Mathematical and Computational Applications

MDPI

Article

Error Estimates and Generalized Trial Constructions for Solving ODEs Using Physics-Informed Neural Networks

Atmane Babni [1], Ismail Jamiai [1,*] and José Alberto Rodrigues [2,3]

[1] LaR2A Laboratory, Department of Mathematics, Faculty of Sciences of Tetouan, Abdelmalek Essaadi University, Tetouan 93030, Morocco; atmane.babni@etu.uae.ac.ma
[2] CIMA and Department of Mathematics, Instituto Superior de Engenharia de Lisboa—ISEL, Polytechnic University of Lisbon, 1959-007 Lisbon, Portugal; jose.rodrigues@isel.pt
[3] Centro de Investigação em Modelação e Optimização de Sistemas Multifuncionais—CIMOSM, Instituto Superior de Engenharia de Lisboa—ISEL, 1959-007 Lisbon, Portugal
* Correspondence: ijamiai@uae.ac.ma

Abstract

In this paper, we address the challenge of solving differential equations using physics-informed neural networks (PINNs), an innovative approach that integrates known physical laws into neural network training. The PINN approach involves three main steps: constructing a neural-network-based solution ansatz, defining a suitable loss function, and minimizing this loss via gradient-based optimization. We review two primary PINN formulations: the standard PINN I and an enhanced PINN II. The latter explicitly incorporates initial, final, or boundary conditions. Focusing on first-order differential equations, PINN II methods typically express the approximate solution as $\tilde{u}(x,\theta) = P(x) + Q(x)N(x,\theta)$, where $N(x,\theta)$ is the neural network output with parameters θ, and $P(x)$ and $Q(x)$ are polynomial functions. We generalize this formulation by replacing the polynomial $Q(x)$ with a more flexible function $\phi(x)$. We demonstrate that this generalized form yields a uniform approximation of the true solution, based on Cybenko's universal approximation theorem. We further show that the approximation error diminishes as the loss function converges. Numerical experiments validate our theoretical findings and illustrate the advantages of the proposed choice of $\phi(x)$. Finally, we outline how this framework can be extended to higher-order or other classes of differential equations.

Keywords: physics-informed neural networks (PINNs); differential equations; universal approximation theorem; loss function; error estimates

Academic Editor: Alexandre Souto Martinez

Received: 23 September 2025
Revised: 23 October 2025
Accepted: 19 November 2025
Published: 24 November 2025

Citation: Babni, A.; Jamiai, I.; Rodrigues, J.A. Error Estimates and Generalized Trial Constructions for Solving ODEs Using Physics-Informed Neural Networks. *Math. Comput. Appl.* **2025**, *30*, 127. https://doi.org/10.3390/mca30060127

1. Introduction

The numerical solution of ordinary differential equations (ODEs) underpins much of scientific computation. Classical numerical methods (e.g., Runge–Kutta schemes) can struggle with stiff systems and complex boundary conditions (see [1,2] for a detailed discussion). In recent years, Physics-Informed Neural Networks (PINNs) have emerged as a promising mesh-free alternative that integrates physical laws directly into the training loss [3]. However, standard PINN formulations typically enforce initial or boundary conditions via penalty terms, which may lead to reduced accuracy near boundaries and slow convergence [4]. In a PINN, a neural network $N(x;\theta)$ is trained so that its output nearly satisfies the ODE $\mathcal{F}[N] = f$, and any specified initial/boundary conditions, across the domain without relying on labeled solution data. One minimizes a loss comprising the mean-squared residual of the differential operator plus terms enforcing the constraints.

While standard PINN formulations typically enforce boundary or initial conditions via penalty terms added to the loss function [5], this soft-constraint approach may lead to suboptimal accuracy near the boundaries, slow convergence, and sensitivity to the choice of penalty weights. Our approach aims to address certain limitations of soft constraints by exactly satisfying the initial/boundary conditions. To overcome these limitations, we adopt a generalized trial solution strategy that embeds the conditions into the structure of the approximation itself. Specifically, we construct trial functions of the form

$$u_\theta(x) = u_0 + \phi(x) \cdot N_\theta(x),$$

where $\phi(x)$ is a smooth function vanishing at the constraint point(s). This ensures that the network output automatically satisfies the prescribed condition(s), thus eliminating the need for explicit penalization. Such hard-constrained formulations are expected to improve training stability and enforce boundary consistency in challenging scenarios.

This approach combines the expressive power of neural networks with physical constraints: by teaching the network the differential law, one can achieve solutions that are data-efficient and physics-consistent. By the universal approximation theorem [6,7], a sufficiently large feedforward network can approximate any continuous function on a compact domain. Hence, there exists in principle a network arbitrarily close to the true solution of a well-posed ODE. However, the classical universal approximation result is non-constructive and does not offer training guidance or error control. This motivates the introduction of structured trial functions in PINNs.

Lagaris et al. [4] proposed using trial solutions of the form

$$\tilde{u}(x, \theta) = P(x) + Q(x)N(x, \theta)$$

where $P(x)$ satisfies the boundary/initial conditions, and $Q(x)$ vanishes on those boundaries. The trainable network $N(x, \theta)$ fine-tunes the solution while inherently respecting the constraints. This has been shown to accelerate convergence compared to unconstrained networks. Generalizations have introduced basis expansions (polynomial, Fourier, and radial) to enforce smoothness and asymptotic properties. These enhancements remain within the scope of universal approximators [6,7].

In this work, we systematically generalize and analyze the class of hard-constrained trial constructions (Lagaris-type), providing admissibility conditions for ϕ and proving error estimates linking the PINN loss to the true solution error for first-order ODEs.

Related Work

Physics-Informed Neural Networks (PINNs) were formalized by Raissi et al. [3] as a deep learning framework that integrates physical law constraints into the training process. PINNs use automatic differentiation to evaluate the differential equation residuals and include these residuals, along with boundary and initial condition terms, as soft constraints in the loss function during training [3]. By penalizing violations of the governing equations and boundary conditions in the loss, a PINN can learn solutions without any observed data, effectively bridging data-driven learning and physics-based modeling. This mesh-free formulation avoids the need for spatial discretization and can mitigate issues of traditional methods (like meshing or the curse of dimensionality in high-dimensional problems). PINNs have achieved notable success across diverse application areas, including fluid dynamics (e.g., solving Navier–Stokes flow problems) [5], hemodynamics and biomedical flow modeling [8,9], and structural/mechanical system simulations [10]; see also [11] for solar coronal magnetic fields. Their ability to embed physical knowledge leads to improved generalization from limited data, and numerous studies report PINNs outperforming

black-box neural networks in data-scarce scenarios. Comprehensive surveys of the field document these wide-ranging successes and the rapid methodological advancements in PINNs [12,13].

Despite their promise, standard PINN formulations have certain limitations. A key issue is that PINNs typically enforce initial and boundary conditions via penalty terms added to the loss, rather than satisfying them exactly [3]. This soft-constraint approach can result in suboptimal accuracy near domain boundaries and can slow down convergence, especially if the penalty weights are not tuned properly [14]. To address this, recent research has revisited the idea of hard constraint trial functions that inherently satisfy the boundary or initial conditions. In the original ANN-based solver of Lagaris et al. [4], the neural network was embedded in a trial solution of the form $u(x) = P(x) + Q(x)N(x, \theta)$, where $P(x)$ satisfies the boundary conditions and $Q(x)$ vanishes on the boundaries— thus the network $N(x, \theta)$ only learns the residual part of the solution and automatically respects the constraints. This strategy eliminates the need for boundary penalties and was shown to accelerate convergence compared to unconstrained networks. Following this approach, contemporary PINN works have employed similar hard-constrained trial solutions to improve training stability and accuracy [4,15]. Müller and Zeinhofer [15] analyzed the effect of imposing exact boundary values and proved that enforcing constraints explicitly can change the convergence behavior of residual minimization, leading to more efficient training. In our context of ODEs, we adopt a generalized trial solution strategy (inspired by these works) where the neural network output is composed with a function that vanishes at the boundaries, ensuring the PINN exactly satisfies the initial or boundary conditions from the outset. This approach, also used in some recent PINN extensions [14,15], removes the burden of penalty tuning and improves solution accuracy near the boundaries.

Another active area of research on PINNs is the development of theoretical error analyses and convergence guarantees. Early theoretical studies provided insight into why and when PINNs work. For example, Shin et al. proved the convergence of PINN solutions for certain linear elliptic and parabolic PDEs, under appropriate training conditions [14]. Building on this, Shin et al. derived a priori error estimates for PINNs by analyzing the residual minimization error for linear PDE problems, obtaining bounds on the solution error in terms of the PINN's training loss [16]. Mishra and Molinaro focused on generalization error bounds for PINNs, showing that for some inverse PDE problems the error between the PINN solution and true solution can be bounded (with high probability) as the network capacity and training set grow, effectively providing learning guarantees for PINNs [17,18]. In the specific context of ODEs, Yoo et al. [19] established rigorous L^2 error estimates for PINNs solving initial value problems, proving that the PINN's solution error is controlled by (and in fact, on the order of) the training loss—thus as the loss is driven to zero, the PINN solution converges to the true solution. This result offers a theoretical foundation that the PINN error will vanish if one can sufficiently minimize the loss [19]. Likewise, in the domain of fluid dynamics, Biswas et al. [20] provided one of the first analyses of PINN-type methods, deriving explicit error estimates and stability analyses for deep learning solutions of Navier–Stokes equations. Their work quantifies how the approximation error depends on network parameters and training iterations for a fixed network size [20]. These theoretical advances (e.g., [14,16,19–21]) not only increase confidence in PINNs but also guide the design of improved algorithms.

Beyond these, researchers have proposed various enhancements to improve PINN performance. For instance, adaptive activation functions and architecture changes (including the use of radial basis function networks or input feature embedding based on the Kolmogorov–Arnold theorem) have been explored to help the network train faster or capture complex solution behavior [22]. Domain-decomposition strategies and hybrid

algorithms that combine PINNs with traditional solvers have achieved significant speed-ups and accuracy gains in large-scale problems [23]. Some studies integrate PINNs with multi-fidelity or transfer learning techniques to handle multi-scale physics, while others introduce Sobolev training and other modified loss functions to improve convergence in stiff or chaotic systems [24]. A recent trend is to develop open-source tools and libraries to facilitate PINN implementations—for example, NeuroDiffEq provides a Python library for solving ODEs/PDEs with neural networks, lowering the barrier to experimenting with PINN models [25]. With the community rapidly expanding, several extensive review articles have appeared that catalog these developments and outline future directions for PINNs [12,13]. These surveys highlight that while PINNs have achieved impressive results, there remain open challenges such as ensuring training stability, handling complex geometries or multi-physics coupling, and providing stronger theoretical guarantees for general cases.

In summary, the literature shows a clear trajectory from early ANN-based differential equation solvers [4,6,7] to the modern PINN framework [3] and its many variants. PINNs have been applied successfully across scientific and engineering domains, and researchers have introduced various improvements—from hard-constrained trial functions [4,15] and penalty/stability enhancements [26], to advanced training techniques and hybrid models; see also hands-on tutorials [27], to enhance their accuracy and efficiency. At the same time, our understanding of why PINNs work is improving thanks to new convergence and error analyses [14,16,19]. Building on these insights, the present work contributes to this evolving landscape by combining rigorous error estimates with an enhanced PINN formulation that employs generalized trial solutions for ODEs. Our approach aims to ensure that the PINN exactly satisfies the problem's initial conditions (removing penalty-induced errors) while providing theoretical guarantees on the solution error. This fusion of improved trial function design with provable error bounds addresses some of the key limitations identified in prior works, moving one step closer to reliable and efficient physics-informed neural network solvers for ODEs.

The remainder of this paper is structured as follows: Section 2 introduces the mathematical background. In Section 3 the general formulation of the PINN framework is presented along with the construction of trial solutions tailored for first-order nonlinear ODEs. Section 4 presents the main theoretical results, including universal approximation properties and rigorous error estimates linking the training loss to the approximation error. Section 5 provides numerical experiments illustrating the accuracy and efficiency of the proposed method on benchmark nonlinear problems. Finally, Section 6 concludes the work and outlines future directions, including possible extensions to higher-order equations and inverse problems.

2. Mathematical Background

2.1. Cybenko's Theorem and Universality of 3-Layer Wide Neural Networks

2.1.1. Cybenko's Theorem

Let $\Omega \subset \mathbb{R}^d$ be compact. Let $\sigma : \mathbb{R} \to \mathbb{R}$. Let f_θ represent width-N 2-layer neural networks

$$f_\theta(x) = \sum_{i=1}^{N} u_i \sigma(a_i^T x + b_i), \tag{1}$$

in this formulation, $a_i \in \mathbb{R}^d$ are the weight vectors (often collectively denoted as a weight matrix), $b_i \in \mathbb{R}$ are the bias terms, and σ is the activation function applied element-wise. The coefficients u_i are the output weights that linearly combine the activated neu-

rons' outputs. Together, $\theta = \{a_i, b_i, u_i\}$ denotes the set of all network parameters, where $\theta \in \Theta^{(N)}$ and

$$\Theta^{(N)} = \{(a_1, \ldots, a_N, b_1, \ldots, b_N, u_1, \ldots, u_N) \mid a_1, \ldots, a_N \in \mathbb{R}^d,\ b_1, \ldots, b_N, u_1, \ldots, u_N \in \mathbb{R}\}.$$

To clarify, $\Theta^{(N)} \cong \mathbb{R}^{dN+2N}$.

