r_h(v) \in V_h$ such that

$$\lim_{h \to 0} \|r_h(v) - v\| = 0. \tag{A.22}$$

This ensures that as the discretization parameter h approaches zero, the finite-dimensional approximation converges to the true function in the Hilbert space norm.

The bilinear form $a(\cdot, \cdot)$ and the linear form $l(\cdot)$, initially defined on $V \times V$ and V, are now restricted to $V_h \times V_h$ and V_h, respectively. Thus, the original problem $\mathcal{P}$ is approximated by the following finite-dimensional problem:

$$\mathcal{P}_h\text{: } \textit{Find } u_h \in V_h \textit{ such that } a(u_h, v_h) = l(v_h), \textit{ for all } v_h \in V_h.$$

The term **internal approximation** is used because we assume $V_h \subset V$. If $N = \dim(V_h)$, let $(\varphi_i)_{1 \leq i \leq N}$ be a basis of V_h. The solution u_h can be expressed in terms of this basis as $u_h = \sum_{i=1}^{N} u_i \varphi_i$. Substituting this expansion into the problem $\mathcal{P}_h$ leads to:

$$\sum_{j=1}^{N} u_j a(\varphi_j, \varphi_i) = l(\varphi_i), \quad 1 \leq i \leq N. \tag{A.23}$$

Introducing the **stiffness matrix** $A_h = (a_{ij}) \in \mathbb{R}^{N \times N}$ with coefficients $a_{ij} = a(\varphi_i, \varphi_j)$ for all $1 \leq i, j \leq N$, the vector of unknowns $U_h = (u_i)_{1 \leq i \leq N}$, and the load vector $F_h = (f_i)_{1 \leq i \leq N}$ with $f_i = l(\varphi_i)$, we obtain the matrix equation:

$$u_h \text{ is a solution of } \mathcal{P}_h \iff A_h U_h = F_h.$$

The coercivity of the bilinear form $a(\cdot,\cdot)$ on V implies the existence and uniqueness of the solution u_h for the problem $\mathcal{P}_h$. However, since V_h is finite-dimensional, it is sufficient to ensure that:

$$a(v_h, v_h) = 0 \implies v_h = 0, \quad \forall v_h \in V_h. \tag{A.24}$$

This property guarantees that the stiffness matrix A_h is positive definite. Hence, the finite-dimensional problem $\mathcal{P}_h$ has a unique solution, and the matrix equation $A_h U_h = F_h$ can be solved using standard numerical linear algebra techniques.

Error Estimates

It is necessary to evaluate the error that results when replacing V with a finite-dimensional subspace. Assuming that $\mathcal{P}$ and $\mathcal{P}_h$ are both well-posed, particularly that the bilinear form $a(\cdot,\cdot)$ is continuous and coercive on V, there exists a constants $M > 0$ $\alpha_h > 0$ such that:

$$\forall v_h, w_h \in V_h \quad a(v_h, w_h) \leq M\|v_h\|\|w_h\| \tag{A.25}$$

$$\forall v_h \in V_h, \quad a(v_h, v_h) \geq \alpha_h \|v_h\|_V^2. \tag{A.26}$$

Proposition 16. Under the previous assumptions, and denoting by u the solution of $\mathcal{P}$ and by u_h the solution of $\mathcal{P}_h$, we have the orthogonality identity:

$$\forall v_h \in V_h, \quad a(u - u_h, v_h) = 0. \tag{A.27}$$

Proof. Since $V_h \subset V$, we have that for all $v_h \in V_h \subset V$

$$\begin{aligned} a(u, v_h) = l(v_h) = a(u_h, v_h) \quad &\text{therefore,} \\ a(u, v_h) - a(u_h, v_h) &= 0 \\ a(u - u_h, v_h) &= 0 \end{aligned}$$

which resembles an orthogonal projection.

