Networks and Heterogeneous Media

https://www.aimspress.com/journal/nhm

NHM, 20(3): 1026–1060.
DOI: 10.3934/nhm.2025045
Received: 08 April 2025
Revised: 03 September 2025
Accepted: 17 September 2025
Published: 26 September 2025

Research article

A modified iterative PINN algorithm for strongly coupled system of boundary layer originated convection diffusion reaction problems in MHD flows with analysis

Arihant Patawari[1], **Shridhar Kumar**[2] **and Pratibhamoy Das**[1,*]

[1] Department of Mathematics, Indian Institute of Technology Patna, India

[2] School of Mathematics, Indian Institute of Science Education and Research Thiruvananthapuram, India

* **Correspondence:** Email: pratibhamoy@iitp.ac.in, pratibhamoy@gmail.com.

Abstract: In this work, we design an iteration-based unsupervised neural network algorithm and the theoretical bound of the loss function through Barron space to solve reaction–convection–diffusion-based magnetohydrodynamic (MHD) coupled systems for 1D and 2D problems. Our algorithm is also be able to capture the presence of boundary layers. In general, these systems are characterized by strong coupling in the reaction and convection processes and involve non-diagonally dominant matrices in the context of convection coupling. Traditional numerical techniques face challenges in approximating these problems due to the failure of the required maximum principle, which is used for the well-posedness and further convergence analysis of numerical solutions. We specifically provide a new modified iterative physics-informed neural network (MI-PINN)-based unsupervised deep learning algorithm to capture the layer behavior of singularly perturbed strongly coupled steady-state problems, appearing in MHD flows where theoretical analysis and numerical methods are limited. A different analysis based on the sigmoid activation function is provided for the steady-state case, which shows that the empirical loss under the L^2 norm is bounded and converges for the two layer-based networks- whenever the solution lies in the Barron space. Additionally, the proposed algorithm improves the neural network's output without using the boundary layer functions a priori and does not use hard constraints or interpolation with the neural network's solution. The experimental results show that the proposed algorithm performs very well for MHD flows appearing in the form of strongly coupled systems.

Keywords: multiple scale problems; neural network; strongly coupled system; PINN; MI-PINN; Barron space; unsteady MHD flow; boundary layer in singular perturbation; convergence analysis; multiple scale problems in 1D and 2D

Mathematics subject classification: 68T07, 68T20, 35G50, 68T05, 34E20, 76N20.

1. Introduction

Convection–diffusion systems play a pivotal role in modeling magnetohydrodynamic (MHD) flow problems, which involve the interaction of electrically conducting fluids, such as plasma or liquid metals, with magnetic fields. These systems are fundamental in describing the transport of physical quantities, including momentum, heat, and magnetic fields, inside fluid flow. The combined effects of convection, driven by fluid flow and diffusion due to viscosity, thermal conductivity, or resistivity, are the central point in understanding the behavior of such flows for applications.

In particular, MHD flows are characterized by coupling the Navier–Stokes equations for fluid motion with Maxwell's equations for electromagnetic fields [1–3]. The convection–diffusion equation governs several key aspects of this coupling. For example, heat transfer inside these coupled systems is described by a convection–diffusion equation that accounts for the transport of thermal energy by the fluid and its dissipation through conduction [4]. Similarly, the evolution of the magnetic field in the presence of a moving conducting fluid follows a convection–diffusion type equation, where magnetic diffusion competes with the advection of the magnetic field by the fluid [5].

Moreover, these systems arise in a wide range of practical applications. In astrophysical phenomena, such as solar winds and accretion disks, convection–diffusion processes dictate the dynamics of plasma flows in the presence of magnetic fields [5, 6]. In industrial applications, MHD is utilized in processes such as electromagnetic casting of metals, where controlling the flow of liquid metal under a magnetic field is critical to the products' quality [7]. Furthermore, these flow models are essential in the design of fusion reactors, where the plasma's stability and heat transfer must be carefully managed to maintain optimal operating conditions [4].

The governing equations of MHD flows are derived by combining the Navier–Stokes equations of fluid dynamics with Maxwell's equations for electromagnetism. For an incompressible conducting fluid, the equations are written as follows:

$$\begin{aligned} \rho\left(\frac{\partial u}{\partial t} + u \cdot \nabla u\right) &= -\nabla p + \eta \nabla^2 u + J \times B, \\ \nabla \cdot u &= 0, \\ J &= \sigma\left(E + u \times B\right), \\ \nabla \times B &= \mu_0 J, \\ \nabla \cdot B &= 0, \end{aligned} \tag{1.1}$$

where u is the fluid velocity, ρ is the density, p is the pressure, η is the dynamic viscosity, J is the current density, B is the magnetic field, E is the electric field, σ is the electrical conductivity, and μ_0 is the permeability of the free space.

The MHD system can be simplified under certain assumptions, such as a low magnetic Reynolds number or under steady-state conditions. In particular, when analyzing boundary layer flows or flows where convection dominates, the system can be reduced to a coupled system of convection–diffusion problems of the following form for the magnetic field:

$$\frac{\partial B}{\partial t} = \nabla \times (u \times B) - \nabla \times (\varepsilon \nabla \times B), \tag{1.2}$$

where ε is the magnetic diffusivity. In cases where the magnetic field and velocity field are aligned or nearly aligned, the non-linear convection term $\nabla \times (u \times B)$ dominates, and the equation above resembles a steady-state version of one-dimensional (1D) convection–diffusion equation of the following form:

$$\frac{\partial B}{\partial t} + u \cdot \nabla B = \varepsilon \nabla^2 B. \tag{1.3}$$

Assume that the fluid is driven by a constant axial pressure gradient except the flow direction, with a uniform transverse magnetic field of strength B_0 applied in the x-direction. Under these assumptions, the governing equations for the motion of a conducting fluid within the domain $\Omega \equiv (0, 1)$, in the presence of the uniform magnetic field B_0, can be written as a strongly coupled system of the following form [2, 3, 8]:

$$\begin{aligned} &\tfrac{\partial u}{\partial t} - \nabla^2 u - M\tfrac{\partial B}{\partial x} = 1, \quad x \in \Omega, \quad t \in (0, 1], \\ &\tfrac{\partial B}{\partial t} - \nabla^2 B - M\tfrac{\partial u}{\partial x} = 0, \quad x \in \Omega, \quad t \in (0, 1], \end{aligned} \tag{1.4}$$

with the following boundary and initial conditions:

$$\begin{aligned} &u(x, t) = B(x, t) = 0, \quad x \in \partial\Omega, \quad t \in (0, 1], \\ &u(x, 0) = B(x, 0) = 0, \quad x \in \Omega, \end{aligned} \tag{1.5}$$

where $\partial\Omega = \{0\} \cup \{1\}$ is the boundary points of Ω and $M = B_0\, L\sqrt{\sigma/\rho\,\eta}$ and L is the characteristic length in a 1D set-up. This form is known as a singularly perturbed form of a convection-diffusion system when the diffusion coefficient $0 < \varepsilon \ll 1$ is arbitrarily small, and its appearance leads to sharp boundary layers where standard numerical methods often struggle to accurately capture the rapid variations in the multiple scale [9] solution. In such cases, special numerical techniques such as adaptive meshing [10–12] are required to capture sharp gradients without introducing spurious oscillations.

The current study presents the MHD flow in a channel under an external magnetic field, which is perpendicular to the side of the channel. For simplicity, the flow is modeled within a rectangular form of a pipe (channel), and the cross-section is considered in the $x\,y$-plane. The interaction between the conducting fluid and the external magnetic field induces a magnetic field within the fluid, described as $B = (B_0, 0, B)$, while the velocity field of the fluid is given as $V = (0, 0, u)$. Here, the induced magnetic field B and velocity of the fluid u are functions of x for the 1D steady-state case. The MHD flow is conducted between two boundary points $\partial\Omega \equiv \{x^0_{bd}, x^1_{bd}\}$, where x^0_{bd} denotes the left boundary and x^1_{bd} denotes the right boundary, and the interior points are $\Omega \equiv (x^0_{bd}, x^1_{bd})$.