Theorem 1 (Theorem 1 in [28]). *Let $\Omega \subset \mathbb{R}^d$ be compact. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a continuous function satisfying*

$$\lim_{r \to -\infty} \sigma(r) = 0, \qquad \lim_{r \to \infty} \sigma(r) = 1.$$

The class of functions

$$\bigcup_{N \in \mathbb{N}} \{f_\theta\}_{\theta \in \Theta^{(N)}}$$

is dense in $C(\Omega), \|\cdot\|_\infty$, i.e.,

$$closure\Big(span\Big(\{\sigma(a^T x + b)\}_{a \in \mathbb{R}^d, b \in \mathbb{R}}\Big)\Big) = (C(\Omega), \|\cdot\|_\infty).$$

The consequence of this theorem is that for any $f_\star \in C(\Omega)$ and $\varepsilon > 0$, there exists a large enough N and network parameter $\theta \in \Theta^{(N)}$ such that

$$\sup_{x \in \Omega} |f_\theta(x) - f_\star(x)| < \varepsilon.$$

Smaller ε will likely require larger N, but this theorem or its proof will not allow us to make any quantitative claims. Moreover, this is an existence result; it does not tell us how to find N and $\theta \in \Theta^{(N)}$.

2.1.2. Universality of 3-Layer Wide Neural Networks

Consider three-layer neural networks of the form

$$f_\theta(x) = A^{(3)}\sigma_2\Big(A^{(2)}\sigma_1\Big(A^{(1)}x + b^{(1)}\Big) + b^{(2)}\Big) + b^{(3)}, \tag{2}$$

where $N_1, N_2 \in \mathbb{N}$, $A^{(1)} \in \mathbb{R}^{N_1 \times d}$, $b^{(1)} \in \mathbb{R}^{N_1}$, $A^{(2)} \in \mathbb{R}^{N_2 \times N_1}$, $b^{(2)} \in \mathbb{R}^{N_2}$, $A^{(3)} \in \mathbb{R}^{1 \times N_2}$, $b^{(3)} \in \mathbb{R}^1$, $\sigma_1 : \mathbb{R} \to \mathbb{R}$, $\sigma_2 : \mathbb{R} \to \mathbb{R}$, and σ_1 and σ_2 are applied elementwise.

Universality of wide neural networks of depth 3 or deeper, requires a little bit of additional work to establish.

Theorem 2 (Theorem 12 in [28]). *Let $\Omega \subseteq \mathbb{R}^d$ be compact. Assume $\sigma_2 \in C^\infty(\mathbb{R})$ and $\sigma_1 \in C(\mathbb{R})$ are non-polynomial. Then the class of 3-layer neural networks of the form* (2) *are dense in $(C(\Omega), \|\cdot\|_\infty)$.*

Theorem 2 provides a universality result for neural networks with three layers (depth-3), under the assumption that the activation functions σ_1 and σ_2 are non-polynomial, with σ_2 being infinitely differentiable. It confirms that such networks can approximate any continuous function defined on a compact domain $\Omega \subseteq \mathbb{R}^d$ arbitrarily closely in the uniform norm. This expands the scope of classical universal approximation theorems, which typically focus on depth-2 networks, by demonstrating that adding depth still preserves—if not enhances—the expressive power. Importantly, the result is purely existential: it ensures the existence of suitable network parameters and width but does not offer a constructive approach to identify them in practice.

2.2. Existence, Uniqueness of the ODE Solution

Consider the initial value problem given by the ODE,

$$\begin{cases} u'(t) = G(t,\, u(t)), \\ u(t_0) = u_0\,, \end{cases} \tag{3}$$

with $(t_0, u_0) \in \mathbb{R}^2$. We assume that G is Lipschitz continuous in its second argument on the rectangle $[\,t_0 - a_1,\, t_0 + a_2\,] \times [\,u_0 - b_1,\, u_0 + b_2\,]$. In particular, suppose there exist constants $M_0 > 0$ and $K > 0$ such that

$$|G(t,u)| \le M_0, \qquad \text{for all } (t,u) \text{ in } [\,t_0 - a_1,\, t_0 + a_2\,] \times [\,u_0 - b_1,\, u_0 + b_2\,],$$

and

$$|G(t,u_1) - G(t,u_2)| \le K\,|u_1 - u_2|,$$

for all (t,u_1) and (t,u_2) in $[\,t_0 - a_1,\, t_0 + a_2\,] \times [\,u_0 - b_1,\, u_0 + b_2\,]$.

Define $a := \min\{a_1, a_2\}$ and $b := \min\{b_1, b_2\}$, and let

$$c := \min\{\,a,\; b/M_0\,\}\,.$$

Consider the compact time interval $J := [\,t_0 - c,\, t_0 + c\,]$. By construction of c, we have $J \subseteq [\,t_0 - a_1,\, t_0 + a_2\,]$ and also $c \le b/M_0$. The latter implies that any function on J with derivative bounded by M_0 stays in the strip $[\,u_0 - b,\, u_0 + b\,]$. In particular, any solution $u(t)$ of (3) on J will satisfy $u(t) \in [\,u_0 - b,\, u_0 + b\,]$ for all $t \in J$. Thus, a solution on J remains in the region where G is Lipschitz continuous and bounded.

It is a classical result (obtained via Picard iteration; see, e.g., ([1], Chapter 5, Section 4)) that under the above conditions there exists a continuously differentiable function $\varphi : J \to \mathbb{R}$ which solves the initial value problem (3) on J. In other words, $\varphi(t)$ satisfies

$$\varphi'(t) = G(t,\, \varphi(t)) \quad \text{for all } t \in J, \qquad \varphi(t_0) = u_0\,,$$

and φ is the unique function with these properties. More precisely, if $\psi : I \to \mathbb{R}$ is any other continuously differentiable solution of (3) on some compact interval $I \subseteq [\,t_0 - a_1,\, t_0 + a_2\,]$ containing t_0, then $\psi(t) = \varphi(t)$ for all $t \in I \cap J$. This establishes the existence and uniqueness of a local solution. Moreover, the solution $\varphi(t)$ depends continuously on the initial data (t_0, u_0), a property often referred to as the *stability* of the solution with respect to the initial condition.

3. Physics-Informed Neural Networks Method

3.1. Function Approximation via Neural Networks

In this section, we present the core concept of approximating real-valued functions using neural networks within the context of machine learning. A neural network, while biologically inspired by the structure of the human brain, is rigorously defined as a composition of affine transformations interlaced with nonlinear activation functions. Among the various architectures, the three-layer neural network (with two hidden layers) is particularly prominent and widely utilized in practice. This architecture offers a good balance between expressive power and ease of training. As shown by Chester [29], networks with two hidden layers can approximate complex functions more efficiently than single-layer networks, while avoiding the training difficulties of deeper architectures. To approximate

a real-valued function on a subset of $\mathbb{R}$, we consider a mapping $N : \mathbb{R} \to \mathbb{R}$ defined as follows:

$$\boxed{N(t,\theta) = A_3\sigma(A_2\sigma(A_1 t + b_1) + b_2) + b_3}, \quad t \in \mathbb{R}. \tag{4}$$

Here, $A_1 \in \mathbb{R}^{n_1\times 1}$, $A_2 \in \mathbb{R}^{n_2\times n_1}$, and $A_3 \in \mathbb{R}^{1\times n_2}$ are weight matrices, while $b_1 \in \mathbb{R}^{n_1\times 1}$, $b_2 \in \mathbb{R}^{n_2\times 1}$, and $b_3 \in \mathbb{R}$ are bias vectors. The function σ denotes a nonlinear activation function applied element-wise; in this context, we employ the hyperbolic tangent (`tanh`). The parameter vector θ encompasses all trainable components: $\theta = \theta(n_1, n_2, A_1, A_2, A_3, b_1, b_2, b_3)$.

The justification for employing neural networks to approximate functions is grounded in a fundamental result from functional analysis known as the *universal approximation theorem* (see [4,19]). This theorem guarantees that a neural network of sufficient width can uniformly approximate any continuous function on a compact domain. Specifically, given any continuous target function g defined on a compact interval I and any tolerance $\varepsilon > 0$, there exist integers $n_1, n_2 \in \mathbb{N}$ and corresponding matrices A_1, A_2, A_3 and vectors b_1, b_2, b_3 such that the neural network $N(t,\theta)$ satisfies

$$|N(t,\theta) - g(t)| < \varepsilon, \quad \text{for all } t \in I,$$

where $\theta = \theta(A_1, A_2, A_3, b_1, b_2, b_3)$. This result ensures that g can be approximated arbitrarily well by a suitable neural network. The practical challenge then becomes identifying the parameters θ that yield a satisfactory approximation. By fixing n_1 and n_2 to sufficiently large values, the task reduces to optimizing the entries of the matrices and vectors comprising θ.

In supervised learning scenarios, the approximation is typically guided by empirical data. Suppose we are given a dataset of the form

$$D = \{(t_i, y_i)\}_{1\le i\le n}, \quad \text{where } y_i = g(t_i),$$

for some function g and $n \in \mathbb{N}$. The neural network is trained to minimize the discrepancy between its output and the data using the loss function

$$\text{Loss}(\theta) := \left(\frac{1}{n}\sum_{i=1}^{n}(N(t_i,\theta) - y_i)^2\right)^{1/2}.$$

Optimization algorithms such as gradient descent are employed to adjust θ in a way that minimizes this loss, thereby aligning $N(\cdot,\theta)$ with the underlying function g.

We now turn to the application of the neural network defined in (4) for solving initial value problems associated with ordinary differential equations, as specified in (3). Unlike data-driven tasks, in this context we do not rely on predefined datasets. Instead, the training objective stems from the differential operator itself, encoding the governing physical law. Accordingly, we will define a loss function based on the differential equation in order to guide the neural network towards a solution. The construction of such a loss function will be discussed in the following subsection.

3.2. Two Approaches to Approximating the Solution of (3) Using PINNs

We now focus on approximating the solution of the initial value problem (3) through the use of neural networks within the Physics-Informed Neural Network (PINN) framework.

PINN I: We begin by considering a neural network N of the form introduced in (4). Two distinct loss components are defined: the differential equation loss L_{de} and the initial condition loss L_{ic}, given by

$$L_{de}(\theta) := \left(\frac{1}{n} \sum_{i=1}^{n} \left(N'(t_i, \theta) - G(t_i, N(t_i, \theta)) \right)^2 \right)^{1/2}, \quad L_{ic}(\theta) := |N(t_0, \theta) - u_0|. \tag{5}$$

Here, $S = \{t_1, \ldots, t_n\}$ denotes a set of sample points chosen uniformly at random from the domain where the solution is sought. We opted for the absolute error formulation

$$L_{ic}(\theta) := |N(t_0, \theta) - u_0|$$

to enforce a strict match at the initial condition. Since this term contributes only once to the overall loss (as opposed to the averaged residuals in $L_{de}(\theta)$), we found that the absolute error sufficiently penalizes deviations at the initial point without introducing unnecessary complexity.

That said, we acknowledge that using a mean squared error (MSE) formulation, such as

$$L_{ic}(\theta) := (N(t_0, \theta) - u_0)^2,$$

is also a valid and commonly used alternative. This choice can yield smoother gradients, which might benefit optimization in certain settings. However, given that L_{ic} is a single scalar term, the practical difference between absolute and squared error is often negligible unless weighted heavily.

The overall loss function is then defined as:

$$L(\theta) := L_{de}(\theta) + L_{ic}(\theta). \tag{6}$$

The parameters θ of the neural network are optimized to minimize $L(\theta)$, typically using gradient descent. It is important to emphasize that the loss function L does not represent the exact error between the true solution and the neural network approximation. However, as we will demonstrate in Section 4, a rigorous mathematical relationship exists between the loss and the actual approximation error.

PINN II: Since L_{de} and L_{ic} may not simultaneously approach zero during training, an alternative strategy involves modifying the trial function itself to enforce the initial condition exactly. Define a new function N_A by

$$N_A(t, \theta) := u_0 + (t - t_0) N(t, \theta), \quad t \in \mathbb{R}, \tag{7}$$

where N is the neural network given by (4). This construction guarantees that $N_A(t_0, \theta) = u_0$, thus inherently satisfying the initial condition of (3).

Correspondingly, we define the modified loss function L_A based solely on the differential equation residual:

$$L_A(\theta) = \left(\frac{1}{n} \sum_{i=1}^{n} \left(N_A'(t_i, \theta) - G(t_i, N_A(t_i, \theta)) \right)^2 \right)^{1/2}, \tag{8}$$

where the sampling set $S = \{t_1, \ldots, t_n\}$ is chosen as before. Optimization is then performed on $L_A(\theta)$ to find the best-fit parameters θ that minimize the residual.