Indeed, if the bilinear form $a(\cdot,\cdot)$ is symmetric, continuous, and coercive on $V \times V$ with constants M and α respectively, then we can define an **energy norm** associated with it as:

$$\|u\|_a = a(u,u)^{1/2}. \tag{A.28}$$

By the continuity and coercivity of $a(\cdot,\cdot)$, this norm is equivalent to the norm of V:

$$\sqrt{\alpha}\|u\|_V \leq \|u\|_a \leq \sqrt{M}\|u\|_V, \quad \forall u \in V. \tag{A.29}$$

The approximate solution u_h is thus the orthogonal projection of the solution u onto the subspace V_h, with the "inner product" $a(\cdot,\cdot)$. □

A strong advantage of the internal variational approximation is that it provides an optimal estimate of the error between the exact solution u of the problem $\mathcal{P}$ and the approximate solution u_h of the problem $\mathcal{P}_h$. The error $\|u - u_h\|_V$ is comparable to the minimum of $\|u - v_h\|$ when v_h spans V_h. This result is due to Jean Céa (1932 - 2024) and proves a "best approximation" result.

Lemma 17 (Céa)**.** Under the previous hypothesis, if $a(\cdot, \cdot)$ is continuous and coercive, we have that

$$\|u - u_h\|_V \leq \frac{M}{\alpha}\|u - v_h\|, \quad \forall v_h \in V_h. \tag{A.30}$$

Proof. By the coercivity of $a(\cdot, \cdot)$, we get that

$$\begin{aligned}\alpha\|u - u_h\|^2 &\leq a(u - u_h, u - u_h)\\ &\leq a(u - u_h, u - v_h + v_h - u_h), \quad \forall v_h \in V_h\\ &= a(u - u_h, u - v_h) + a(u - u_h, v_h - u_h).\end{aligned}$$

Notice that both u_h and v_h belong to V_h, therefore by the a-orthogonality

$$a(u - u_h, v_h - u_h) = 0. \tag{A.31}$$

So, by continuity,

$$\alpha\|u - u_h\|^2 \leq a(u - u_h, u - v_h) \leq M\|u - u_h\|_V\|u - v_h\|, \quad \forall v_h \in V_h. \tag{A.32}$$

□

Corollary 17.1. Under the previous hypothesis, let $(V_h)_h$ be a set of finite-dimensional subspaces of V and assume that

$$\forall v \in V, \quad \inf_{v_h \in V_h} \|v - v_h\|_V \to 0 \quad \text{as} \quad h \to 0. \tag{A.33}$$

The objective for the "ideal" approximation method is to define suitable approximation spaces V_h to apply the **Galerkin** approach. To this end, we search for a relationship between the dimension N of V_h and the accuracy of the numerical solution u_h. We shall also consider spaces V_h that allow for the easy computation of the quantities $a(\varphi_j, \varphi_i)$ and $l(\varphi_i)$. Finally, specific spaces V_h may result in **sparse** matrices A_h where the number of non-zero elements is small, or **well-conditioned** matrices with a small condition number, making them easy to solve. In this spirit, the **finite element method** tends to meet all these requirements. Before detailing the main concepts of the finite element method, let us discuss the Ritz and Petrov-Galerkin methods.

Petrov-Galerkin methods

The Petrov-Galerkin approach considers two finite-dimensional approximation sub-spaces V_h and W_h in V such that

$$\dim V_h = \dim W_h = N. \tag{A.34}$$

The approximate solution u_h is searched in the space V_h but the test functions in the variational formulation are now taken in W_h. For these reasons, V_h is called the **approximation space** and W_h is the space of **test functions**. The problem to solve now is the following:

$$(\mathcal{P}_{h,PG}) \qquad \text{Find } u_h \in V_h \text{ such that } a(u_h, v_h) = l(v_h), \text{ for all } v_h \in W_h.$$

Suppose that $(\varphi_j)_{1 \leq j \leq N}$ is a basis of V_h and $(\psi_j)_{1 \leqq j \leqq N}$ is a basis of W_h, then every $u_h \in V_h$ can be decomposed as

$$u_h = \sum_{j=1}^{N} u_j \varphi_j \tag{A.35}$$

and we can rewrite the problem as follows:

$$\textit{find } u_h \in V_h, \text{ such that } \sum_{j=1}^{N} u_j a(\varphi_j, \psi_i) = l(\psi_i) \text{ for } i = 1, \ldots, N,$$

and the linear system to solve is $A_h U = F_h$, where $A_h = (a_{ij}) \in R^{N \times N}$ with $a_{ij} = a(\varphi_i, \psi_j)$, $F_h = (f_i) \in R^N$ with $(f_i) = l(\psi_i)$, for $i = 1, \ldots, N$.