We have chosen a viscous, incompressible, electrically conducting fluid between the parallel plates. The fluid starts with a constant pressure gradient in the z-direction. Thus the coupled form of the MHD flow equations in steady-state 1D case of [13], reduced to the following system:

$$\begin{aligned} &\tfrac{d^2u}{dx^2} + M\,\tfrac{dB}{dx} = -1, \quad x \in \Omega, \\ &\tfrac{d^2B}{dx^2} + M\,\tfrac{du}{dx} = 0, \quad x \in \Omega, \end{aligned} \tag{1.6}$$

with boundary condition

$$u(x) = B(x) = 0, \quad x \in \partial\Omega, \tag{1.7}$$

The physical representation of MHD flow is shown in Figure 1.

Moreover, boundary layer-induced, convection–diffusion systems arise in scenarios; such as turbulent interactions between currents and waves [5, 14, 15], tubular models in chemical reactor theory, bio-heat

flow, neuron transport with small diffusion coefficients [16, 17], and convective flow in fractured porous media. These problems are categorized as singularly perturbed when the diffusion parameter becomes arbitrarily small. Another well-known example of such problems is the Korteweg–De Vries (KdV) equation [18] and non-linear coupled systems of parabolic problems [19].

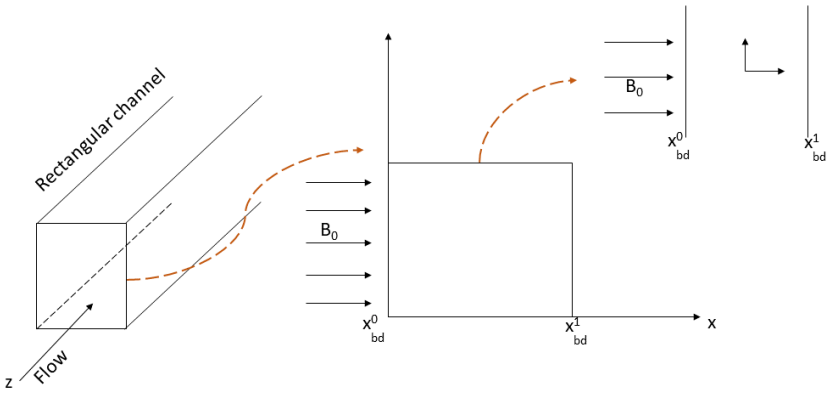


Figure 1. Rectangular channel flow in MHD and its cross section with parallel plates.

The presence of small parameters in the diffusion matrix results in the multiscale nature of the solution [20, 21], characterized by rapid variations in localized regions and slower variations elsewhere. This phenomenon leads to unbounded solution derivatives with respect to the diffusion parameters. This makes it challenging to establish the convergence of numerical approximations. Specifically, consistency analysis requires a priori bounds on the solution's derivatives, which are difficult to derive for the current class of problems with zero diagonal entries in the convection matrix.

Hence, convection–diffusion systems in MHD flows often present significant challenges for numerical simulation due to the presence of boundary layers and sharp gradients [22, 23], particularly in regions near walls or interfaces where the fluid's velocity, temperature, and magnetic fields change rapidly. As a result, advanced numerical techniques, such as layer-adapted meshes, are frequently employed to capture these behaviors accurately [10, 11]. The study of MHD flows is essential in fields like fusion research, where magnetic fields influence plasma behavior, and in the cooling processes of liquid metals in nuclear reactors [4]. In these applications, these problems often occur in complex environments where strongly coupled convection terms lead to thin boundary layers for singularly perturbed problems. Standard numerical solvers often fail, as they diverge unless the size of the stiffness matrix scales inversely with the perturbation parameters, resulting in high computational costs, especially for coupled systems of partial differential equations (PDEs). As discussed in the numerical section of present work, even layer-adapted meshes can suffer from computational inefficiencies.

In addition, the aforementioned theoretical and numerical research is limited to the availability of the M matrix structure of the convection coefficient. Having these challenges inside traditional numerical techniques, modified physics-informed neural networks (PINNs) have shown a remarkable success in diverse fields of applications, including MHD flows, especially when the M matrix structure is unavailable in the coupled system of boundary layer model problems, which will be observed in the present work.

This paper focuses on solving a more general class of strongly coupled steady-state singularly perturbed convection–reaction–diffusion systems, motivated by MHD flows, where the reaction matrix

is known to be zero [2, 3, 8]. We employ finite-difference methods on piecewise uniform layer adapted meshes which are widely available for several different classes of coupled systems in order to check their effectiveness for the present class of problems.

In recent years, deep neural networks have shown their full potential for solving differential equations over standard numerical approaches. See, for example, [24–28]. Generally, standard numerical techniques use appropriate meshes, which require a priori information on the location of large solution gradients and their widths [23, 29]. An appropriate structure of the deep neural networks can relax this prior information. The numerous contributions of deep learning approaches were mainly started by Raissi et al., who published a series of papers [24,30,31] on PINNs. A series of comprehensive overviews and several developments are also available, including GitHub code; see, for example, [32–34].

The flexibility of PINNs in approximating the solutions of differential equations is based on the universal approximation theorem [35], which states that any continuous functions, including measurable functions, can be approximated by the deep neural network. However, the theorem itself does not necessarily imply algorithmic convergence. There have been very few theoretical analyses of PINNs and their mathematical justifications to date. In the mathematical justification of PINNs, [36] considered linear second-order elliptic and parabolic PDEs and provided the consistency and convergence of the PINNs. The authors of [37] provide a generalization of the error bounds for semilinear parabolic PDEs, 1D quasilinear convection–dominated diffusion PDEs, and incompressible Euler equations using a quadrature rule. The authors of [38] also provided generalization error bounds for the Navier-Stokes equation. For more details, see [39, 40].

PINNs have several key advantages, including meshless properties and adaptability to various physical scenarios. The basic idea of this algorithm is to use the training points (known information about the physical scenarios, such as boundary conditions, and unknown information like the interior of the domain) to train the model and predict an optimal approximation. Backpropagation does this training [41] and the error is evaluated through the defined loss function. If the loss function does not satisfy the user-defined tolerance, the current parameters are updated through the backpropagation and ADAM optimizer (two moments, say $\bar{z}$ and $\bar{l}$, are used to update the neural parameter θ). Once the algorithm is fully trained (that is, the optimal parameters are achieved), it can predict an efficient output. However, the PINN has several open problems, including the theoretical convergence of the algorithms. The theoretical justification for selecting appropriate hyperparameters, including the number of hidden layers for the optimal output, is still open. Currently, these hyperparameters are tuned by users on the basis of experimental results.

PINNs have shown significant success in several fields despite their challenges to in solving singularly perturbed problems with interior and boundary layers. Sudden changes in the solution create problems for accurate prediction [42] through deep learning. The following literature provides the significance of PINNs in singularly perturbed problems. The authors of [43] used the corrector/monitor function in the output of the neural network to predict the layer behavior of the solution. They used the corrector function according to the layer behavior of the solution. Moreover, [44] developed a new hp-PINN by defining the test space (a neural network) and the trial space (a piecewise polynomial), which can detect singularities in 1D and 2D problems having rough data. In [45], the authors developed a C-PINN using a composite asymptotic expansion [46] in the neural network solution. They used two parallel subnetworks, where the output of these networks was multiplied by a factor of 1 and an exponential term. Further, sub-network outputs were combined to predict the solutions of linear and nonlinear

ordinary differential equations (ODEs) and 1D parabolic PDEs. One can see [47,48] other related papers on singularly perturbed problems with different versions of PINNs. Few theoretical works have been published to provide the generalization bound of the error term. One work [49] used the Barron space for higher-dimensional PDEs to find the generalization bound, whereas our work will use the Barron space for strongly coupled systems to obtain the generalization bound using the sigmoid activation function and updates the parameters of the neural network accordingly.