4. Main Results

In general, the PIIN-II approach aims to construct an approximate solution to problem (3) that satisfies the initial, final, or boundary conditions. However, such constructions typically rely exclusively on polynomial-based representations, even when dealing with higher-order differential equations. To extend this framework beyond polynomials, we consider a more general trial function of the form:

$$N_\phi(t,\theta) := u_0 + \phi(t) \cdot N(t,\theta), \quad t \in \mathbb{R}, \tag{9}$$

specifically in the context of first-order ordinary differential equations. The objective is to identify the conditions under which the chosen function ϕ enables our construction to provide a good approximation to the exact solution u of (3). Furthermore, we investigate whether minimizing the associated loss function effectively leads to a reduction in the approximation error between the predicted and exact solutions of (9).

Theorem 3. *Under assumptions of (3) and (4), and let I be a compact interval in the domain of u with $t_0 \in I$. Let ϕ be a function on I only vanishes at t_0, differentiable on t_0, and $\phi'(t_0) \neq 0$. Let*

$$N_\phi(t,\theta) := u_0 + \phi(t) \cdot N(t,\theta), \quad t \in \mathbb{R}. \tag{10}$$

Then N_ϕ is a good uniform approximation for the solution to (3).

Proof of Theorem 3. By the existence and uniqueness theorem, the solution u to (3) is continuously differentiable on its domain containing t_0. Now define a function v on I as follows:

$$\begin{cases} v(t) = \dfrac{u(t) - u_0}{\phi(t)} & \text{if } t \in I \setminus \{t_0\}, \\ v(t_0) = \frac{u'(t_0)}{\phi'(t_0)}. \end{cases}$$

It is clear that v is continuous on $I \setminus \{t_0\}$. At t_0, it is sufficient to observe that, under the assumptions of Theorem

$$v(t) = \frac{u(t) - u_0}{\phi(t)} = \frac{u(t) - u_0}{t - t_0} \times \frac{t - t_0}{\phi(t)}$$

it converges to $v(t_0)$ as t approaches t_0.

By the universal approximation theorem for 3-layer neural networks (Theorem 2), given $\varepsilon > 0$ there exist $n_1, n_2 \in \mathbb{N}$, $n_1 \times 1$, $n_2 \times n_1$, $1 \times n_2$ matrices A_1, A_2 and A_3, respectively, and $n_1 \times 1$, $n_2 \times 1$ and 1×1 matrices b_1, b_2, b_3, respectively, such that if $\theta = \theta(n_1, n_2, A_1, A_2, A_3, b_1, b_2, b_3)$, then

$$|N(t,\theta) - v(t)| < \frac{\varepsilon}{\alpha}, \quad \text{for all } t \in I,$$

where $\alpha = \max\{|\ \phi(t)\ |; t \in I\}$. Thus, we obtain that

$$u(t) = u_0 + \phi(t) \cdot v(t) \quad \text{for all } t \in I,$$

and

$$|N_\phi(t,\theta) - u(t)| = |\phi(t) \cdot (N(t,\theta) - v(t))| \leq \alpha \cdot |N(t,\theta) - v(t)| < \varepsilon \quad \text{for all } t \in I.$$

Thus, we can regard $N_\phi(\cdot,\theta)$ as a good approximation of u in the perspective of the universal approximation theorem. □

Example 1. *By application of Theorem 3, all function below are good uniform approximation for the solution u of the problem (3).*

$$N_1(t,\theta) := y_0 + (t-t_0)N(t,\theta), \quad t \in \mathbb{R}, \tag{11}$$

$$N_2(t,\theta) := y_0 + (1-e^{-(t-t_0)})N(t,\theta), \quad t \in \mathbb{R}, \tag{12}$$

$$N_3(t,\theta) := y_0 + \sin(t-t_0)N(t,\theta), \quad t \in \mathbb{R}, \tag{13}$$

$$N_4(t,\theta) := y_0 + \left(\frac{1}{1+e^{-(t-t_0)}} - \frac{1}{2}\right)N(t,\theta), \quad t \in \mathbb{R}, \tag{14}$$

$$N_5(t,\theta) := y_0 + \left(\ln\left(1+e^{t-t_0}\right) - \ln(2)\right)N(t,\theta), \quad t \in \mathbb{R}. \tag{15}$$

Given an interval $J = [a,b]$, we write $|J| = b-a$. Theorems 4 and 5 are mathematical results to derive error estimates for PINN II.

Theorem 4. *Let $(t_0,u_0) \in \mathbb{R}^2$, and suppose $G : [t_0-a_1,\ t_0+a_2] \times [u_0-b_1,\ u_0+b_2] \to \mathbb{R}$ is continuous in both arguments and Lipschitz continuous in u with Lipschitz constant $\kappa > 0$, and $M_0 > 0$.*

Let $u(t)$ denote the unique solution of the initial value problem (3), existing on an interval $I := [t_0-c_1,\ t_0+c_2]$ for some $c_1, c_2 > 0$.

Let $N_\phi(t;\theta)$ defined under the same conditions of Theorem 3.

Let

$$\mathcal{L}_\phi(\theta) := \int_I |N'_\phi(s,\theta) - G(s,N_\phi(s,\theta))|\,ds, \tag{16}$$

Then,

$$|N_\phi(t,\theta) - u(t)| \le e^{\kappa|t-t_0|}\mathcal{L}_\phi(\theta) \quad \textit{for all } t \in I.$$

Proof of Theorem 4. The proof is based on [25], using the triangle and the Grönwall's inequalities.

Under the stated assumptions, the error $\tau(t) := N_\phi(t;\theta) - u(t)$ satisfies $\tau(t_0) = 0$ and

$$\tau'(t) \;=\; N'_\phi(t;\theta) - u'(t) \;=\; N'_\phi(t;\theta) - G(t,N_\phi(t;\theta)) - \left(u'(t) - G(t,N_\phi(t;\theta))\right).$$

Using $u'(t) = G(t,u(t))$ and the Lipschitz continuity of G in its second argument, we obtain the differential inequality

$$\left|\tau'(t)\right| \;=\; \left|N'_\phi(t;\theta) - G(t,N_\phi(t;\theta)) - \left(G(t,u(t)) - G(t,N_\phi(t;\theta))\right)\right| \;\le\; \rho(t) + \kappa\,|\tau(t)|\,,$$

where

$$\rho(t,\theta) = |N'_\phi(t,\theta) - G(t,N_\phi(t,\theta))| \tag{17}$$

for all $t \in I$. By Grönwall's inequality (applied on $[t_0,\ t]$ or $[t,\ t_0]$ as appropriate), it follows that

$$|\tau(t)| \;\le\; e^{\kappa\,|t-t_0|}\int_I \rho(s,\theta)\,ds \;=\; e^{\kappa\,|t-t_0|}\,\mathcal{L}_\phi(\theta)\,,$$

for any $t \in I$, where we set $\mathcal{L}_\phi(\theta) := \int_I \rho(s,\theta)\,ds$. This $\mathcal{L}_\phi(\theta)$ represents the total residual loss over I. □

Theorem 5. *Under the same conditions of Theorems 1 and 2, let* $\{\xi_i\}_{i=1}^n$ *be an independent, identically distributed sample of n random points, each drawn from the uniform distribution on I. Then, for any* $t \in I$*, the following error estimate holds*

$$\left|N_\phi(t;\theta) - u(t)\right| \;\le\; |I|\, e^{\kappa\,|t-t_0|} \left(\mathbb{E}\Big[\frac{1}{n}\sum_{i=1}^{n} \big(N'_\phi(\xi_i;\theta) \;-\; G(\xi_i,\, N_\phi(\xi_i;\theta))\big)^2\Big]^{1/2} \;+\; \frac{\nu}{\sqrt{n}} \right),$$

where $\nu := \left(\frac{1}{|I|}\int_I \rho(s)^2\, ds \;-\; \left(\frac{1}{|I|}\int_I \rho(s)\, ds\right)^2 \right)^{1/2} \ge 0$ *is the standard deviation of the pointwise residual on I, and we define* $\rho(t,\theta)$ *as in (17), for* $t \in I$*.*

Proof of Theorem 5. Consider the n i.i.d. sample points $\xi_1, \ldots, \xi_n$ uniform on I. Define the random residual values $Y_i := \rho(\xi_i) = \left|N'_\phi(\xi_i;\theta) - G(\xi_i, N_\phi(\xi_i;\theta))\right|$ for $1 \le i \le n$, and let $\bar{Y} := \frac{1}{n}\sum_{i=1}^n Y_i$ be the sample average of the residual magnitudes. By construction, $\mathbb{E}[Y_i] = \frac{1}{|I|}\int_I \rho(s)\, ds = \frac{\mathcal{E}(\theta)}{|I|}$. Denoting $m := \mathbb{E}[\bar{Y}] = \mathbb{E}[Y_i]$ (the true mean residual) and $\sigma^2 := \mathrm{Var}(Y_i) = \frac{1}{|I|}\int_I \rho(s)^2 ds - \left(\frac{1}{|I|}\int_I \rho(s) ds\right)^2$ (so that $\sigma = \nu$ in the notation of the theorem), standard properties of independent sampling give $\mathrm{Var}(\bar{Y}) = \sigma^2/n$. Next, applying the triangle inequality in $L^2(\Omega)$ (with respect to the underlying probability space of the sample), we obtain

$$|m| \;=\; \left\| m \right\|_{L^2} \;\le\; \left\| \bar{Y} \right\|_{L^2} + \left\| \bar{Y} - m \right\|_{L^2} \;=\; \sqrt{\mathbb{E}[\bar{Y}^2]} \;+\; \frac{\sigma}{\sqrt{n}}\,.$$

Using Jensen's inequality (or the fact $\mathbb{E}[\bar{Y}^2] \le \mathbb{E}[Y_1^2]$ for $n \ge 1$), we further have $\sqrt{\mathbb{E}[\bar{Y}^2]} \le \sqrt{\mathbb{E}[Y_1^2]}$. But $\mathbb{E}[Y_1^2] = \frac{1}{|I|}\int_I \rho(s)^2 ds$. Combining these estimates yields

$$\mathbb{E}[Y_i] \;=\; m \;\le\; \frac{\sigma}{\sqrt{n}} \;+\; \sqrt{\frac{1}{|I|}\int_I \rho(s)^2 ds}\,.$$

Multiplying through by $|I|$, we conclude that

$$\mathcal{L}_\phi(\theta) \;=\; \int_I \rho(s)\, ds \;\le\; |I| \left(\sqrt{\frac{1}{|I|}\int_I \rho(s)^2 ds} \;+\; \frac{\sigma}{\sqrt{n}} \right).$$

Finally, substituting this bound on $\mathcal{L}_\phi(\theta)$ back into the earlier error estimate for $|e(t)|$ gives

$$|N_\phi(t;\theta) - u(t)| \;\le\; e^{\kappa\,|t-t_0|}\, \mathcal{L}_\phi(\theta) \;\le\; |I|\, e^{\kappa\,|t-t_0|} \left(\sqrt{\frac{1}{|I|}\int_I \rho(s)^2 ds} \;+\; \frac{\sigma}{\sqrt{n}} \right),$$

which is exactly the stated inequality (since $\mathbb{E}\big[\frac{1}{n}\sum_{i=1}^n (N'_\phi(\xi_i;\theta) - G(\xi_i, N_\phi(\xi_i;\theta)))^2\big] = \frac{1}{|I|}\int_I \rho(s)^2 ds$). This completes the proof. □

This theoretical result illustrates that minimizing the PINN's training loss directly leads to a small error in the solution. In particular, the bound shows that if the residual $R'(t;\theta) - G(t, R(t;\theta))$ (which is what the loss function penalizes) is small in the mean-square sense, then the network output $R(t;\theta)$ stays uniformly close to the true solution $u(t)$ on I (up to the stability factor $e^{\kappa|t-t_0|}$ arising from the ODE's dynamics). In essence, a smaller training loss (residual error) guarantees a smaller actual solution error, corroborating the fundamental principle that PINN training drives the approximate solution

towards the correct solution. This kind of error estimate is important as it provides rigorous assurance that the PINN's learned solution will converge to the true solution as the training loss is minimized (and as $n \to \infty$, the sampling error term $\nu/\sqrt{n}$ becomes negligible). corroborating the fundamental principle that PINN training drives the approximate solution towards the correct solution. This kind of error estimate is important as it provides rigorous assurance that the PINN's learned solution will converge to the true solution as the training loss is minimized (and as $n \to \infty$, the sampling error term $\nu/\sqrt{n}$ becomes negligible).

5. Experiment Results

In this section, we provide a numerical validation of the derived error estimates by examining the relationship between the approximation error and the training loss functions. This analysis is conducted within the framework of our proposed PINN construction, applied to various differential equations. The experiments are performed using different trial functions, as specified in examples.

The neural network employed in this study follows a fully connected architecture composed of three layers: an input layer, two hidden layers with 16 and 32 neurons, respectively, and a single-node output layer. The activation function used in the hidden layers is the hyperbolic tangent (`tanh`), which is known for its smoothness and effectiveness in approximating nonlinear functions. The network is trained using the Adam optimizer [30], with a fixed learning rate set to 0.01. The training process is performed over 1000 epochs, utilizing 10,000 uniformly distributed collocation points within the domain $[0, 1]$. The trial solution is constructed to satisfy the initial condition by design, and the loss function is defined as the mean squared residual of the differential equation evaluated at the training points. The training aims to minimize this physics-informed loss, thereby guiding the network to approximate the true solution of the initial value problem.