Under this framework then, the Galerkin method and the Petrov-Galerkin method differ primarily in their choice of test functions. In the Galerkin method, the test functions are chosen to be the same as the basis functions used to approximate the solution. This ensures that the residual error is orthogonal to the space spanned by these trial functions. In contrast, the Petrov-Galerkin method allows for different test and trial functions. This added flexibility can be beneficial for improving numerical stability and accuracy, especially in problems where standard Galerkin methods might struggle, such as those involving convection-dominated processes.

A.2.2 The Finite Element Method

In the finite element method, the domain Ω is subdivided into a partition or a **mesh** T_h, that is, a collection of geometrically simple elements, and the approximation space V_h is composed of piece-wise polynomial functions on each element K of the partition T_h. The parameter h represents the grain of discretization, i.e., the size of the elements K in T_h as defined by:

$$h = \max_{K \in T_h} \operatorname{diam}(K) \tag{A.36}$$

Typically, a basis of V_h will be composed of functions whose support is restricted on one or a few elements of T_h and the polynomials are usually loew degree. Hence, when $h \to 0$ the space V_h will better and better approximate the space V and the stiffness matrix A will be sparse, most of its coefficients being zero.

In the context of the Finite Element Method (FEM), both the Galerkin and Petrov-Galerkin methods are used to transform a PDE into a weak form that can be discretized

and solved numerically. FEM involves discretizing the problem domain into finite elements and using basis functions to approximate the solution within each element. The Galerkin method applies directly within this framework by using the same basis functions for both the trial and test functions. On the other hand, the Petrov-Galerkin method introduces a different set of test functions, which can help to stabilize the numerical solution and improve accuracy for certain types of PDEs.

Among the advantages of the FEM there's

1. The versatility of the formulation on arbitrary complex geometries, and the possibility to locally refine the partition T_h to approximate solutions with singularities,
2. the boundary conditions are naturally taken into account in the space V in the variational formulation and in its integral approximation V_h,
3. the general framework of the variational approximation is convenient for the error analysis.

The one dimensional case

let us introduce the general principle of the Lagrange finite element method in a one dimensional space. Without loss of generality we can restrict ourselves to the domain $\Omega = (0, 1)$.

Definition 33. A **mesh** is simply a set of points $(x_j)_{1\leq j\leq N+1}$ or intervals

$$K_j = [x_j, x_{j+1}]$$

such that $0 = x_0 < x_1 \dots x_{N+1} = 1$. The mesh is said to be **uniform** if the points (x_j) are separated by the same distance on from the other along the interval $[0, 1]$, in this way we can state that the point in the mesh have the form

$$x_j = j\frac{1}{N+1}, \quad 0 \leq j \leq N+1.$$

In general the size parameter is given by $h = \max |x_{j+1} - x_j|$

Lagrange $\mathbb{P}_1$ elements

Definition 34. The vector space on polynomials in one variable of degree less or equal than k is denoted by $\mathbb{P}_k$

$$\mathbb{P}_k = \left\{ p(x) = \sum_{j=0}^{k} \alpha_j x^j \quad \alpha_j \in \mathbb{R} \right\} \tag{A.37}$$

let us denote by

$$V_h^1 = \left\{ v_h \in C^0([0,1]), \quad v_h|_{K_j} \in \mathbb{P}_1, \ 0 \leq j \leq N \right\} \tag{A.38}$$

and

$$V_{0,h}^1 = \left\{ v_h \in V_h^1, \quad v_h(0) = v_h(1) = 0 \right\} \tag{A.39}$$

the approximation space, which is made of piece-wise linear polynomial functions. The finite element method consist then in applying the internal variational approximation approach to the spaces V_h^1 and $V_{0,h}^1$.