Earlier, we mentioned singularly perturbed problems and their difficulties that are based on the boundary/interior layer prediction. Approximating layer-oriented solutions is difficult for both machine learning and numerical techniques due to their unbounded derivatives of the singular component. In general, machine learning algorithms do not cooperate with the boundary condition for layer-oriented functions. Therefore, machine learning algorithms need the appropriate information or modification to accurately predict the boundary layers that cooperate with the boundary conditions. We refer to [43, 50], where a proper modification of the neural network output is achieved by adding or multiplying predefined boundary layer functions. However, in practice, such a priori information is generally unavailable for practical problems. Therefore, we develop a new PINN algorithm, called the advanced iterative PINN algorithm, which does not require any a priori information about the solution. We choose two architectures inside the current algorithm. The first architecture is simple, and it is not able to detect the boundary layer due to the spectral bias (F-frequency). However, we utilize the output of the first architecture as an initial guess, which helps us to develop an iterative PINN through the second architecture. The detailed structure is presented in Section 3.1. We describe the problem statement and its analytical properties and the corresponding difficulties in the following section.

1.1. Problem statement

Motivated by the diverse applications of singularly perturbed convection–diffusion systems in MHD flow problems, we consider 1D and 2D steady-state versions of following general class of strongly coupled boundary layer-originating problems:

$$1\text{D}: \begin{cases} \mathcal{L}[\mathbf{v}(x)] := \mathcal{L}_\epsilon \mathbf{v}(x) - \mathbf{F}(x) = 0, \quad \text{for} \quad x \in \Omega \equiv (x_{bd}^0, x_{bd}^1), \\ \mathbf{v}(x_{bd}^0) = B_1, \quad \mathbf{v}(x_{bd}^1) = B_2, \quad \text{for} \quad x \in \partial\Omega \equiv \{x_{bd}^0, x_{bd}^1\}, \end{cases} \tag{1.8}$$

where $\mathbf{v}(x) = [\mathbf{v}_1(x), \mathbf{v}_2(x)]^T$ denotes the solution vector of Eq (1.8), and

$$\mathcal{L}_\epsilon \mathbf{v}(x) := -\beta_\epsilon \mathbf{v}''(x) + \mathbf{P}(x)\mathbf{v}'(x) + \mathbf{Q}(x)\mathbf{v}(x), \tag{1.9}$$

$$\beta_\epsilon = \text{diag}\{\epsilon_1, \epsilon_2\}, \ \mathbf{P}(x) = \begin{bmatrix} p_{11}(x) & p_{12}(x) \\ p_{21}(x) & p_{22}(x) \end{bmatrix}, \ \mathbf{Q}(x) = \begin{bmatrix} q_{11}(x) & q_{12}(x) \\ q_{21}(x) & q_{22}(x) \end{bmatrix},$$

$$\mathbf{F}(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix}, \ B_1 = \begin{bmatrix} b_{11} \\ b_{21} \end{bmatrix}, \ B_2 = \begin{bmatrix} b_{12} \\ b_{22} \end{bmatrix},$$

and $\bar{\Omega} = \Omega \cup \partial\Omega$ denotes the closure of Ω.

The operators in problem Eq (1.8) will be of the following form, which will be utilized for the further analysis of the proposed deep learning algorithm for $i = 1, 2$:

$$\mathcal{L}_i[\mathbf{v}_i(x)] := \quad -\epsilon_i \mathbf{v}_1''(x) + p_{i1}(x)\mathbf{v}_1'(x) + p_{i2}(x)\mathbf{v}_2'(x) + q_{i1}(x)\mathbf{v}_1(x) + q_{i2}(x)\mathbf{v}_2(x) - f_i(x). \tag{1.10}$$

Without loss of generality, assume $0 < \epsilon_1 \leq \epsilon_2 \leq 1$. For simplicity, we choose the 2×2 system above for further study, as the present analysis can be extended analogously for $n \times n$ systems. However, the proposed process will automatically follow in a similar manner for a general class of $n \times n$ systems, where the convection matrix $\mathbf{P}$ and the reaction matrix $\mathbf{Q}$ are coupled matrices of order $n \times n$ and $\beta_\epsilon = \text{diag}\{\epsilon_1, \epsilon_2, \ldots, \epsilon_n\}$, where $0 < \epsilon_i \leq 1$ and we control the thickness of the boundary layer in the flow.

Similarly, motivated by Eq (1.1), the general formulation of the 2-D MHD model is defined as follows [51]:

$$\begin{cases} u_t - \epsilon\Delta u + u \cdot \nabla u + \nabla p + \mu B \times curl B = f, \quad (\mathbf{x}, t) \in \Omega \times (0, T], \\ B_t + \sigma^{-1} curl(curl B) - \mu curl(u \times B) = g, \quad (\mathbf{x}, t) \in \Omega \times (0, T], \\ \nabla \cdot u = 0, \quad (\mathbf{x}, t) \in \Omega \times (0, T], \\ \nabla \cdot B = 0, \quad (\mathbf{x}, t) \in \Omega \times (0, T], \end{cases} \tag{1.11}$$

with the initial and boundary conditions

$$\begin{cases} u(\mathbf{x}, 0) = u_0(\mathbf{x}), \quad B(\mathbf{x}, 0) = b_0(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ u(\mathbf{x}, t)|_{\partial\Omega} = 0, \quad (B(\mathbf{x}, t) \cdot n)|_{\partial\Omega} = 0, \quad (n \times curl B(\mathbf{x}, t))|_{\partial\Omega} = 0. \end{cases}$$

here, $\Omega \in \mathbb{R}^2$, $\partial\Omega$ denotes the boundary, p is the pressure, ϵ is the kinematic viscosity, σ is the electric conductivity, μ is the magnetic permeability, f and g are the source functions, u_0 and b_0 are given smooth functions, T is the final time, and n is the unit normal vector outside the $\partial\Omega$.

In the context of systems of singularly perturbed problems, recent works such as [10, 11, 29, 52, 53] provide insights into systems with diagonal convection matrices. In addition, systems with convection matrices that satisfy the properties of the M matrix have been analyzed in [54]. In such cases, both theoretical and numerical analyses are available, based on the non-standard maximum principle [55], which is required for the uniqueness of the solution and to obtain the standard error bounds.

However, to the best of our knowledge, no existing literature addresses a theoretical or numerical convergence analysis through mathematics for systems with non-diagonally dominant nonconstant matrices in the context of convection coupling scenarios. Therefore, this paper aims to develop a new deep learning algorithm to approximate solutions for such problems, achieving accuracy up to a user-defined residual error tolerance.

2. Numerical difficulties for strongly coupled systems

Let us first consider the simple case for the 1D system (1.8) where the convection matrix is a diagonal matrix, i.e., $p_{ij}(x) = 0$ for $i, j = 1, 2$, and $i \neq j$. In this case, the corresponding problem is referred to as a weakly coupled convection-dominated problem, and the analytical properties of its solution and its a priori behavior are available. A priori bounds on the derivatives of the components can be derived through the solution's decomposition. Let us first decompose the solution into its regular $\mathbf{v}^r$ and singular $\mathbf{v}^s$ components [52, 55, 56], so that $\mathbf{v} = \mathbf{v}^r + \mathbf{v}^s$. This decomposition is intended to separate the regular part which behaves smoothly, i.e., the derivatives of this component upto some order, are bounded. Hence, it does not lead to a boundary layer component. The components of the singular derivative are unbounded. This means that this component includes the boundary layer part of the original solution.