5.1. Example 1

Consider the following initial value problem:

$$\frac{dy}{dt} = 1.27\, y(1-y), \quad t \in [0,1], \quad y(0) = 0.67.$$

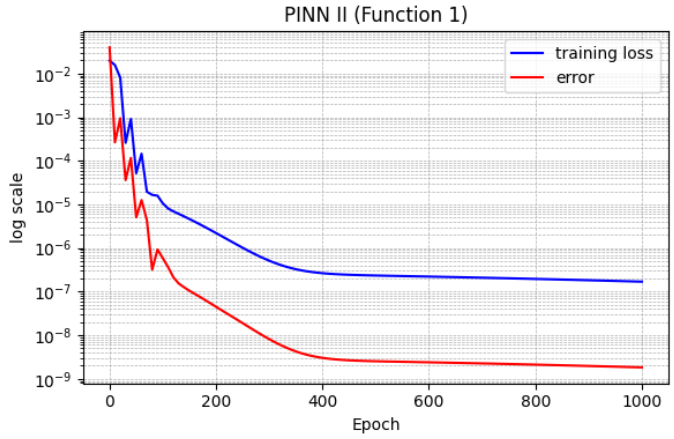
The exact solution is given by

$$y(t) = \frac{1}{1 + \frac{33}{67} e^{-1.27t}}, \quad t \in [0,1].$$

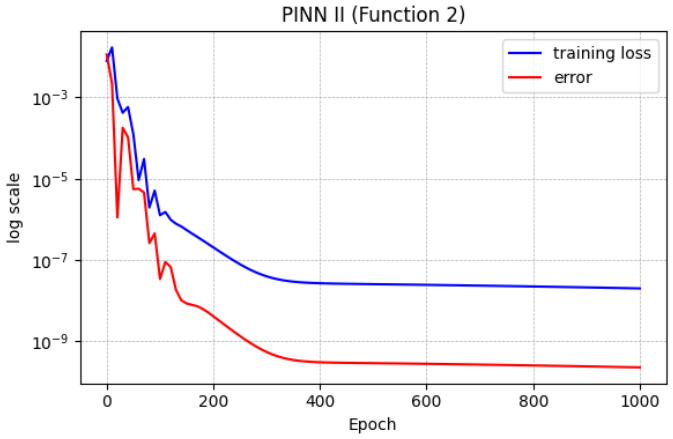
In Figure 1, we illustrate both the approximation error and the training loss obtained during the PINN training process. This visualization helps to assess the consistency between the empirical loss minimization and the actual error behavior.

In this example, we tested five different functions to construct trial solutions, each defined explicitly in Table 1. Figure 1 presents the training loss and approximation error corresponding to these trial solutions, shown in the same order as in the table.

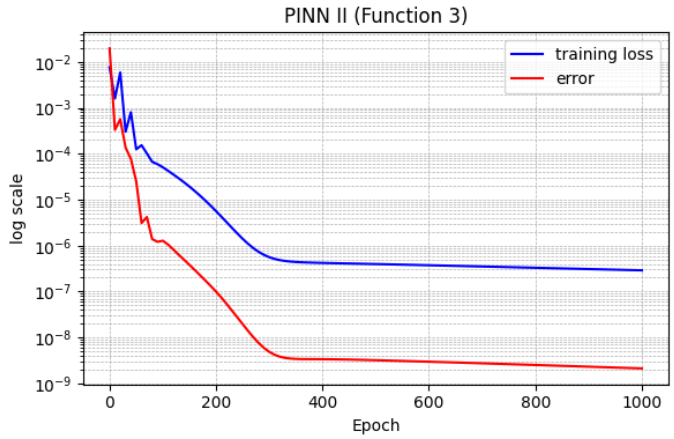
We now consider two constructions of the trial solution in the form (10). In each example of Table 2, the function ϕ is deliberately chosen so that it does not satisfy one of the assumptions of Theorem 3. This approach is intended to demonstrate the necessity of the stated conditions. For each case, we provide the plots of both the trial and exact solutions in order to visually compare the results (see Figure 2).



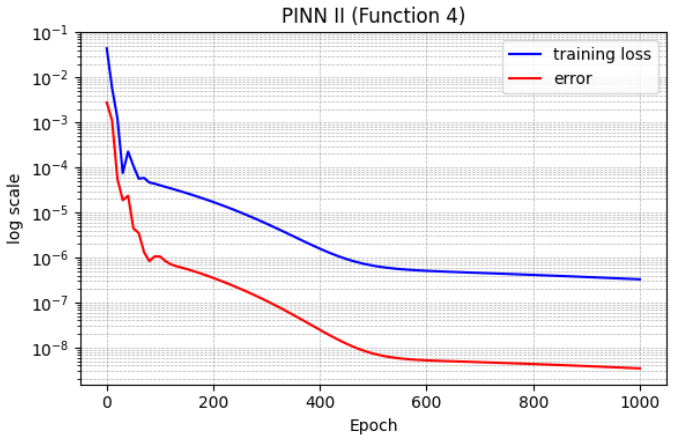
(**a**)



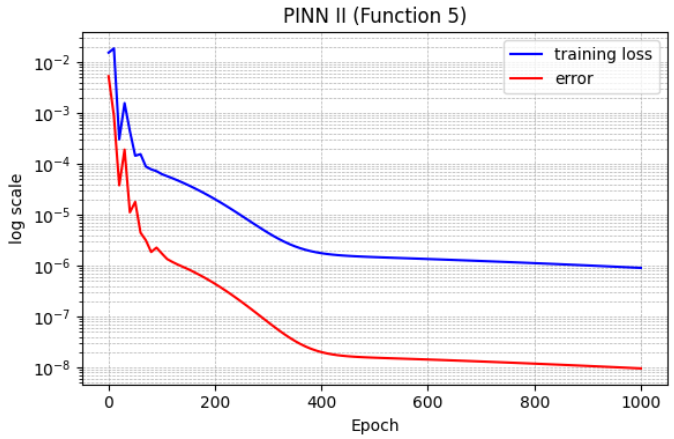
(**b**)



(**c**)



(**d**)



(**e**)

Figure 1. (**a**) Training loss and prediction error corresponding to the first trial solution listed in Table 1. (**b**) Training loss and prediction error corresponding to the second trial solution listed in Table 1. (**c**) Training loss and prediction error corresponding to the third trial solution listed in Table 1. (**d**) Training loss and prediction error corresponding to the fourth trial solution listed in Table 1. (**e**) Training loss and prediction error corresponding to the fifth trial solution listed in Table 1.

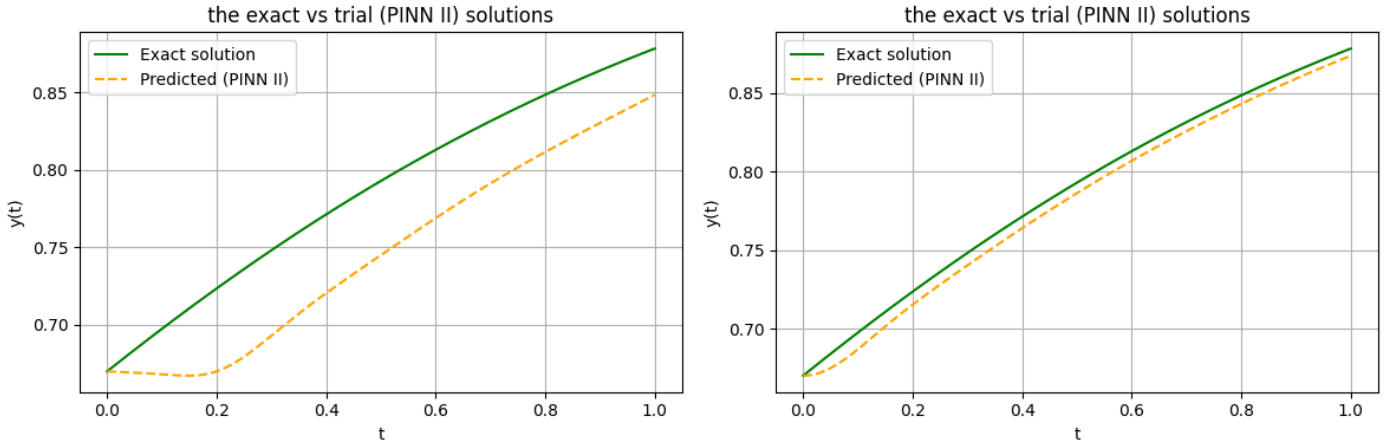
Table 1. Table of several trial solution construction.

Function	Trial Solution Formula	Figure	Max Absolute Error L_∞
1 *	$y(0) + tN(t,\theta)$	Figure 1a	6.62×10^{-5}
2	$y(0) + (1 - e^{-t})N(t,\theta)$	Figure 1b	2.42×10^{-5}
3	$y(0) + \sin(t)N(t,\theta)$	Figure 1c	8.66×10^{-5}
4	$y(0) + \left(\frac{1}{1+e^{-(t)}} - \frac{1}{2}\right)N(t,\theta)$	Figure 1d	8.78×10^{-5}
5	$y(0) + \left(\ln(1+e^{t}) - \ln(2)\right)N(t,\theta)$	Figure 1e	1.48×10^{-4}

* The standard polynomial construction.

Table 2. Table of several trial solution construction.

Function	Trial Solution Formula	Figure	Max Absolute Error L_∞
1	$y(0) + t(t-0.2)N(t,\theta)$	Figure 2a	5.66×10^{-2}
2	$y(0) + t^2N(t,\theta)$	Figure 2b	1.07×10^{-2}



(**a**) (**b**)

Figure 2. (**a**) The exact and predicted solutions for the first trial solution construction in Table 2. (**b**) The exact and predicted solutions for the second trial solution construction in Table 2.

We observe that in Example 1 (the logistic ODE case), all five trial solutions that satisfy the assumptions of Theorem 3 produce highly accurate approximations. In these admissible cases, where $\phi(t_0) = 0$ and $\phi'(t_0) \neq 0$, the maximum absolute error L_∞ remains on the order of 10^{-5}. Table 1 reports the corresponding values, with the smallest and largest errors being 2.42×10^{-5} and 1.48×10^{-4}, respectively. Figure 1a–e confirms this observation, showing that the training loss decreases rapidly and reaches very small values. In contrast, when non-admissible trial functions are employed (as in Table 2), the error increases by at least two orders of magnitude. For example, selecting $\phi(t) = t^2$ (which violates $\phi'(t_0) \neq 0$) results in $L_\infty \approx 1.07 \times 10^{-2}$, while $\phi(t) = (t - t_0)(t - 0.2)$ leads to $L_\infty \approx 5.66 \times 10^{-2}$. These outcomes are illustrated in Figure 2a,b, where a clear divergence between the predicted and exact solutions is visible. Overall, these results highlight that only the trial solution satisfying the theoretical constraints (a function ϕ vanishing solely at t_0 with a nonzero derivative there) yields the near-perfect approximations reported in Table 1.

5.2. Example 2

Consider the following initial value problem:

$$\frac{dy}{dt} = \frac{4t^3 + 3t^2 + 6t + 2}{2y + 4}, \quad t \in [0,1], \quad y(0) = 0.$$

The exact solution is given by

$$y(t) = -2 + \sqrt{t^4 + t^3 + 3t^2 + 2t + 4}, \quad t \in [0, 1].$$

The following table presents several distinct constructions as described in Theorem 3. These constructions are illustrated in the accompanying figure which are arranged in Table 3.

From Figures 3–7, we illustrate both the approximation error and the training loss obtained during the PINN training process. In addition the exact and the predicted solutions are traced to give a simple visual comparison.

As in the first example, we give now in Table 4 some choices of the function ϕ that do not satisfy a condition for Theorem 3. We plot the predicted solution in each case with the exact solution to give the comparison.

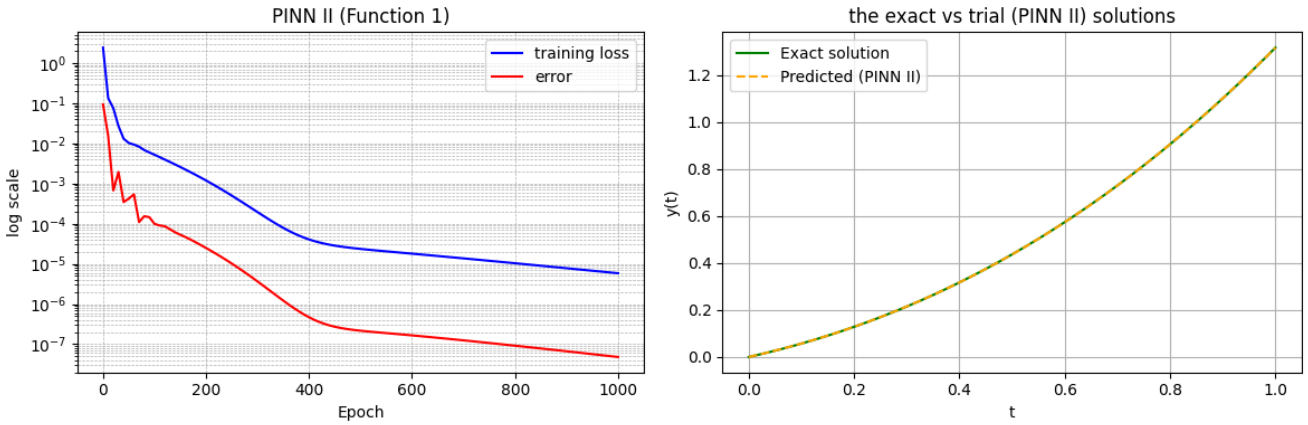
Table 3. Table of several trial solution construction.