The approximation space V_h^1 is in itself a very good finite dimensional space in which we can try to make approximations using the internal variational approach, but in order to make calculations even easier, it is usefully to refine even more the search for a basis in which we can make projections.

Lemma 18.

The space V_h^1 is a subspace of $H^1(\Omega)$ of dimension $N+2$. Every functions $v_h \in V_h^1$ is uniquely determined by its values at the mesh vertices (x_j) :

$$v_h(x) = \sum_{j=0}^{N+1} v_h(x_j)\varphi(x), \quad \forall x \in [0,1] \tag{A.40}$$

where $(\varphi_j)_{0 \leq j \leq N+1}$ is the basis if the "shape" functions with compact support in each interval $[x_{j-1}, x_{j+1}]$ defined as:

$$\varphi_j(x) = \begin{cases} \frac{x - x_{j-1}}{h} & x \in [x_{j-1}, x_j] \\ \frac{x_{j+1} - x}{h} & x \in [x_j, x_{j+1}] \end{cases} \qquad \text{such that} \quad \varphi_j(x_i) = \delta_{ij} \tag{A.41}$$

Similarly, the space $V_{0,h}^1$ is a subspace of $H_0^1(\Omega)$ of dimension N and every function v_h of the space is determined by its values at the vertices $(x_j)_j$

$$v_h(x) = \sum_{j=0}^{N+1} v_h(x_j)\varphi(x), \quad \forall x \in [0,1] \tag{A.42}$$

Proof. let us work on the 1-dimensional case, let us write:

$$\varphi_1(x) = \frac{x_2 - x}{x_2 - x_1}, \quad \varphi_2(x) = \frac{x - x_1}{x_2 - x_1}$$

Notice how these elements are linear independent, assume that there exist constants c_1 and c_2 such that a linear combination of these functions equals zero for all $x \in [x_1, x_2]$:

$$c_1\varphi_1(x) + c_2\varphi_2(x) = 0$$

For all x, this implies:

$$c_1 \left(\frac{x_2 - x}{x_2 - x_1} \right) + c_2 \left(\frac{x - x_1}{x_2 - x_1} \right) = 0$$

Multiplying through by $x_2 - x_1$, we get:

$$c_1(x_2 - x) + c_2(x - x_1) = 0$$

This simplifies to:

$$c_1x_2 - c_1x + c_2x - c_2x_1 = 0$$

$$(c_1 - c_2)x + (c_2x_1 - c_1x_2) = 0$$

For this equation to hold for all x, both coefficients must be zero:

$$c_1 - c_2 = 0 \quad \text{and} \quad c_2x_1 - c_1x_2 = 0$$

From $c_1 - c_2 = 0$, we get $c_1 = c_2$. Substituting $c_1 = c_2$ into the second equation:

$$c_2x_1 - c_2x_2 = 0$$

Since $x_1 \neq x_2$, it must be that $c_2 = 0$. Thus, $c_1 = 0$ as well. Hence, the only solution is $c_1 = c_2 = 0$, showing that $\varphi_1(x)$ and $\varphi_2(x)$ are linearly independent.

We need to show that any such polynomial can be expressed as a linear combination of $\varphi_1(x)$ and $\varphi_2(x)$ which is consequence of the interpolation property of these basis functions. At $x = x_1$:

$$\varphi_1(x_1) = 1, \quad \varphi_2(x_1) = 0$$

At $x = x_2$:

$$\phi_1(x_2) = 0, \quad \phi_2(x_2) = 1$$

then for any linear polynomial $v_h(x)$, we can write:

$$v_h(x) = v_h(x_1)\phi_1(x) + v_h(x_2)\phi_2(x)$$

Moreover, these functions belongs to $C^1(\Omega)$ which is a subspace of $H^1(\Omega)$, then the result follows. Additionally the space $V^1_{0,h}$ is a subspace of $H^1_0(\Omega)$. □

These functions as presented are good enough by themselves to implement the FEM, however, it is useful for the computations of the linear system that solves the problem, to consider a slight variations.