The following lemma shows that the continuous solution of the present problem (where the convection matrix is a diagonal matrix with positive diagonal entries) is unique.

Lemma 2.1. *[57] Let* $\mathbf{v} \in C(\overline{\Omega}) \cap C^2(\Omega)$ *solve the weakly coupled version of Eq (1.8) and* $\mathcal{L}\mathbf{v} \geq 0$ *in* Ω *and* $\mathbf{v} \geq 0$ *in* $\{x_{bd}^0, x_{bd}^1\}$. *Then* $\mathbf{v} \geq 0$ *for all* $x \in \overline{\Omega}$.

Note that the analytical solution is uniformly bounded from the following lemma. In addition, all the solution's components are also uniformly bounded. This means that the solution's bound does not depend on the inverse powers of the diffusion parameters.

Lemma 2.2. *[55] The continuous solution* $\mathbf{v}$ *of the weakly coupled version of Eq (1.8) satisfies the following uniform bound:*

$$\|\mathbf{v}\|_{\overline{\Omega}} \leq \max\{|\boldsymbol{B}_1|, |\boldsymbol{B}_2|\} + \frac{1}{\alpha^*}\|\boldsymbol{F}\|_{\overline{\Omega}},$$

where $p_{kk} \geq \alpha^* > 0$, $k = 1, 2$, *and* $\|.\|_{\overline{\Omega}}$ *denotes the maximum norm over* $\overline{\Omega}$.

We now state the unbounded behavior of the derivative bounds of the analytical solution for weakly coupled convection–diffusion systems.

Lemma 2.3. *The regular component* $\overline{\mathbf{v}}^r = [\mathbf{v}_1^r, \mathbf{v}_2^r]^T$ *satisfies the following estimates, for* $\varepsilon_1 \leq \varepsilon_2$:

$$\|\partial_x^i \mathbf{v}_j^r\|_\Omega \leq C, \quad i = 0, 1, 2, \ j = 1, 2. \quad \|\partial_x^3 \mathbf{v}_1^r\|_\Omega \leq C\varepsilon_1^{-1}, \quad \|\partial_x^3 \mathbf{v}_2^r\|_\Omega \leq C\varepsilon_2^{-1}.$$

The singular component $\mathbf{v}^s = [\mathbf{v}_1^s, \mathbf{v}_2^s]^T$ *satisfies the following bounds:*

$$\left|\partial_x^i \mathbf{v}_1^s(x)\right| \leq C\left(\varepsilon_1^{-i}\mathfrak{T}_{\varepsilon_1}(x) + \varepsilon_2^{-i+1}\mathfrak{T}_{\varepsilon_2}(x)\right), \quad \left|\partial_x^i \mathbf{v}_2^s(x)\right| \leq C\varepsilon_2^{-i}\mathfrak{T}_{\varepsilon_2}(x), \quad i = 1, 2,$$

$$\left|\partial_x^3 \mathbf{v}_1^s(x)\right| \leq C\left(\varepsilon_1^{-3}\mathfrak{T}_{\varepsilon_1}(x) + \varepsilon_2^{-2}\mathfrak{T}_{\varepsilon_2}(x)\right), \quad \left|\partial_x^3 \mathbf{v}_2^s(x)\right| \leq C\varepsilon_2^{-1}\left(\varepsilon_1^{-1}\mathfrak{T}_{\varepsilon_1}(x) + \varepsilon_2^{-2}\mathfrak{T}_{\varepsilon_2}(x)\right),$$

where $\mathfrak{T}_{\varepsilon_1}(x) = e^{-(x_{bd}^1 - x)\alpha^*/\varepsilon_1}, \quad \mathfrak{T}_{\varepsilon_2}(x) = e^{(x_{bd}^0 - x)\alpha^*/\varepsilon_2}$.

These derivative bounds are crucial for the a priori information of the solution. It is important to note that these derivative components of the singular component become unbounded as $\varepsilon \to 0$. This complicates the computational analysis. To address this, we must construct a specialized grid structure, as described in [22, 23, 58–60], to rigorously prove the convergence of standard numerical algorithms on adaptive meshes. However, as we shall show in the numerical Section (5) that the existing standard computational algorithms do not detect the boundary layers in the present class of problems Eq (1.8).

In general, the maximum principle [55, 61] is used to obtain the uniqueness of the continuous or approximate solution and its stability. It is also useful to ensure that the error satisfies certain bounds according to the comparison principle. However, the standard maximum principle does not hold even when the coupled convection matrix is considered an M matrix [54]. Hence, the absence of the maximum principle significantly complicates the theoretical analysis. In this context, studies such as [54] guarantee the existence and uniqueness of the solutions of strongly coupled convection-dominated systems by introducing a non-standard maximum principle, which also restricts the sign of the solution's derivatives. This approach provides a foundation for the stability analysis of the continuous solution. Although the work in [54] focuses on coupled convection processes through M matrix conditions and derives derivative bounds that do not involve exponential terms (unlike [11, 62]). It obtains an approximate solution according to the following finite-difference scheme on the fitted Shishkin mesh [11, 54, 63]

$\{x\}_i^N = \{0 = x_0 < x_1 < \cdots < x_N = 1\}$. Let $\mathbf{v}_i^N = [\mathbf{v}_1(x_i), \mathbf{v}_2(x_i)]^T$ denotes the numerical solution of the following discrete version of the continuous problem Eq (1.8):

$$\begin{aligned} \mathcal{L}^N \mathbf{v}_i^N &:= \mathcal{L}_\epsilon^N \mathbf{v}_i^N - \mathbf{F}_i = 0, \quad \text{for } i = 1, \cdots, N-1, \\ \mathbf{v}_0^N &= B_1, \quad \mathbf{v}_N^N = B_2, \end{aligned} \tag{2.1}$$

where

$$\mathcal{L}_\epsilon^N \mathbf{v}_i^N := -\beta_\epsilon \delta_x^2 \mathbf{v}_i^N + \mathbf{P}_i \partial_x^+ \mathbf{v}_i^N + \mathbf{Q}_i \mathbf{v}_i^N,$$

$$\mathbf{P}_i = \{p_{jk}(x_i)\}_{2\times 2}, \ \mathbf{Q}_i = \{q_{jk}(x_i)\}_{2\times 2}, \ \mathbf{F}_i = \{f_j(x_i)\}_{2\times 1},$$

$$\delta_x^2 \mathbf{v}_i^N = \frac{\partial_x^+ \mathbf{v}_{i+1}^N - \partial_x^- \mathbf{v}_i^N}{\overline{h}_i}, \quad \partial_x^+ \mathbf{v}_i^N = \frac{\mathbf{v}_{i+1}^N - \mathbf{v}_i^N}{h_{i+1}}, \quad \partial_x^- \mathbf{v}_i^N = \frac{\mathbf{v}_i^N - \mathbf{v}_{i-1}^N}{h_i},$$

and $h_{i+1} = x_{i+1} - x_i$, $\overline{h}_i = \frac{h_i + h_{i+1}}{2}$. Here, the authors demonstrate that the discrete operator Eq (2.1) satisfies a non-standard discrete maximum principle, thereby establishing the uniqueness and stability of the discrete operator. However, our aim is to consider a more general form of convection matrices, particularly, the non-diagonally dominant matrices, which arise in the steady-state version of the 1D version of MHD flow problems. We conduct numerical experiments in Section 5 for the scheme Eq (2.1) on our problem of interest. The scheme struggled to achieve convergence and failed to provide an oscillation-free solution adapted to the boundary layer. This motivated us to develop the proposed PINN algorithm, which aims to produce an oscillation-free solution accurately within a user-defined error tolerance. For coupled systems with certain specific types of convection matrices, only a few studies are available [11, 57, 62, 64], where the analysis is based on the energy norm without relying on pointwise derivative bounds for the solution's components as stated in Lemma 2.3. An interesting practical scenario arises when $p_{ii}(x) = 0$ for $i = 1, 2$, and the remaining coefficients are not restricted to constants. In such cases, it has been observed, both theoretically and through numerical experiments on coarse grids, that existing methods such as [54] fail to solve the general case considered in this paper.