Function	Trial Solution Formula	Figure	Max Absolute Error L_∞
1 *	$y(0) + (t - t_0)N(t, \theta)$	Figure 3	3.73×10^{-4}
2	$y(0) + \log(1 + t - t_0)N(t, \theta)$	Figure 4	4.09×10^{-4}
3	$y(0) + \sin(t - t_0)N(t, \theta)$	Figure 5	4.06×10^{-4}
4	$y(0) + \tan(t - t_0)N(t, \theta)$	Figure 6	9.10×10^{-4}
5	$y(0) + (1 - e^{(t-t_0)})N(t, \theta)$	Figure 7	7.86×10^{-4}

* The standard polynomial construction.

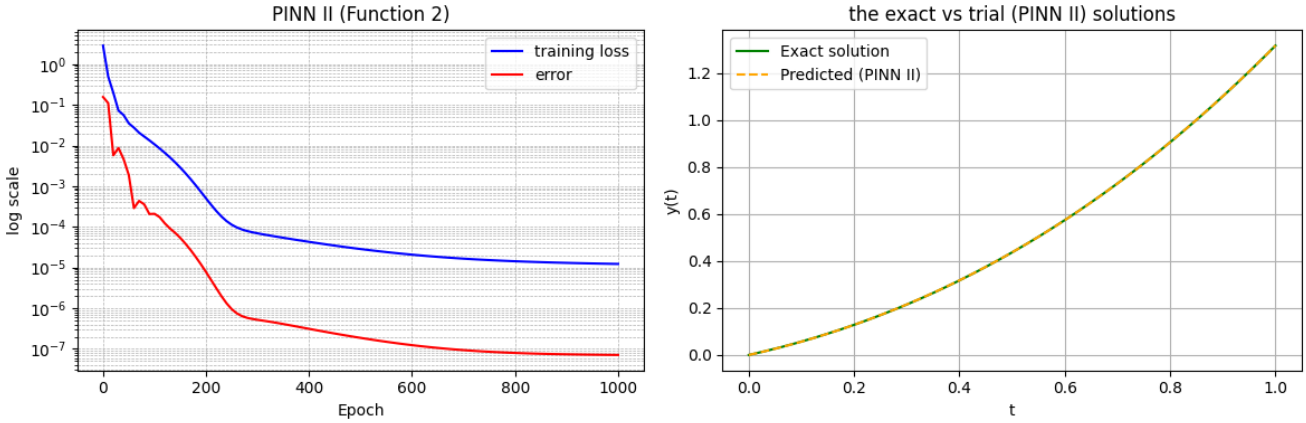
Table 4. Table of several trial solution construction.

Function	Trial Solution Formula	Figure	Max Absolute Error L_∞
1	$y(0) + \cos(t - t_0)N(t, \theta)$	Figure 8a	1.34×10^{0}
2	$y(0) + (1 - \exp(t - t_0))^2 N(t, \theta)$	Figure 8b	3.71×10^{-2}
3	$y(0) + (t - t_0)(t - 0.5)N(t, \theta)$	Figure 8c	4.38×10^{-1}



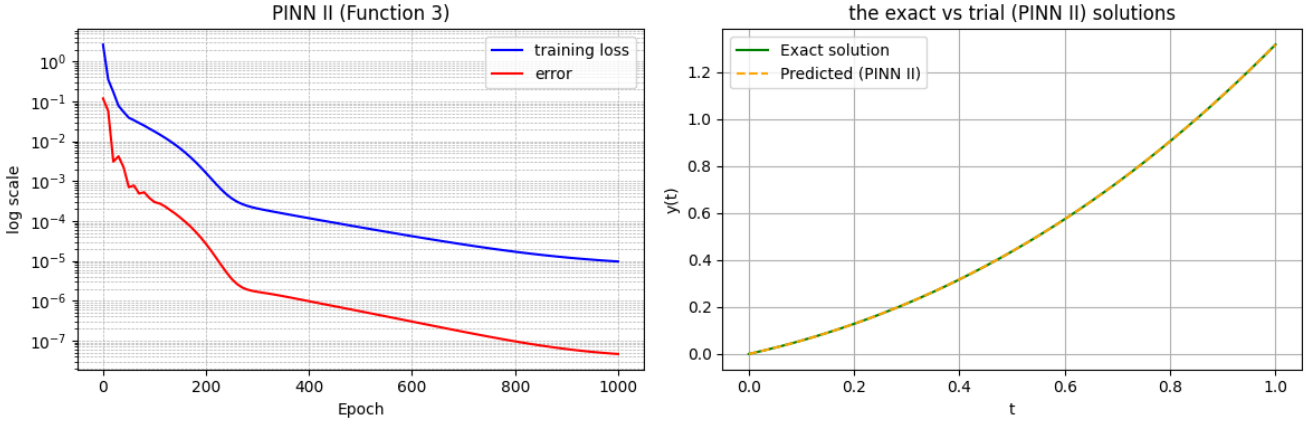
(**a**) (**b**)

Figure 3. (**a**) The training loss and error for the first trial solutions in Table 3; (**b**) the curves of the exact and predicted solutions.



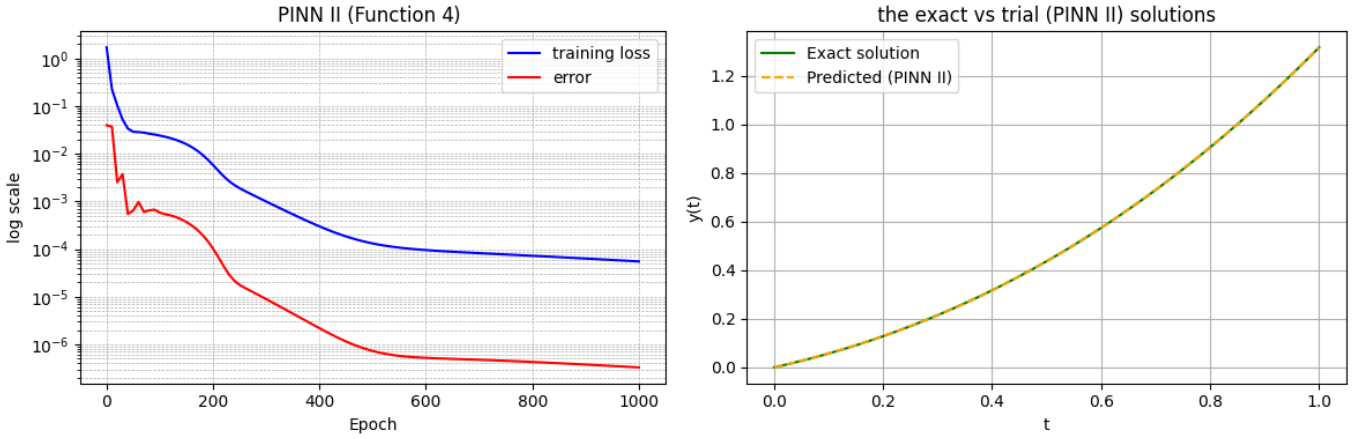
(**a**) (**b**)

Figure 4. (**a**) The training loss and error for the second trial solution in Table 3; (**b**) the curves of the exact and predicted solutions.



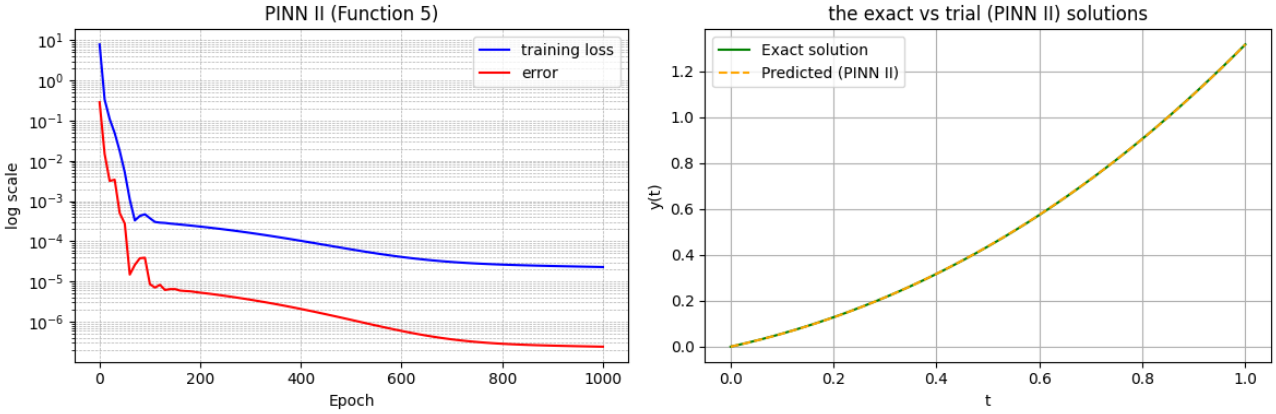
(**a**) (**b**)

Figure 5. (**a**) The training loss and error for the third trial solution in Table 3; (**b**) the curves of the exact and predicted solutions.



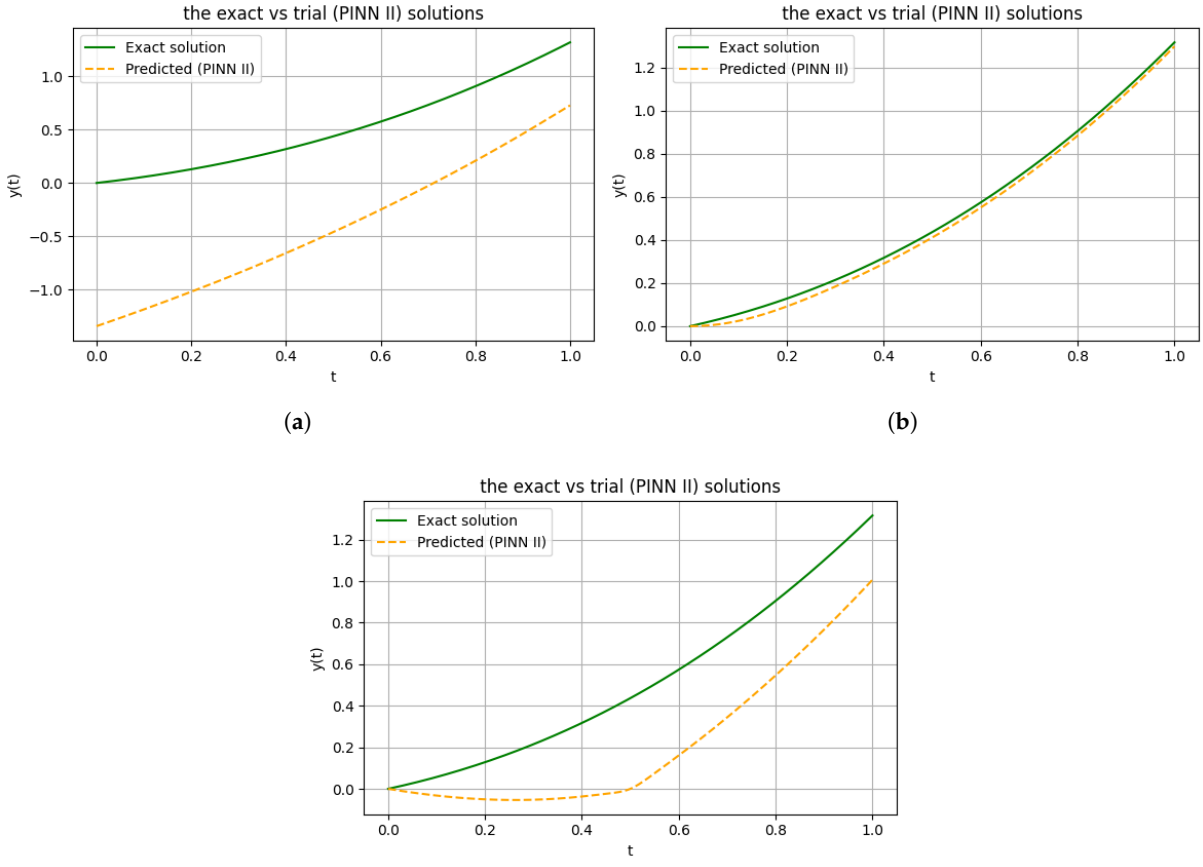
(**a**) (**b**)

Figure 6. (**a**) The training loss and error for the fourth trial solution in Table 3; (**b**) the curves of the exact and predicted solutions.



(**a**) (**b**)

Figure 7. (**a**) The training loss and error for the fifth trial solution in Table 3; (**b**) the curves of the exact and predicted solutions.