We introduce the functions

$$\omega_0(x) = 1 - x, \quad \omega_1(x) = x. \tag{A.43}$$

With these, the shape functions are defined as the composition

$$\varphi_j(x) = \begin{cases} \omega_1\left(\frac{x-x_{j-1}}{x_j-x_{j-1}}\right), & x \in [x_{j-1}, x_j] \\ \omega_0\left(\frac{x-x_j}{x_{j+1-x_j}}\right), & x \in [x_j, x_{j+1}] \end{cases} \qquad \forall\ 0 \leq j \leq N. \tag{A.44}$$

and $\varphi_{N+1} = \omega_1(\frac{x+x_N}{h})$

To set the ideas and without loss of generality let us restrict to $\Omega = (0,1)$ and consider the boundary-value problem given by the one-dimensional Poisson equation (3.11):

Given $f \in L^2(\Omega)$, find the function u which solves:

$$\begin{cases} -u''(x) = f(x), & x \in (0,1) \\ u(0) = u(1) = 0 \end{cases} \tag{A.45}$$

The variational formulation consists in finding $u \in H_0^1(\Omega)$, such that:

$$\int_\Omega u'(x)v'(x)\,dx = \int_\Omega f(x)v(x)\,dx, \quad \forall v \in H_0^1(\Omega) \tag{A.46}$$

The variational formulation of the internal approximation consists then in finding $u_h \in V_{0,h}$, such that:

$$\int_\Omega u_h'(x)v_h'(x)\,dx = \int_\Omega f(x)v_h(x)\,dx, \quad \forall v_h \in V_{0,h} \tag{A.47}$$

Which leads to the approximate problem:

Find $u_h(x_1), \ldots, u_h(x_N)$ such that for all $i = 1, \ldots, N$:

$$\sum_{j=1}^{N} \left(\int_\Omega \varphi_i'(x)\varphi_j'(x)\,dx \right) u_h(x_j) = \int_\Omega f(x)\varphi_i(x)\,dx \tag{A.48}$$

Which is, as expected, equivalent to solving in R^N the linear system:

$$A_h U_h = F_h \tag{A.49}$$

Where,

- $U_h = (u_h(x_j))_{1 \le j \le N}$
- $F_h = (\int_\Omega f(x)\varphi_i(x)\,dx)_{1 \le i \le N}$
- $A_h = \left(\int_\Omega \varphi_i'(x)\varphi_j'(x)\,dx\right)_{1 \le i,j \le N}$

Coefficients of the Matrix A_h

First, recall that the shape functions φ_j have small supports for which we can rewrite

$$a_{ij} = \int_\Omega \varphi_i'(x)\varphi_j'(x)\,dx = \sum_{k=0}^{N} \int_{x_k}^{x_{k+1}} \varphi_i'(x)\varphi_j'(x)\,dx. \tag{A.50}$$

For most of the coefficients in A_h, the respective supports $[x_k, x_{k+1}]$ are actually disjoints, therefore most of the coefficients are zero. More precisely, for a given index i, there's only

three consecutive values of j such that the coefficient a_{ij} is potentially non-zero. The structure of the matrice is then easy to deduce: A_h is a tridiagonal matrix. The coefficients of A_h are thus given by:

$$\begin{aligned} a_{jj-1} &= \int_{x_{j-1}}^{x_j} \varphi_j'(x)\varphi_{j-1}'(x)\,dx \\ a_{jj} &= \int_{x_{j-1}}^{x_{j+1}} (\varphi_j'(x))^2\,dx \\ a_{jj+1} &= \int_{x_j}^{x_{j+1}} \varphi_j'(x)\varphi_{j+1}'(x)\,dx \end{aligned}$$

let us consider the element $K_j = [x_j, x_{j+1}]$; on this element there is only two non-zero shape functions:

$$\begin{aligned} \varphi_j|_{K_j} &= \frac{x_{j-1}-x}{x_{j+1}-x_j} = \frac{x_{j+1}-x}{h} \qquad & \varphi_{j+1}|_{K_j} &= \frac{x-x_j}{x_{j+1}-x_j} = \frac{x-x_j}{h} \\ \varphi_j'|_{K_j} &= \frac{-1}{x_{j+1}-x_j} = \frac{-1}{h} & \varphi_j'|_{K_j} &= \frac{1}{x_{j+1}-x_j} = \frac{1}{h} \end{aligned} \tag{A.51}$$