Direct applications of such non-M matrix couplings appear in works such as [1, 2, 4, 6, 7, 51], which motivate our consideration of the present form of coupled systems. From an application perspective, our formulation is relevant for MHD flow problems. For example, reference [1] considers a strongly coupled MHD system involving a non-M matrix. In this context, reference [2] performs an exponentially compact higher-order von Neumann analysis. However, their analysis is restricted to constant convection and reaction coefficients. In reference [51], a related nonlinear model is considered, where the analysis is limited to high diffusion coefficients and theoretically fails due to unbounded derivatives (see Lemma 2.2 in [51]). Furthermore, the same non-diagonally dominant nature of the stiffness matrix makes the problem numerically challenging. Our aim, in contrast, is to address a more general case, such as strongly coupled systems with arbitrarily small diffusion terms and variable convection and reaction terms. We consider that the solution is continuous or lies inside Barron space.

Hence our focus is on using the recent and increasingly popular class of PINN algorithms, together with their advanced modifications, to solve more complex and diverse physical problems. To address the previously discussed challenges, we propose a novel, advanced modified iterative PINN (MI-PINN) algorithm that is capable of detecting boundary layer phenomena for a broader class of problems in which the convective flow is not necessarily constant. This approach extends the capabilities of PINNs to address complex physical problems. In contrast to previous studies [1, 2, 4, 6, 7, 51], it tackles a more

general class of problems, specifically, strongly coupled systems with variable convection and reaction terms and arbitrarily small diffusion coefficients, while effectively handling nonlinearity.

3. Modification of PINNs

This section provides a suitable modification of PINN-based algorithms, including some basic terminology for solving strongly coupled systems of boundary layer problems (1.8).

3.1. Physics-informed neural networks

A PINN takes input and approximates the output through the neural network structure. For example, a PINN takes input values from the domain $\overline{\Omega}$ and predicts the output using backpropagation along with the appropriate hyperparameters; see, for example, Table 1. An approximation of a deep neural network is a composition of the affine transformations and linear/non-linear activation functions. A mathematical expression of the deep neural network is as follows:

$$\mathbf{v}(x;\theta) = A_{k+1} \circ \sigma \circ A_k \circ \cdots \circ A_2 \circ \sigma \circ A_1(x), \quad k = 1, 2, \cdots, K-1. \tag{3.1}$$

From Eq (3.1), it is clear that the depth of the neural network architecture is K and the number of hidden layers is $K-1$. Here, the neural network function is defined as $\mathbf{v}(x;\theta)$; $A_k : \mathbb{R}^{n_{k-1}} \to \mathbb{R}^{n_k}$ denotes the affine transformation within the layers of the deep network, where n_k denotes the number of neurons in the kth layer; σ is the activation function; $\theta = \{w, b\}$ denotes the parameter of the neural networks; and $\circ$ denotes the composition of functions. The parameters of the deep neural network, such as weights and biases, can be represented mathematically. The matrix representation of the weights for the kth layer in the depth of the neural network is denoted as $w^k : \mathbb{R}^{n_k} \times \mathbb{R}^{n_{k-1}}$, where n_k denotes the neurons of the kth layer and n_{k+1} denotes the neurons corresponding to the $(k+1)$th layer and $k = 1, 2, \cdots, K-1$. The bias for the kth layer is represented as $b^k \in \mathbb{R}^{n_k}$. We can now easily understand several different structures, with one output, two outputs, and many more. This process is generally known as the forward pass.

Table 1. Hyper-parameters used for the present numerical experiments via the iterative PINN architecture for solving coupled systems of problems.

Hyper-parameters	Value
Activation function	Sigmoid
Optimizer	*ADAM*
Epoch	1000
Learning rate	Adaptive lr
Size of hidden layers	1
No. of neurons in $N2$'s hidden layer	100

Singularly perturbed problems usually have interior or boundary layers in the solution. The basic deep learning algorithms do not capture the sharpness of the boundary layers in the solution because of the high complexity of a singular component arising from spectral bias. The authors of [65] have shown that deep learning algorithms learn the regular parts faster for these problems compared to the singular parts, where the solution varies rapidly. To overcome these issues, we develop an advanced iterative PINN algorithm to efficiently solve coupled systems of singularly perturbed problems in the following section.

3.2. Iterative PINN algorithm

The standard PINN architecture in Section 3.1 is a key component of the proposed iterative PINN algorithm used to solve the coupled system of singularly perturbed problems Eq (1.8) because standard deep neural networks [65] struggle to capture the layer in the solution due to spectral bias. An overview of the modified iterative PINN is as follows.

- The accurate prediction of singularly perturbed problems through the PINN uses prior information of the boundary layer, see, i.e., [25, 43]. Here, we employ the MI-PINN without any prior information.
- The $N1$ structure provides an intuitive understanding of the boundary layer's appearance and helps us to provide the initial guesses of the neural parameter.
- These initial guesses provide a more accurate representation of the smooth component compared with random initialization. It is well known that, due to spectral bias, neural networks learn smooth features more rapidly than singular features. Utilizing the improved initial guesses of the smooth part, the network $N2$ can focus more effectively on learning the singular component and achieve better performance than the traditional PINN approach.

This iterative approach in Figure 2 begins with the initial guesses used to generate an approximate neural network solution. The $N1$ architecture produces initial guesses for the $N2$ architecture after completing a user-specified number of epochs. Iterative algorithms are applied in the $N2$ architecture to minimize the losses of the coupled system of singularly perturbed problems until the user-defined tolerance is achieved or the provided number of iterations is completed. The general approach of the systematic flow chart of the newly developed iterative PINN algorithm is given in Figure 2.

The present proposed algorithm uses numerous hyperparameters and a non-linear activation function, which are all listed in Table 1. These hyperparameters are used in backpropagation to find the optimal values of the parameters in the neural network. Hence, backpropagation plays a trivial role in updating the parameters θ of the network during training and in quantifying the required error of the loss function.

A general way of writing the loss function for the system of ODEs is as follows:

$$\mathbf{L}[\mathbf{v}(x;\theta)] := \mathbf{L}^{int}[\mathbf{v}(x;\theta)] + \mathbf{L}^{bd}[\mathbf{v}(x;\theta)],$$

where $\mathbf{L}^{int}[\mathbf{v}(x;\theta)] = [L_1^{int}[\mathbf{v}_1(x;\theta)], L_2^{int}[\mathbf{v}_2(x;\theta)]]^T$ represents the loss functions corresponding to the residual of the differential equation at interior points Ω and

$$\mathbf{L}^{bd}[\mathbf{v}(x;\theta)] = [L_1^{bd}[\mathbf{v}_1(x;\theta)], L_2^{bd}[\mathbf{v}_2(x;\theta)]]^T$$

represents the loss functions of the differential equation corresponding to the boundaries $\partial\Omega$. The primary goal of the newly developed algorithm is to minimize the error computed by the abovementioned loss functions for the layer-oriented approximated solution of the coupled system of singularly perturbed problems. The proposed iterative algorithm removes the difficulties of spectral bias and the requirement of a priori information such as the boundary layer functions within the neural network solution by choosing the initial guesses. In each iteration, the loss function computes the error, which can be minimized by updating the weights and biases. A general form of the cost function can be expressed in the following manner:

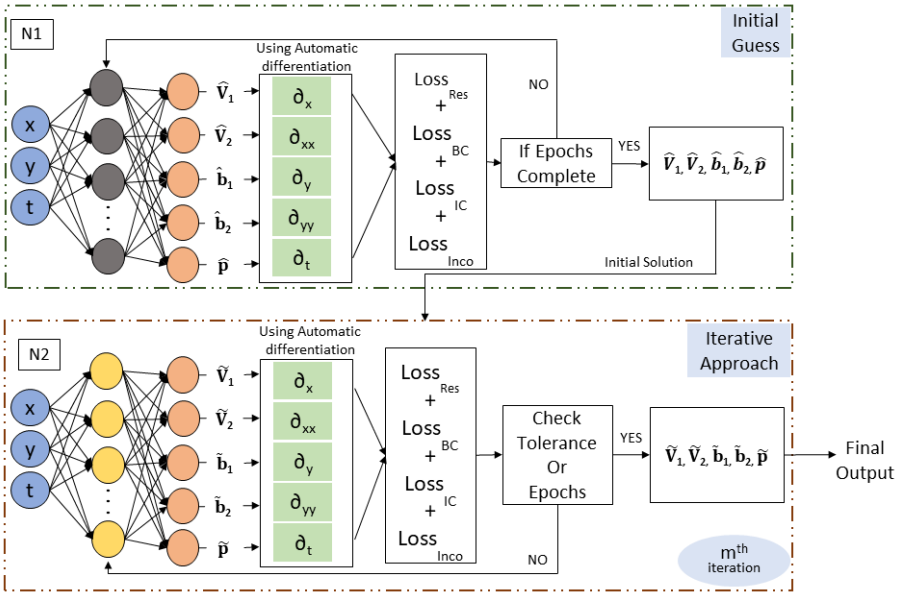


Figure 2. This is an architecture of the modified iterative PINN. In this architecture, we first, find an initial guess of the solutions of the 2D MHD flow, say, $[\hat{\mathbf{v}}_1(\mathbf{x},t;\theta), \hat{\mathbf{v}}_2(\mathbf{x},t;\theta), \hat{b}_1(\mathbf{x},t;\theta), \hat{b}_2(\mathbf{x},t;\theta), \hat{p}(\mathbf{x},t;\theta)]^T$, and the corresponding weight and bias with the help of the $N1$ architecture. Next, this initial guess is used in the $N2$ architecture which can have a different number of neurons from the $N1$ architecture and will be updated iteratively for finding an accurate prediction of $[\tilde{\mathbf{v}}_1(\mathbf{x},t;\theta), \tilde{\mathbf{v}}_2(\mathbf{x},t;\theta), \tilde{b}_1(\mathbf{x},t;\theta), \tilde{b}_2(\mathbf{x},t;\theta), \tilde{p}(\mathbf{x},t;\theta)]^T$ for the coupled system of MHD flows. In this figure, we have only shown the mth iteration, where $1 \le r \le m \le l$.

$$L[\mathbf{v}(x;\theta)] := \sum_{i=1}^{2} L_i[\mathbf{v}_i(x;\theta)], \tag{3.2}$$

where $L_i[\mathbf{v}_i(x;\theta)]$ is defined in the following manners:

$$L_i[\mathbf{v}_i(x;\theta)] := L_i^{int}[\mathbf{v}_i(x_{int};\theta)] + L_i^{bd}[\mathbf{v}_i(x_{bd}^0;\theta)] + L_i^{bd}[\mathbf{v}_i(x_{bd}^1;\theta)], \quad \textit{for all } i = 1,2, \tag{3.3}$$

where $L_i^{int}(x;\theta)$, *and* $L_i^{bd}(x;\theta)$ are defined in the following manners:

$$\begin{aligned} L_i^{int}[\mathbf{v}_i(x_{int};\theta)] &= \tfrac{1}{M_{int}} \textstyle\sum_{int=1}^{M_{int}} |\mathcal{L}_i[\mathbf{v}_i(x_{int};\theta)]|^2, \quad i = 1,2, \\ L_i^{bd}[\mathbf{v}_i(x_{bd}^0;\theta)] &= \tfrac{1}{M_{bd}^0} \textstyle\sum_{bd=1}^{M_{bd}^0} |\mathbf{v}_i(x_{bd}^0;\theta) - b_{i1}|^2, \quad i = 1,2, \\ L_i^{bd}[\mathbf{v}_i(x_{bd}^1;\theta)] &= \tfrac{1}{M_{bd}^1} \textstyle\sum_{bd=1}^{M_{bd}^1} |\mathbf{v}_i(x_{bd}^1;\theta) - b_{i2}|^2, \quad i = 1,2. \end{aligned} \tag{3.4}$$

From the expressions above, we can rewrite $L_i[\mathbf{v}_i(x;\theta)]$ in Eq (3.2) as follows for $i = 1,2$:

$$L_i[\mathbf{v}_i(x;\theta)] := \frac{1}{M_{int}} \sum_{int=1}^{M_{int}} |\mathcal{L}_i[\mathbf{v}_i(x_{int};\theta)]|^2 + \cdots$$

$$\frac{1}{M_{bd}^0}\sum_{bd=1}^{M_{bd}^0}|\mathbf{v}_i(x_{bd}^0;\theta)-b_{i1}|^2+\frac{1}{M_{bd}^1}\sum_{bd=1}^{M_{bd}^1}|\mathbf{v}_i(x_{bd}^1;\theta)-b_{i2}|^2, \tag{3.5}$$

where x_{int} are the points in the interior of the domain Ω, and M_{int} is the total number of points taken from the interior of the domain Ω, x_{bd}^0 denotes the first boundary, and x_{bd}^1 denotes the second boundary. M_{bd}^0 denotes the total number of points taken from the first boundary, and M_{bd}^1 denotes the total number of points taken from the second boundary. Here, the total number of points in the interior of the domain is $M_{int} = (M-1)$, provided that $M > 1$ is the number of partitions that correspond to the space. The boundary points M_{bd}^0, M_{bd}^1 help the $N2$ structure train the boundary regions accurately for singularly perturbed problems. One can also reduce the total number of points taken from the boundary in order to reduce the computational cost. However, for singularly perturbed problems with boundary layers, we need to train the boundary regions more efficiently to capture the boundary layers. Similarly, the loss function corresponding to the 2D MHD equation Eq (1.11) is defined as follows:

$$\begin{aligned} L[\xi] :&= \frac{1}{M_{int}}\sum_{int=1}^{M_{int}}|\mathcal{L}[\xi_{int}]|^2+\frac{1}{M_{bd}}\sum_{bd=1}^{M_{bd}}|\mathcal{L}[\xi_{bd}]|^2+\frac{1}{M_{ini}}\sum_{ini=1}^{M_{ini}}|\mathcal{L}[\xi_{ini}]|^2,\\ L[\phi^{inco}] :&= \frac{1}{M_{int}}\sum_{int=1}^{M_{int}}|\nabla\cdot\phi_{int}|^2, \end{aligned} \tag{3.6}$$

where $\xi \equiv \xi(\mathbf{x},t;\theta)$, $\xi_{int} \equiv \xi_{int}(\mathbf{x}_{int},t_{int};\theta)$, $\xi_{bd} \equiv \xi_{bd}(\mathbf{x}_{bd},t_{bd};\theta)$, $\xi_{ini} \equiv \xi_{bd}(\mathbf{x}_{ini},t_{ini};\theta)$, and$\phi^{inco} \equiv \phi^{inco}(\mathbf{x},t;\theta)$, $\phi_{int} \equiv \phi_{int}(\mathbf{x},t;\theta)$. $\mathbf{x}\in\mathbb{R}^2$, for $\xi=[u_1,u_2]^T, \xi=[b_1,b_2]^T$, $\phi^{inco}=(u^{inco},B^{inco})$ and $\nabla\cdot\phi_{int}=(\nabla\cdot u_{int},\nabla\cdot B_{int})$. Here, u^{inco} and B^{inco} denote the incompressible condition of the velocity and magnetic field, respectively. Moreover, x_{int}, t_{int} denote the points in the interior of the domain M_{int} is the total number of points in the interior of the domain, x_{bd}, t_{bd} are boundary points, M_{bd} is the total number of points from the boundary, x_{ini} and t_{ini} denote the points of the initial condition, and M_{ini} denotes the total number of points from the initial condition. We use the uniform partition to generate the input data. The proposed algorithm takes these input data as X_1^0 in the Algorithm 1 and computes the mean square error. The size of the X_1^0 is equal to the size of the sum of the M_{int}, M_{bd}, M_{ini}; X_2^0 is defined in a similar way. We can define the component wise MHD loss functions as