(**a**) (**b**)

(**c**)

Figure 8. (**a**) The exact and predicted solutions for the first trial solution construction in Table 4. (**b**) The exact and predicted solutions for the second trial solution construction in Table 4. (**c**) The exact and predicted solutions for the third trial solution construction in Table 2.

In Example 2 the same pattern holds. All admissible choices of ϕ (those meeting $\phi(t_0) = 0$, $\phi'(t_0) \neq 0$) again produce very small errors (on the order of 10^{-4}). Table 3 lists the five valid trial functions and their errors: for instance, $\phi(t) = t - t_0$, $\log(1 + t - t_0)$, and $\sin(t - t_0)$ yield $L_\infty \approx 3.73 \times 10^{-4}$, 4.09×10^{-4}, and 4.06×10^{-4}, respectively, while $\phi(t) = \tan(t - t_0)$ and $1 - e^{-(t-t_0)}$ both remain below 10^{-4}. In each case the predicted solution curve overlaps almost exactly with the exact solution (Figures 3–7). In contrast,

violating the construction assumptions causes large errors. Table 4 shows that using $\phi(t) = \cos(t - t_0)$ (so that $\phi(t_0) \neq 0$) gives $L_\infty \approx 1.34$, and $\phi(t) = (t - t_0)(t - 0.5)$ gives $L_\infty \approx 4.38 \times 10^{-1}$. These errors (two to three orders of magnitude larger) are evident in Figure 8, where the predicted and exact curves clearly diverge. Thus, as in Example 1, only the trial solutions satisfying the theoretical conditions (Table 3) achieve high accuracy.

Both examples demonstrate that the theoretical requirements on the trial function ϕ are decisive for PINN accuracy. When $\phi(t_0) = 0$, ϕ is differentiable and $\phi'(t_0) \neq 0$, training converges efficiently with very small errors ($L_\infty \sim 10^{-4}$). In contrast, ignoring these assumptions causes errors to grow by orders of magnitude (often into the 10^{-1}–10^{0} range) and yields slow or unstable convergence. These numerical findings (supported by Tables 1–4 and Figures 1–8 confirm the paper's theoretical framework and underscore the critical importance of constructing the trial solution to satisfy the stated conditions.

5.3. Example 3

To probe a more challenging regime, we consider the linear ODE with high-frequency forcing

$$\frac{dy}{dt} + y = \cos(10\pi t), \quad t \in [0, 1], \quad y(0) = 0.$$

The exact solution is given by:

$$y(t) = \frac{10\pi \sin(10\pi t) + \cos(10\pi t)}{1 + 100\pi^2} - \frac{e^{-t}}{1 + 100\pi^2}$$

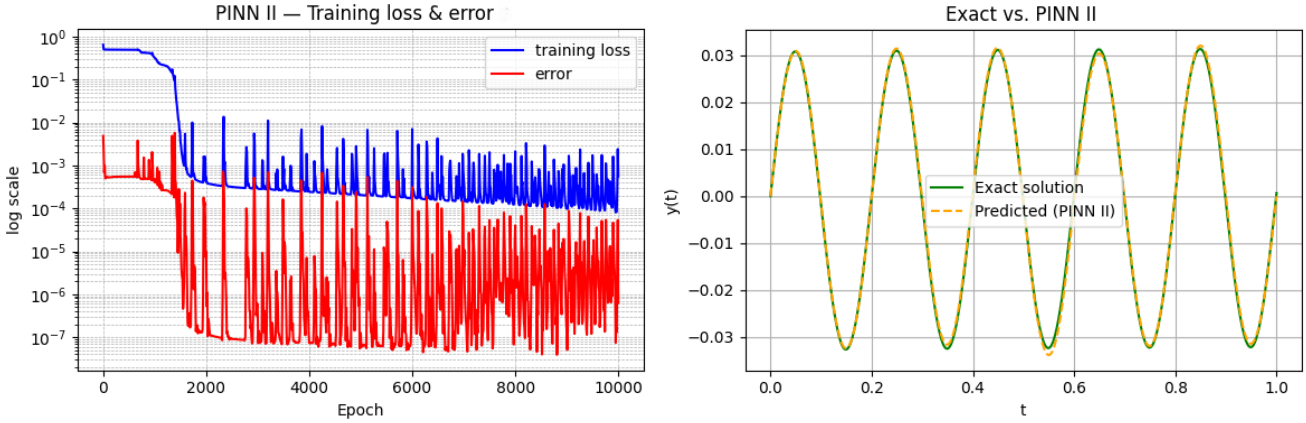
In this setting we intentionally restrict the collocation set to 70 points and allow up to 10,000 training epochs so as to stress-test convergence under stricter sampling and training budgets. We then evaluate our hard-constrained formulation (PINN II) with several *admissible* trial functions ϕ (satisfying $\phi(t_0) = 0$ and $\phi'(t_0) \neq 0$) alongside *non-admissible* choices that violate these conditions, thereby directly testing the hypothesis of Theorem 3.

Table 5 and Figures 9–15 show a clear pattern: admissible PINN II trials reach sub-millipercent accuracy (with L_∞ typically in the 10^{-4}–10^{-3} range) and produce solution curves that closely track the exact response, while non-admissible constructions (e.g., $\phi(t) = \cos(t - t_0)$ or squared/exponentially–flattened variants) remain orders of magnitude less accurate within the same budget and exhibit visible deviations in the plots.

Table 5. Table of several trial solution construction.

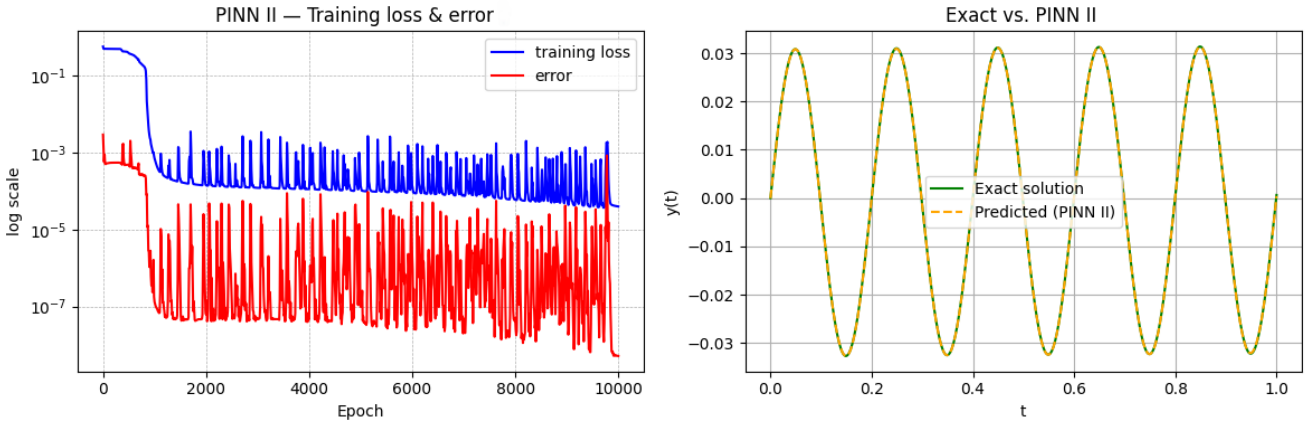
Function	Trial Solution Formula	Figure	Max Absolute Error L_∞
1	$y(0) + (t - t_0)N(t, \theta)$	Figure 9	2.05×10^{-3}
2	$y(0) + \log(1 + t - t_0)N(t, \theta)$	Figure 10	3.09×10^{-4}
3	$y(0) + \sin(t - t_0)N(t, \theta)$	Figure 11	4.76×10^{-4}
4	$y(0) + \frac{1}{1+\exp(-(t-t_0))}N(t, \theta)$	Figure 12	2.16×10^{-3}
5	$y(0) + (1 - e^{(t-t_0)})N(t, \theta)$	Figure 13	2.84×10^{-3}
6 *	$y(0) + \cos(t - t_0)N(t, \theta)$	Figure 14	3.53×10^{0}
7 *	$y(0) + (t - t_0)^2 N(t, \theta)$	Figure 15	1.20×10^{-2}

* In this case, one of the assumptions required by Theorem 3 is not satisfied.



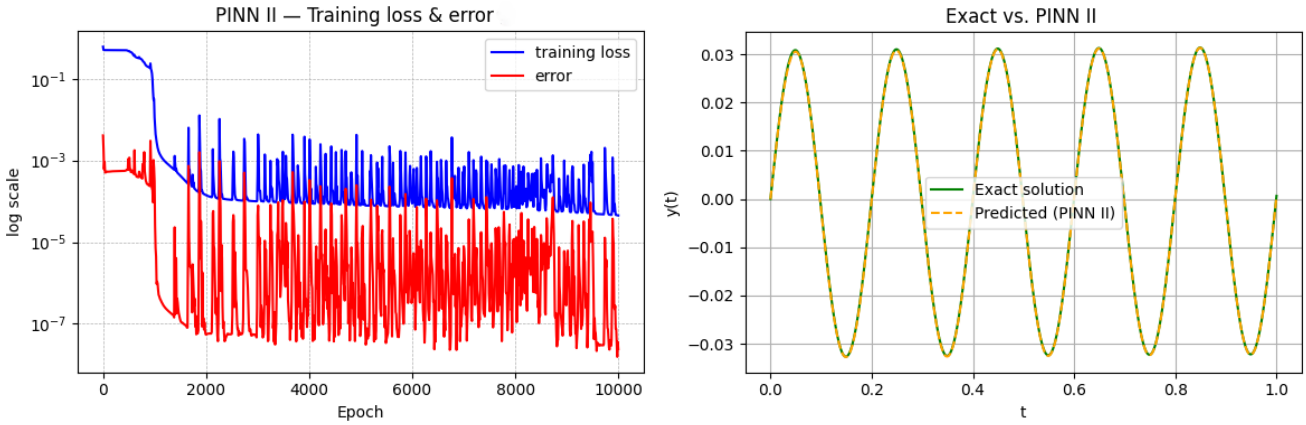
(**a**) (**b**)

Figure 9. (**a**) The training loss and error for the first trial solutions in Table 5; (**b**) the curves of the exact and predicted solutions.



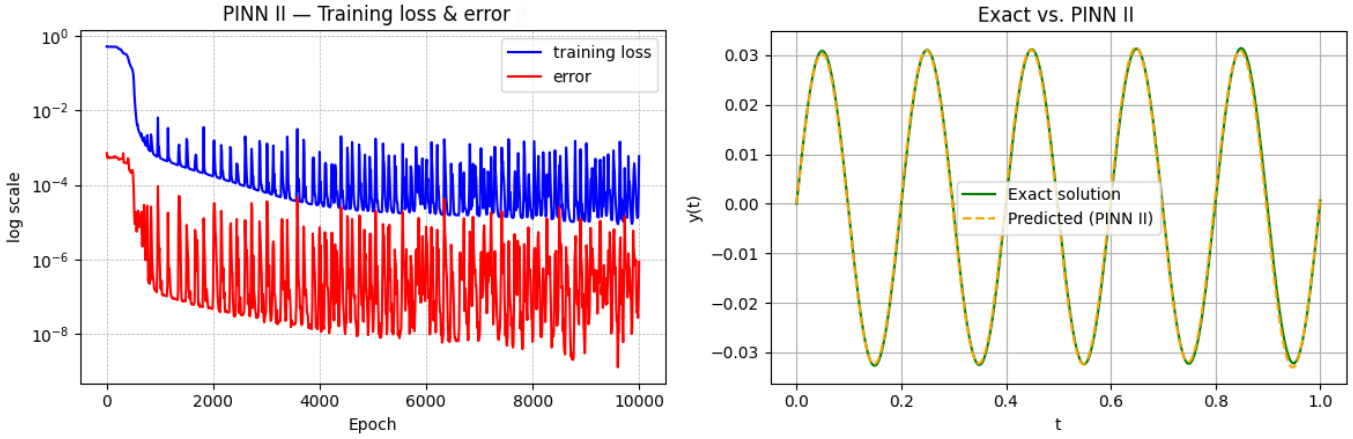
(**a**) (**b**)

Figure 10. (**a**) The training loss and error for the second trial solution in Table 5; (**b**) the curves of the exact and predicted solutions.



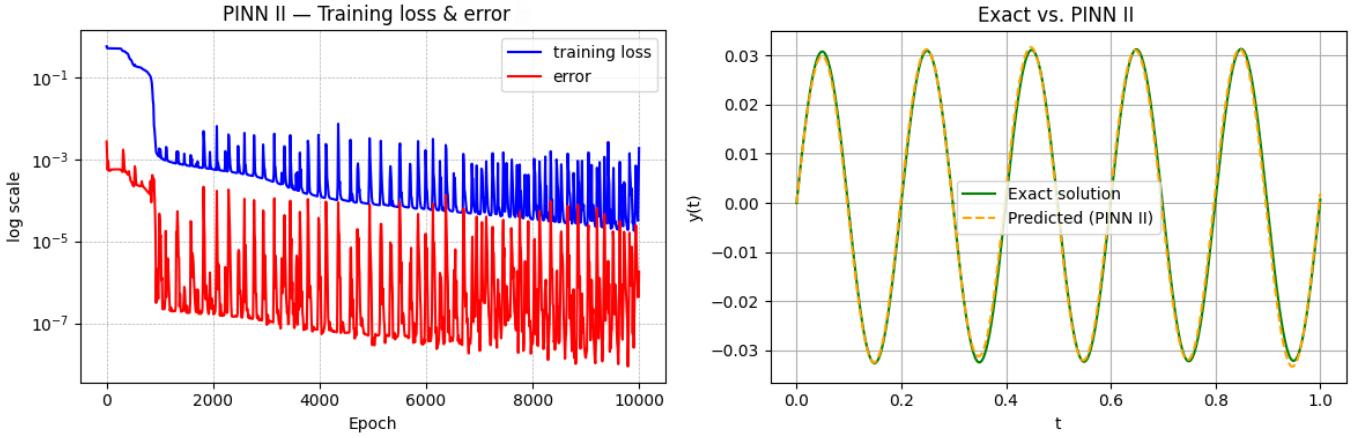
(**a**) (**b**)

Figure 11. (**a**) The training loss and error for the third trial solution in Table 5; (**b**) the curves of the exact and predicted solutions.