Then we can arrange the elementary contributions of the elements K_j as a 2×2 symmetric matrix:

$$EK_j = \begin{pmatrix} k_{11}^j & k_{12}^j \\ k_{21}^j & k_{22}^j \end{pmatrix} \tag{A.52}$$

with

$$\begin{aligned} k_{11}^j &= \int_{x_{j-1}}^{x_j} \varphi_j'(x)\varphi_{j-1}'(x)\,dx, & k_{12}^j = k_{21}^j &= \int_{x_{j-1}}^{x_{j+1}} (\varphi_j'(x))^2\,dx, & k_{22}^j &= \int_{x_j}^{x_{j+1}} \varphi_j'(x)\varphi_{j+1}'(x)\,dx \\ &= \int_{x_{j-1}}^{x_j} \frac{1}{h^2}\,dx = \frac{1}{h} & &= \int_{x_{j-1}}^{x_{j+1}} -\frac{1}{h^2}\,dx = -\frac{1}{h} & &= \int_{x_j}^{x_{j+1}} \frac{1}{h^2}\,dx = \frac{1}{h} \end{aligned} \tag{A.53}$$

and thus

$$EK_j = \frac{1}{h}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{A.54}$$

which altogether form the matrix A_h by taking the sum

$$A_h = \frac{1}{h}\begin{bmatrix} 1 & -1 & & & & \\ -1 & 1 & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} + \frac{1}{h}\begin{bmatrix} & & & & & \\ & 1 & -1 & & & \\ & -1 & 1 & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} + \cdots + \frac{1}{h}\begin{bmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & 1 & -1 & \\ & & & -1 & 1 & \end{bmatrix} \tag{A.55}$$

$$A_h = \begin{pmatrix} \frac{1}{h} & -\frac{1}{h} & & & & & \\ -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & & & & \\ & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & \\ & & & & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} \\ & & & & & -\frac{1}{h} & \frac{1}{h} \end{pmatrix} \tag{A.56}$$

Coefficients of the vector F_h

Each component f_i of the vector $F_h \in R^N$ is obtained as:

$$f_i = \sum_{k=1}^{N} \int_{x_k}^{x_{k+1}} f(x)\varphi_i(x)\,dx. \tag{A.57}$$

but since most of the integrals are zero because of the support of $\varphi_i(x)$, it reduces to

$$f_i = \int_{x_{i-1}}^{x_i} f(x)\varphi_i(x)\,dx + \int_{x_i}^{x_{i+1}} f(x)\varphi_i(x)\,dx \tag{A.58}$$

Usually, the function f is not known analytically, but at least we should know its values at the node points, hence, we can decompose f in the basis of the shape functions $(\varphi_j)_j$ as

$$f(x) = \sum_{j=1}^{N} f(x_j)\varphi_j(x) \tag{A.59}$$

for which the problem is reduces to the evaluation of the integrals:

$$\int_{x_k}^{x_{k+1}} \varphi_i(x)\varphi_j(x)\ dx. \tag{A.60}$$

Either way, we can't hope to solve this integral analytically, however we can approximate using a quadrature rule. Using the trapezoidal rule, we have:

$$\begin{aligned} f_i &= \int_{x_{i-1}}^{x_{i+1}} f(x)\varphi_i(x)\ dx \\ &= \int_{x_{i-1}}^{x_i} f(x)\varphi_i(x)\ dx + \int_{x_i}^{x_{i+1}} f(x)\varphi_i(x)\ dx \\ &\approx \frac{(f(x_{i-1})\varphi_i(x_{i-1}) + f(x_i)\varphi_i(x_i))\,h}{2} \\ &\quad + \frac{(f(x_i)\varphi_i(x_i) + f(x_{i+1})\varphi_i(x_{i+1}))\,h}{2} \\ &= \frac{(0 + f(x_i))\,h}{2} + \frac{(f(x_i) + 0)\,h}{2} \\ &= \frac{f(x_i)2h}{2} = f(x_i)h \end{aligned} \tag{A.61}$$

Bibliography

[1] Akrivis, G., Makridakis, C., and Nochetto, R. H. (2006). A posteriori error estimates for the Crank-Nicolson method for parabolic equations. *Mathematics of Computation*, 75(254), 511–531.http://www.jstor.org/stable/4100298.