$$\mathcal{L}[\xi(\mathbf{x}_{int},t_{int};\theta)],\mathcal{L}[\xi(\mathbf{x}_{bd},t_{bd};\theta)],\mathcal{L}[\xi(\mathbf{x}_{ini},t_{ini};\theta)], \quad \text{for} \quad \xi=u,B,$$

in the same way as the ODE formulation. In the 2D case, the geometric representation of the point generation is shown in Figure 3.

In Figure 3, it is clear that the boundary loss function for the 2D problem is formulated on the basis of the specified boundary conditions on the left, right, front, and back boundaries. The loss function corresponding to the initial condition is created from the given initial conditions of the system. The residual loss function is constructed from the velocity residuals, the magnetic field residuals, and the incompressible conditions. Putting the required values in Eq (3.6), we get the complete loss function $L[\mathcal{U}(\mathbf{x},t;\theta)]$ of the MHD flow, which is defined as follows:

$$L[\mathcal{U}(\mathbf{x},t;\theta)] := L[u(\mathbf{x},t;\theta)]+L[B(\mathbf{x},t;\theta)]+L[u^{inco}(\mathbf{x},t;\theta)]+L[B^{inco}(\mathbf{x},t;\theta)], \tag{3.7}$$

where the output of the 2D MHD flow is defined as

$$\mathcal{U}(\mathbf{x},t;\theta)=[u_1(\mathbf{x},t;\theta),u_2(\mathbf{x},t;\theta),b_1(\mathbf{x},t;\theta),b_2(\mathbf{x},t;\theta),p(\mathbf{x},t;\theta)]^T.$$

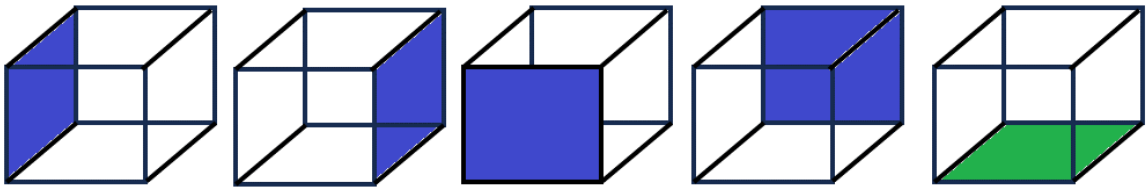


Figure 3. Generation of the boundary and initial data in 2D. Boundary data are created from the left, right, front,and back faces of the cube, highlighted in blue color, and the initial data are created from the bottom face of the cube, highlighted in green color.

For the newly proposed algorithm, we need to define two neural network structures. For simplicity, we denote $N1$ as the first structure and $N2$ as the second structure. We designed these structures in such a way as to obtain the layer behavior in the approximated solution for the coupled systems. For the ODE case in Eq (1.8), let us define the output of the $N1$ architecture as $\hat{\mathbf{v}}(x;\theta) = [\hat{\mathbf{v}}_1(x;\theta), \hat{\mathbf{v}}_2(x;\theta)]^T$; for the $N2$ architecture, the outputs are defined as $\tilde{\mathbf{v}}(x;\theta) = [\tilde{\mathbf{v}}_1(x;\theta), \tilde{\mathbf{v}}_2(x;\theta)]^T$. First, we use the $N1$ architecture to find the initial guesses and store them as $\hat{\mathbf{v}}^{new}(x;\theta) = [\hat{\mathbf{v}}_1^{new}(x;\theta), \hat{\mathbf{v}}_2^{new}(x;\theta)]$, which is directly equal to $\hat{\mathbf{v}}(x;\theta) = [\hat{\mathbf{v}}_1(x;\theta), \hat{\mathbf{v}}_2(x;\theta)]^T$. Next, $N1$ is also trained on the boundary data of the given problem and approximates the output throughout the domain. In between the training and output processes, backpropagation plays an significant role in updating the parameters on the basis of the available user-defined epochs through the optimizer, which is computed through the loss functions (defined in Eq (3.2)). If user-defined epochs are completed, the $N1$ structure will give initial guesses as the output form.

After completion of the procedure of the structure $N1$, we use another network, say $N2$, with the output $\tilde{\mathbf{v}}(x;\theta) = [\tilde{\mathbf{v}}_1(x;\theta), \tilde{\mathbf{v}}_2(x;\theta)]^T$ for the proper implementation of the iterative PINN algorithm. In this structure, we used the combination of the loss function $L_1[\mathbf{v}_1(x;\theta)]$ and $L_2[\mathbf{v}_2(x;\theta)]$ defined in Eq (3.5) and compute them iteratively. First, we replace $N2$'s output $[\tilde{\mathbf{v}}_1(x;\theta), \tilde{\mathbf{v}}_2(x;\theta)]^T$ with the initial guess of $N1$'s output $[\hat{\mathbf{v}}_1(x;\theta), \hat{\mathbf{v}}_2(x;\theta)]^T$ and train them iteratively with the help of the combination of the first and second loss functions, and also use backpropagation for better approximation. During backpropagation, we use an adaptive learning rate. The adaptive learning rate is used in the proposed algorithm for better approximation. For early convergence and oscillation-free loss error, we reduce the learning rate following the same pattern as the increase in iterations. We use the learning rate $= 10^{-2}$ up to the 2nd iteration, 5×10^{-3} is used from the 3rd to the 12th iteration, 10^{-3} is used from the 13th to 25th iteration, 5×10^{-4} is used from the 26th to the 55th iteration, 10^{-4} is used from the 56th to the 100th iteration, and next 10^{-5} is used in the proposed algorithm. One iteration is equal to 1000 epochs. This $N2$ architecture is trained and tested on the same domain points of $\bar{\Omega}$ along with the same hyperparameters used in the structure $N1$. The architecture $N2$ approximates the output based on the user-defined tolerance achieved or up to the user-defined iterations. A general view of the implementation procedure of the iterative PINN is shown in Algorithm 1. The convergence of the proposed algorithm is demonstrated through numerical experiments, and these numerical results are further justified by rigorous theoretical analysis. We have done several 1D and 2D numerical experiments to check the convergence of the algorithm by fixing the number of neurons in the hidden layer and varying the number of training iterations. Motivated by the universal approximation bound of the Barron space for the two-layer neural network mentioned in [66] and [49], we have included an alternative

convergence proof based on the sigmoid activation function used inside the neural network algorithm, updating the parameters on the basis of the number of iterations in the theoretical bound section.

4. Convergence of an approximate solution in Barron space

We now establish a universal approximation bound of the difference between the exact and neural network solutions for the coupled system of ODEs generated through neural network algorithms in the Barron class framework [66]. This analysis can also be used for MHD flow equations in the steady-state case, since the solution of the MHD flow equation is continuous. The concept of universal approximation bound in Barron space is related to the Fourier integral representation of functions. Barron space consists of those functions whose Fourier transformation exists and is bounded. The Fourier transformation is defined as follows.