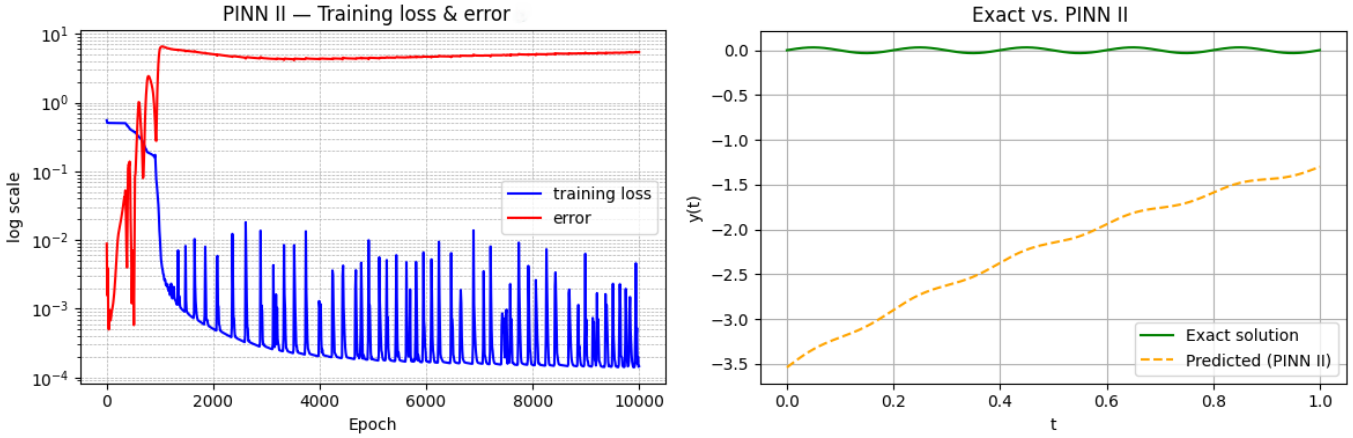
(**a**) (**b**)

Figure 12. (**a**) The training loss and error for the fourth trial solution in Table 5; (**b**) the curves of the exact and predicted solutions.



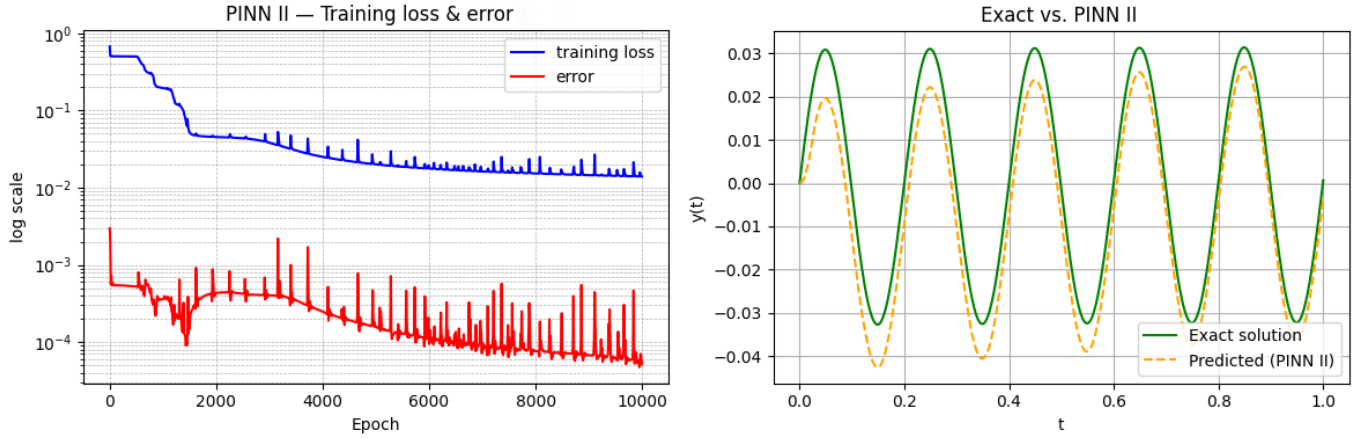
(**a**) (**b**)

Figure 13. (**a**) The training loss and error for the fifth trial solution in Table 5; (**b**) the curves of the exact and predicted solutions.



(**a**) (**b**)

Figure 14. (**a**) The training loss and error for the sixth trial solution in Table 5; (**b**) the curves of the exact and predicted solutions.



(**a**) (**b**)

Figure 15. (**a**) The training loss and error for the seventh trial solution in Table 5; (**b**) the curves of the exact and predicted solutions.

These results reinforce the necessity of the admissibility conditions in Theorem 3 and mirror the trends observed in the earlier examples: when ϕ is well-posed, the hard-constrained trial drives faster and steadier error decay; when ϕ violates admissibility, convergence degrades substantially despite identical training protocols.

5.4. Comparison with the Vanilla PINN I Approach

We compare our hard-constrained formulation (PINN II) against the standard soft-constraint approach (PINN I) under a common protocol: identical network $[1; 16; 32; 1]$ and adam-optimizer, the same collocation (70 points for Examples 1 and 2; 100 for Example 4), and an early-stopping target on the test grid ($L_\infty \leq 7 \times 10^{-5}$ for Example 1 and $L_\infty \leq 7 \times 10^{-4}$ for Example 2), otherwise a cap of 10,000 epochs. Errors and stopping criteria are computed on an independent uniform test grid (e.g., 2001 points on $[0, 1]$), disjoint from the collocation points.

For PINN I we sweep the initial-condition penalty weight $\alpha_{\mathrm{IC}} \in \{0.1, 1, 10, 100\}$, while for PINN II we test admissible and non-admissible trial functions ϕ. Tables 6 and 7 report the number of epochs required to meet the accuracy target, and Figures 16–18 display the corresponding L_∞ trajectories.

Table 6. Epochs required to reach $L_\infty \leq 7 \times 10^{-5}$: PINN-I (with various initial-condition penalty weight α_{IC}) vs. PINN-II (using different admissible/non-admissible trial function ϕ) on Example 1.

Method	Variant	Number of Epoch
PINN I	$\alpha_{IC} = 0.1$	700
PINN I	$\alpha_{IC} = 1$	1300
PINN I	$\alpha_{IC} = 10$	1700
PINN I	$\alpha_{IC} = 100$	3600
PINN II	$\phi(t) = t$	1100
PINN II	$\phi(t) = \exp(t) - 1$	1200
PINN II	$\phi(t) = \frac{1}{1+e^{-t}} - \frac{1}{2}$	600
PINN II	$\phi(t) = \ln(1 + t)$	500
* PINN II	$\phi(t) = \cos(t)$	10,000
* PINN II	$\phi(t) = t^2$	10,000

* Non admissible trial function.

Table 7. Epochs required to reach $L_\infty \leq 7 \times 10^{-4}$: PINN-I (with various initial-condition penalty weight α_{IC}) vs. PINN-II (using different admissible/non-admissible trial function ϕ) on Example 2.

Method	Variant	Number of Epoch
PINN I	$\alpha_{IC} = 0.1$	1600
PINN I	$\alpha_{IC} = 1$	1100
PINN I	$\alpha_{IC} = 10$	1800
PINN I	$\alpha_{IC} = 100$	6200
PINN II	$\phi(t) = t$	700
PINN II	$\phi(t) = \exp(t) - 1$	700
PINN II	$\phi(t) = \sin(t)$	800
PINN II	$\phi(t) = \frac{1}{1+e^{-t}} - \frac{1}{2}$	800
PINN II	$\phi(t) = \ln(1+t)$	1100
* PINN II	$\phi(t) = \cos(t)$	10,000
* PINN II	$\phi(t) = t^2$	10,000

* Non admissible trial function.

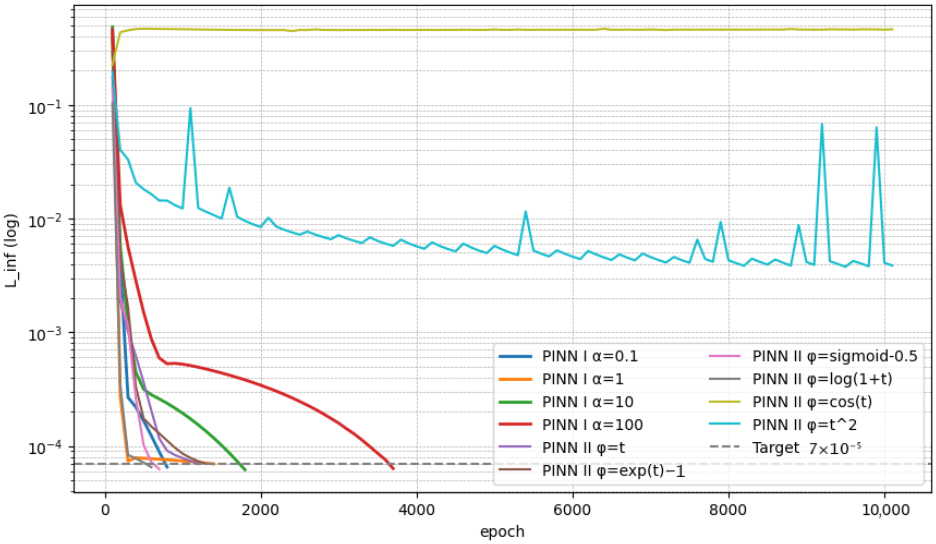


Figure 16. Effect of the IC penalty weight α_{IC} (PINN-I) and of the trial functions ϕ (PINN-II) on the training error L_∞ for Example 1; curves sampled every 100 epochs; dashed line denotes the target 7×10^{-5}.

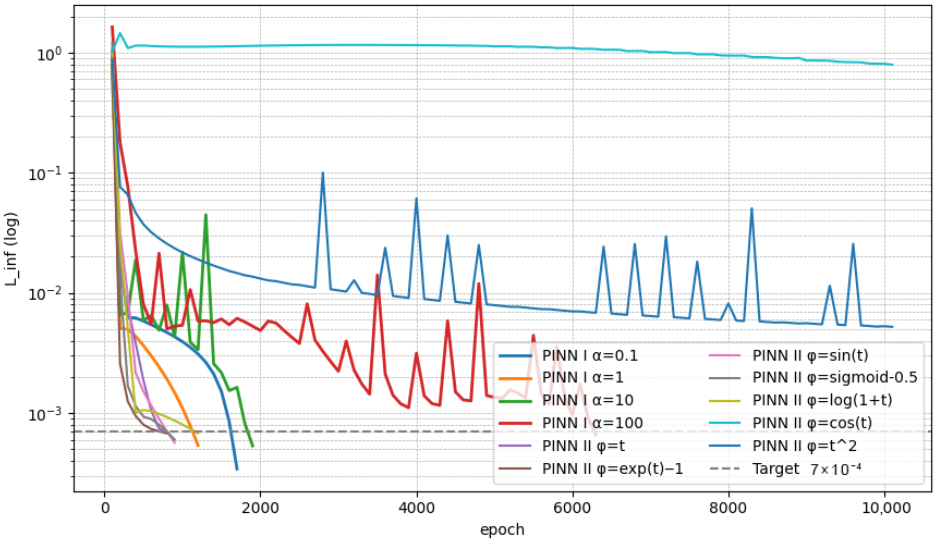


Figure 17. Effect of the IC penalty weight α_{IC} (PINN-I) and of the trial functions ϕ (PINN-II) on the training error L_∞ for Example 2; curves sampled every 100 epochs; dashed line denotes the target 7×10^{-4}.

Across both examples, the data reveal a consistent pattern. In Example 1 (Table 6), admissible PINN II trials achieve the target typically within 500–1200 epochs (e.g., $\phi(t) = \log(1+t)$ at 500 and $\phi(t) = \sigma(t) - \frac{1}{2}$ at 600), whereas PINN I is sensitive to the penalty weight, ranging from 700 epochs ($\alpha_{\mathrm{IC}} = 0.1$) to 3600 epochs ($\alpha_{\mathrm{IC}} = 100$). In Example 2 (Table 7), admissible PINN II trials again reach the target in 700–1100 epochs, while PINN I requires 1100–6200 epochs depending on α_{IC}. Non-admissible choices (e.g., $\phi(t) = \cos t$ or t^2) systematically fail to meet the threshold within 10,000 epochs, underscoring the necessity of the admissibility conditions. The L_∞-vs-epoch curves in Figures 16 and 17 corroborate these trends: admissible PINN II exhibits faster and steadier descent to the dashed target line, whereas PINN I shows slower or oscillatory convergence as α_{IC} varies.