[2] Apostol, T. M. (1984). *Mathematical Analysis*. Addison-Wesley.

[3] Baharlouei, S., Taylor, J. M., Uriarte, C., & Pardo, D. (2025). A Least-Squares-Based Neural Network (LS-Net) for Solving Linear Parametric PDEs. *Computer Methods in Applied Mechanics and Engineering*, 437, 117757. doi:10.1016/j.cma.2025.117757.

[4] Badia, S., Li, W., & Martín, A. F. (2024). Finite element interpolated neural networks for solving forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 418, 116505. doi:10.1016/j.cma.2023.116505.

[5] Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2015). Automatic differentiation in machine learning: A survey. arXiv. https://arxiv.org/abs/1502.05767.

[6] Becker, B. R., & Fricke, B. A. (2004). Heat transfer coefficients for forced-air cooling and freezing of selected foods. *International Journal of Refrigeration*, 27(5), 540–551.

[7] Bellumori, M., Angeloni, G., Guerrini, L., et al. (2021). Effects of different stabilization techniques on the shelf life of cold brew coffee: Chemical composition, flavor profile, and microbiological analysis. *LWT*, 142, 111043.

[8] Berrone, S., Canuto, C., and Pintore, M. (2022). Variational Physics Informed Neural Networks: The Role of Quadratures and Test Functions. *arXiv preprint*, arXiv:2206.04941. https://arxiv.org/abs/2206.04941.

[9] Berrone, S., Canuto, C., & Pintore, M. (2022). Solving PDEs by variational physics-informed neural networks: An a posteriori error analysis. *Annali dell'Università di Ferrara*, 68, 575–595. doi:10.1007/s11565-022-00393-2.

[10] Berrone, S., Canuto, C., & Pintore, M. (2022). Variational physics-informed neural networks: The role of quadratures and test functions. *Journal of Scientific Computing*, 92, 100. doi:10.1007/s10915-022-01877-3.

[11] Brezis, H. (2010). *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. Springer Science and Business Media.

[12] Burden, R. L., & Faires, J. D. (2010). *Numerical Analysis* (9th ed.). Cengage Learning.

[13] Calabrò, F., Cuomo, S., di Serafino, D., Izzo, G., and Messina, E. (2023). Time discretization in the solution of parabolic PDEs with ANNs. *Applied Mathematics and Computation*, 458, 128230. https://doi.org/10.1016/j.amc.2023.128230.

[14] Cengel, Y. A., & Ghajar, A. J. (2020). *Heat and Mass Transfer: Fundamentals and Applications* (6th ed.). McGraw Hill.

[15] Cimrman, R., Lukes, V., & Rohan, E. (2019). Multiscale finite element calculations in Python using sfepy. *Advances in Computational Mathematics*.

[16] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4), 303–314. doi:10.1007/BF02551274.

[17] Cheng, A., & Cheng, D. T. (2005). Heritage and early history of the boundary element method. *Engineering Analysis with Boundary Elements*, 29(3), 268–302.

[18] Delgado, A., & Sun, D. W. (2001). Heat and mass transfer models for predicting freezing processes – a review. *Journal of Food Engineering*, 47, 157–174.

[19] Davies, E. B. (1995). *Spectral Theory and Differential Operators*. Cambridge University Press.

[20] Ern, A., and Guermond, J. L. (2021). *Finite Elements III: First-Order and Time-Dependent PDEs*. Springer. https://doi.org/10.1007/978-3-030-57348-5.

[21] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8.

[22] Jagtap, A. D., Kharazmi, E., & Karniadakis, G. E. (2020). Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365, 113028. doi:10.1016/j.cma.2020.113028.

[23] Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, 3, 422–440. doi:10.1038/s42254-021-00314-5.

[24] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017). Physics Informed Deep Learning (Part I): Data-driven solutions of nonlinear partial differential equations. arXiv.https://arxiv.org/abs/1711.10561.

[25] Karlsson, H. F. (2023). Solving partial differential equations with neural networks (Dissertation). Uppsala University. https://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-499348.

[26] Kharazmi, E., Zhang, Z., & Karniadakis, G. E. (2021). hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374, 113547. doi:10.1016/j.cma.2020.113547.

[27] Kharazmi, E., Zhang, Z., and Karniadakis, G. E. (2019). Variational Physics-Informed Neural Networks for solving partial differential equations. arXiv. https://arxiv.org/abs/1912.00873.

[28] Khodayi-Mehr, R., & Zavlanos, M. (2020). VarNet: Variational neural networks for the solution of partial differential equations. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, PMLR, pp. 298–307.

[29] Khodayi-Mehr, R., & Zavlanos, M. M. (2020). Deep learning for robotic mass transport cloaking. *IEEE Transactions on Robotics*, 36(4), 967–974. doi:10.1109/TRO.2020.2978604.

[30] Metropolis, N., & Ulam, S. (1949). The Monte Carlo method. *Journal of the American Statistical Association*, 44(247), 335–341. doi:10.1080/01621459.1949.10483310.

[31] Moreno, F., Quintanilla-Carvajal, M., Sotelo, L., et al. (2015). Volatile compounds, sensory quality and ice morphology in falling-film and block freeze concentration of coffee extract. *Journal of Food Engineering*, 166, 64–71.

[32] Pinkus, A. (1999). Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8, 143–195. doi:10.1017/S0962492900002919.

[33] Poisson, S. D. (1823). Mémoire sur la théorie du magnétisme en mouvement. *Mémoires de l'Académie Royale des Sciences de l'Institut de France*, 5, 247–338.

[34] Presiga-Posada, N., Restrepo-Díaz, C., Velasquez, S., Gil, L., & Muñoz, D. A. (2024). Study on the industrial freezing process of coffee extracts under different conditions using a validated phenomenological model. Manuscript submitted for publication in *Journal of Food Engineering*.

[35] Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, 686–707. doi:10.1016/j.jcp.2018.10.045.

[36] Rivera, J. A., Taylor, J. M., Omella, Á. J., & Pardo, D. (2022). On quadrature rules for solving partial differential equations using neural networks. *Computer Methods in Applied Mechanics and Engineering*, 393, 114710. doi:10.1016/j.cma.2022.114710.

[37] Rojas, S., Maczuga, P., Muñoz-Matute, J., Pardo, D., and Paszyński, M. (2024). Robust Variational Physics-Informed Neural Networks. *Computer Methods in Applied Mechanics and Engineering*, 425, 116904. https://doi.org/10.1016/j.cma.2024.116904.

[38] Santos, C., Carciofi, B., et al. (2007). Determination of heat transfer coefficient in cooling-freezing tunnels using experimental time–temperature data. *Journal of Food Process Engineering*, 30, 717–728.

[39] Silva, A. C. C., et al. (2023). Effect of concentration and temperature on the physical and thermophysical properties of coffee extract. *Journal of Food Engineering*, 340, 111304.

[40] Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375, 1339–1364. doi:10.1016/j.jcp.2018.08.029.

[41] Taylor, J. M., Bastidas, M., Calo, V. M., and Pardo, D. (2024). Adaptive deep Fourier residual method via overlapping domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 427, 116997. https://doi.org/10.1016/j.cma.2024.116997.

[42] Taylor, J. M., Bastidas, M., Pardo, D., & Muga, I. (2023). Deep Fourier Residual method for solving time-harmonic Maxwell's equations. *Preprint*, CUNEF University, University of the Basque Country, Basque Centre for Applied Mathematics, Ikerbasque, and Pontificia Universidad Católica de Valparaíso.

[43] Taylor, J. M., Pardo, D., and Muga, I. (2022). A deep Fourier residual method for solving PDEs using neural networks. arXiv. https://arxiv.org/abs/2203.09341.