Example 4

To probe robustness in a stiff, oscillatory regime, we consider a high-frequency linear ODE and compare the hard-constrained formulation (PINN-II) with the vanilla soft-constraint baseline (PINN-I) (see Figure 18). Anticipating the theory, admissible trial functions in PINN-II track the exact dynamics (see Figure 19) and reach small L_∞ errors markedly faster, whereas PINN-I is sensitive to the penalty α_{IC} and non-admissible choices degrade accuracy.

The differential equation is:

$$y' + 100y = \cos(10\pi t), \quad y(0) = 0,$$

with the exact solution:

$$y(t) = \frac{-100e^{-100t} + 100\cos(10\pi t) + 10\pi \sin(10\pi t)}{10000 + 100\pi^2}$$

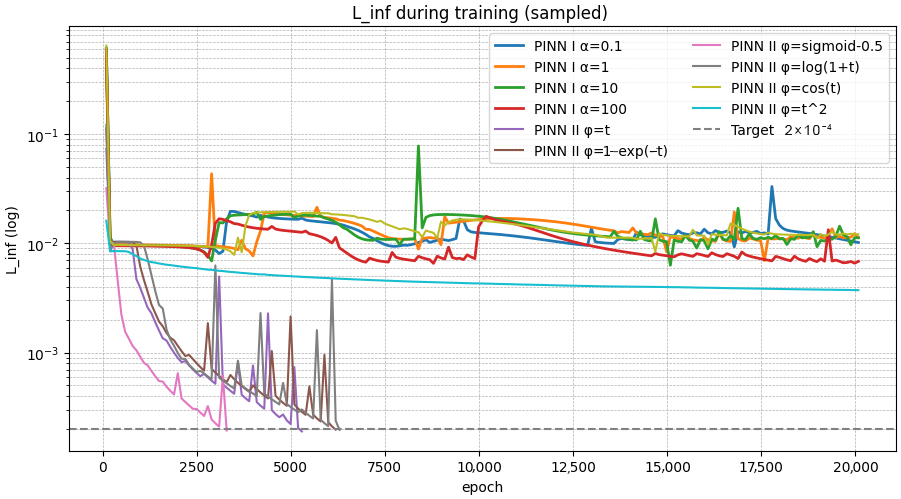


Figure 18. Effect of the IC penalty weight α_{IC} (PINN-I) and of the trial functions ϕ (PINN-II) on the training error L_∞ for Example 4; curves sampled every 100 epochs; dashed line denotes the target 2×10^{-4}.

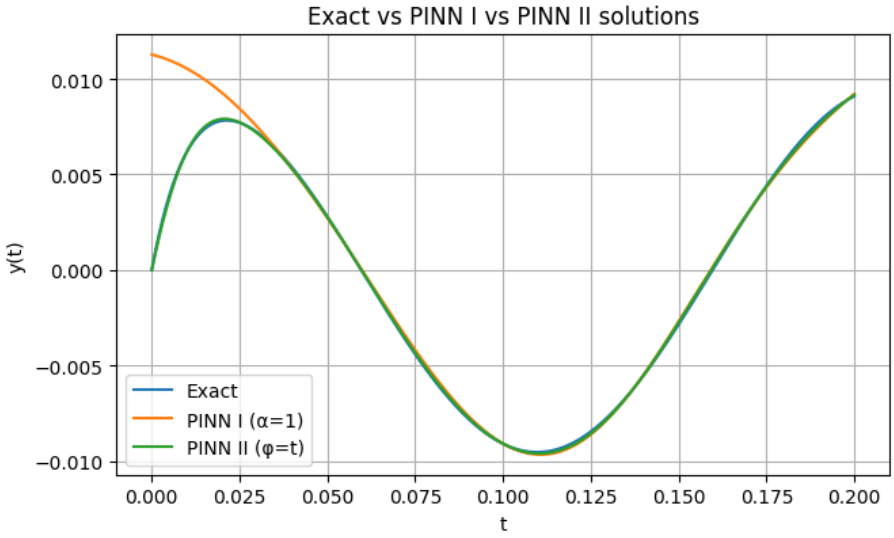


Figure 19. Exact vs PINN-I (with $\alpha_{IC} = 1$) and PINN-II (using $\phi(t) = t$) solutions for the stiff ODE in Example 4.

6. Conclusions

In conclusion, we have developed a generalized hard-constrained trial solution strategy for physics-informed neural networks (PINNs) and demonstrated its effectiveness through both theoretical analysis and comprehensive experiments. The main theoretical contribution is a general framework for constructing admissible trial functions that exactly satisfy prescribed initial or boundary conditions by design. We proved that these hard-constrained trial functions preserve the universal approximation capability of PINNs, and we established error bounds showing that the solution error (in the L^2 sense) is proportional to the PINN training loss under mild assumptions, providing rigorous performance guarantees. This theoretical foundation is reinforced by extensive numerical results: the proposed approach exhibits significantly faster convergence and improved training stability compared to the standard "vanilla" PINN formulation (PINN-I) that uses soft constraints. Across multiple first-order ODE examples—including a challenging stiff oscillatory case—our method consistently achieved high accuracy and robust performance for various choices of trial function ϕ. By contrast, the vanilla PINN converged more slowly and showed sensitivity to penalty weight tuning. In direct comparisons, the hard-constrained PINN reached target error thresholds with far fewer epochs and maintained a steadier error decay, highlighting its advantages in both efficiency and error control. We acknowledge that the present study is limited to first-order scalar ODEs, and our theoretical results apply to problems with Lipschitz-continuous right-hand sides. Extensions to non-smooth or weak-solution settings, and systematic studies of other hyperparameters constitute promising directions for future work. Looking ahead, we plan to extend this framework to more complex settings, such as second-order differential equations and partial differential equations, to assess its broader applicability. Additionally, combining the hard-constraint approach with other PINN enhancements (for instance, adaptive loss-weighting schemes or domain decomposition methods) is a promising direction for further improving convergence and tackling even more challenging physics-informed modeling problems.

Author Contributions: Conceptualization, A.B. and I.J.; methodology, A.B.; software, A.B.; validation, A.B., I.J. and J.A.R.; formal analysis, A.B.; investigation, I.J. and J.A.R.; resources, I.J. and J.A.R.; data curation, A.B.; writing—original draft preparation, A.B.; writing—review and editing, A.B., I.J. and J.A.R.; visualization, A.B.; supervision, I.J. and J.A.R.; project administration, I.J. and J.A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data supporting the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Coddington, E.A. *An Introduction to Ordinary Differential Equations*; Prentice-Hall Mathematics Series; Prentice-Hall, Inc.: Englewood Cliffs, NJ, USA, 1961.
2. Hairer, E.; Nørsett, S.P.; Wanner, G. *Solving Ordinary Differential Equations I*, 3rd ed.; Springer: Berlin/Heidelberg, Germany, 2008.
3. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
4. Lagaris, I.E.; Likas, A.; Fotiadis, D.I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **1998**, *9*, 987–1000. [CrossRef] [PubMed]
5. Hu, B.; McDaniel, D. Applying Physics-Informed Neural Networks to Solve Navier–Stokes Equations for Laminar Flow around a Particle. *Math. Comput. Appl.* **2023**, *28*, 102. [CrossRef]
6. Cybenko, G.V. Approximation by Superpositions of a Sigmoidal Function. In *Mathematics of Control, Signals, and Systems*; van Schuppen, J.H., Ed.; Springer International: Berlin/Heidelberg, Germany, 1989; pp. 303–314.
7. Hornik, K. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Netw.* **1991**, *4*, 251–257. [CrossRef]
8. Rodrigues, J.A. Using Physics-Informed Neural Networks (PINNs) for Tumor Cell Growth Modeling. *Mathematics* **2024**, *12*, 1195. [CrossRef]
9. Du Toit, J.F.; Laubscher, R. Evaluation of Physics-Informed Neural Network Solution Accuracy and Efficiency for Modeling Aortic Transvalvular Blood Flow. *Math. Comput. Appl.* **2023**, *28*, 62. [CrossRef]
10. Zhang, W.; Ni, P.; Zhao, M.; Du, X. A General Method for Solving Differential Equations of Motion Using Physics-Informed Neural Networks. *Appl. Sci.* **2024**, *14*, 7694. [CrossRef]
11. Baty, H.; Vigon, V. Modelling Solar Coronal Magnetic Fields with Physics-Informed Neural Networks. *Mon. Not. R. Astron. Soc.* **2024**, *527*, 2575–2584. [CrossRef]
12. Ren, Z.; Zhou, S.; Liu, D.; Liu, Q. Physics-Informed Neural Networks: A Review of Methodological Evolution, Theoretical Foundations, and Interdisciplinary Frontiers Toward Next-Generation Scientific Computing. *Appl. Sci.* **2025**, *15*, 8092. [CrossRef]
13. Luo, K.; Zhao, J.; Wang, Y.; Li, J.; Wen, J.; Liang, J.; Soekmadji, H.; Liao, S. Physics-informed neural networks for PDE problems: A comprehensive review. *Artif. Intell. Rev.* **2025**, *58*, 323–378. [CrossRef]
14. Shin, Y.; Darbon, J.; Karniadakis, G.E. On the convergence of physics-informed neural networks for linear second-order elliptic and parabolic PDEs. *Commun. Comput. Phys.* **2020**, *28*, 2042–2074. [CrossRef]
15. Müller, J.; Zeinhofer, M. Notes on exact boundary values in residual minimisation. In Proceedings of the Mathematical and Scientific Machine Learning (MSML), Beijing, China, 15–17 August 2022; PMLR: Baltimore, MD, USA, 2022; Volume 190, pp. 231–240. [CrossRef]
16. Shin, Y.; Zhang, Z.; Karniadakis, G.E. Error estimates of residual minimization using neural networks for linear PDEs. *J. Mach. Learn. Model. Comput.* **2023**, *4*, 73–101. [CrossRef]
17. Mishra, S.; Molinaro, R. Estimates on the generalization error of physics-informed neural networks for inverse problems in PDEs. *IMA J. Numer. Anal.* **2022**, *42*, 981–1022. [CrossRef]
18. Mishra, S.; Molinaro, R. Estimates on the generalization error of physics-informed neural networks for approximating PDEs. *IMA J. Numer. Anal.* **2023**, *43*, 1–43. [CrossRef]
19. Yoo, J.; Kim, J.; Gim, M.; Lee, H. Error estimates of physics-informed neural networks for initial value problems. *J. Korean Soc. Ind. Appl. Math.* **2024**, *28*, 33–58. [CrossRef]
20. Biswas, A.; Tian, J.; Ulusoy, S. Error Estimates for Deep Learning Methods in Fluid Dynamics. *Numer. Math.* **2022**, *151*, 753–777. [CrossRef]
21. Zeinhofer, M.; Masri, R.; Mardal, K.A. A unified framework for the error analysis of physics-informed neural networks. *arXiv* **2024**, arXiv:2311.00529. [CrossRef]
22. Zhang, Z.; Wang, Q.; Zhang, Y.; Shen, T.; Zhang, W. Physics-Informed Neural Networks with Hybrid Kolmogorov–Arnold Network and Augmented Lagrangian Function for Solving Partial Differential Equations. *Sci. Rep.* **2025**, *15*, 10523. [CrossRef]
23. Jagtap, A.D.; Karniadakis, G.E. Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space–Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations. *Commun. Comput. Phys.* **2020**, *28*, 2002–2041. [CrossRef]
24. Son, H.; Jang, J.W.; Han, W.J.; Hwang, H.J. Sobolev Training for Physics-Informed Neural Networks. *arXiv* **2021**, arXiv:2101.08932v2. [CrossRef]

25. Chen, F.; Sondak, D.; Protopapas, P.; Mattheakis, M.; Liu, S.; Agarwal, D.; Di Giovanni, M. NeuroDiffEq: A Python package for solving differential equations with neural networks. *J. Open Source Softw.* **2020**, *5*, 1931. [CrossRef]
26. Baty, H. Solving Stiff Ordinary Differential Equations Using Physics-Informed Neural Networks (PINNs): Simple Recipes to Improve Training of Vanilla-PINNs. *arXiv* **2023**, arXiv:2304.08289.
27. Baty, H.; Baty, L. Solving Differential Equations Using Physics-Informed Deep Learning: A Hand-on Tutorial with Benchmark Tests. *arXiv* **2023**, arXiv:2302.12260.
28. Ryu, E.K. *Infinitely Large Neural Networks*; Lecture Notes in Mathematics, Number 58; Research Institute of Mathematics: Seoul, Republic of Korea, 2023.
29. Chester, D.L. Why Two Hidden Layers are Better than One. In Proceedings of the International Joint Conference on Neural Networks (IJCNN-90), Washington, DC, USA, 15–19 January 1990; IEEE: Piscataway, NJ, USA, 1990; Volume 1, pp. 265–268.
30. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.