Let $\mathcal{F}$ be a Fourier transformation of $f \in L^1(\mathbb{R})$. Then the Fourier transformation of $f(x)$ is

$$\mathcal{F}(f)(s) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{-isx} f(x)dx.$$

The inverse Fourier transformation of $\tilde{f} \in L^1(\mathbb{R})$ is denoted as $\mathcal{F}^{-1}(\tilde{f})$ and is defined as

$$\mathcal{F}^{-1}(\tilde{f})(x) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{isx} \tilde{f}(s)ds.$$

If the given function f can be represented in terms of the following integral form using the Fourier transformation:

$$f = \int_{\Omega} w_2\sigma(w_1x + b)\rho(dw_2, dw_1, db).$$

The function f above can also be represented as a two-layer neural network in the discrete sense as follows:

$$f_\tau = \frac{1}{\tau} \sum_{i=1}^{\tau} w_{2i}\sigma(w_{1i}x + b_i),$$

where τ denotes the number of neurons in the hidden layer. Hence, one hidden layer is sufficient to provide the theoretical approximation using Barron space.

Hence, the concept of Barron space is derived from a two-layer neural network, which consists of a single hidden layer with $\tau \in \mathbb{N}$ neurons and an output layer. The representation of a two-layer neural network having two outputs is given by:

$$(v_\tau)_j(x) \quad = \tfrac{1}{\tau} \textstyle\sum_{i=1}^{\tau} w_{j2i}\sigma(w_{1i}x + b_i), \quad \text{for } j = 1, 2, \quad x \in \overline{\Omega}. \tag{4.1}$$

here, σ is an activation function and $(w_{12i}, w_{22i}, w_{1i}, b_i)$ are the neural network's parameters. At the start of the algorithm, these parameters are randomly chosen under some probability distribution, then the averaged sum of Eq (4.1) with an infinite width (the number of neurons in the hidden layer) converges to the following probability integral [49]:

$$(v_\rho)_j(x) \quad := \textstyle\int w_{j2}\sigma(w_1x + b)\rho(dw_{j2}, dw_1, db), \quad \text{for } j = 1, 2, \quad x \in \overline{\Omega}, \tag{4.2}$$

where ρ denotes the probability measure.

Algorithm 1 Modified iterative PINN algorithm for approximation of MHD flow:

Input: Domain (Ω), adaptive learning rate (η), user-selected tolerance (κ), epoch (m), $l > 1$, iteration(r), activation function(σ), depth($K = 2$), boundary points ($\mathbf{x}_{bd}$), total number of boundary points (M_{bd}), initial points ($\mathbf{x}_{ini}$), total number of initial points (M_{ini}), interior points ($\mathbf{x}_{int}$), total number of interior points (M_{int}), and time T, $\delta > 0$.

Result: $u(\mathbf{x}, t; \theta^*)$, $B(\mathbf{x}, t; \theta^*)$, $p(\mathbf{x}, t; \theta^*)$

Define $N1$ ▷ Using Eq (3.1), with the input $(\mathbf{x}, t)$ and output $\mathcal{U}(\mathbf{x}, t; \theta_1)$

Define $N2$ ▷ Using Eq (3.1), with the input $(\mathbf{x}, t)$ and output $\mathcal{U}(\mathbf{x}, t; \theta_2)$

Initialize $\theta_1 = \{w_1, b_1\}$,$\theta_2 = \{w_2, b_2\}$ ▷ w_1 and w_2 are weights, b_1 and b_2 are biases

Initialize Optimizer 1 (θ_1)

for epochs = 1 to m **do**

  for $k = 1$ to $K - 1$ **do**

    $A_1^k = w_1^k X_1^{k-1} + b_1^k$ ▷ X_1^0 are the points from the domain $\overline{\Omega} \times [0, T]$ and $b_1 = 0$

    $X_1^k = \sigma(A_1^k)$ ▷ σ is taken as *sigmoid* and A_1^k represents the affine transformations

  end for

  $A_1^K = w_1^K X_1^{K-1} + b_1^K$,

  $X_1^K = A_1^K$,

  $\hat{\mathcal{U}}(\mathbf{x}; \theta_1) = X_1^K$

  Compute $L[\hat{\mathcal{U}}(\mathbf{x}, t; \theta_1)]$ ▷ Using Eq (3.7)

  Compute $\nabla_{\theta_1}(L[\hat{\mathcal{U}}(\mathbf{x}, t; \theta_1)])$ ▷ Backpropagation

  Update $\theta_1 \leftarrow \theta_1 - \eta \frac{\bar{z}}{\sqrt{\bar{l}+\delta}}$ ▷ Through ADAM Eq (4.5)

end for

$\hat{\mathcal{U}}(\mathbf{x}, t; \theta_1) = [\hat{u}_1(\mathbf{x}, t; \theta), \hat{u}_2(\mathbf{x}, t; \theta), \hat{b}_1(\mathbf{x}, t; \theta), \hat{b}_2(\mathbf{x}, t; \theta), \hat{p}(\mathbf{x}, t; \theta)]^T$ ▷ Initial approximation

Set $\hat{\mathcal{U}}^{new}(\mathbf{x}, t; \theta_1) = \hat{\mathcal{U}}(\mathbf{x}, t; \theta_1)$

Initialize Optimizer 2 (θ_2)

Set $\theta_2 = \theta_1$

Define $\tilde{\mathcal{U}}(\mathbf{x}, t; \theta_2) = \hat{\mathcal{U}}^{new}(\mathbf{x}, t; \theta_1)$

Initialize- $r = 1$

Compute $\tilde{\mathcal{U}}(\mathbf{x}, t; \theta_2)$ for $r = 1$ ▷ Using (3.7)

Set $r = 2$ ▷ κ is used for the required accuracy, also preventing overfitting

while $(L[\tilde{\mathcal{U}}(\mathbf{x}, t; \theta_2)] \geq \kappa)$ or $(r \leq l)$ **do**

  for epochs $1 = 1$ to m_1 **do**

    for $k = 1$ to $K - 1$ **do**

      $A_2^k = w_2^k X_2^{k-1} + b_2^k$ ▷ $X_2^0 = X_1^0$ and $b_2 = 0$ for simplicity

      $X_2^k = \sigma(A_2^k)$ ▷ σ is taken as *sigmoid* and A_2^k represents the affine transformations

    end for

    $A_2^K = w_2^K X_2^{K-1} + b_2^K$,

    $X_2^K = A_2^K$,

    $\tilde{\mathcal{U}}(\mathbf{x}, t; \theta_2) = X_2^K$

    $\tilde{\mathcal{U}}(\mathbf{x}, t; \theta_2) = [\tilde{u}_1(\mathbf{x}, t; \theta_2), \tilde{u}_2(\mathbf{x}, t; \theta_2), \tilde{b}_1(\mathbf{x}, t; \theta_2), \tilde{b}_2(\mathbf{x}, t; \theta_2), \tilde{p}(\mathbf{x}, t; \theta_2)]^T$

    Compute $L[\tilde{\mathcal{U}}(\mathbf{x}, t; \theta_2)]$ ▷ Using (3.7)

    Compute $\nabla_{\theta_2}(L[\tilde{\mathcal{U}}(\mathbf{x}, t; \theta_2)])$ through back propagation ▷ Back propagation

    Update $\theta_2 \leftarrow \theta_2 - \eta \frac{\bar{z}}{\sqrt{\bar{l}+\delta}}$ ▷ Through ADAM Eq (4.5)

  end for

  r=r+1

end while

$\theta^* = \theta_2$ ▷ Optimal parameters

$\mathcal{U}(\mathbf{x}, t; \theta^*) = [\tilde{u}_1(\mathbf{x}, t; \theta_2), \tilde{u}_2(\mathbf{x}, t; \theta_2), \tilde{b}_1(\mathbf{x}, t; \theta_2), \tilde{b}_2(\mathbf{x}, t; \theta_2), \tilde{p}(\mathbf{x}, t; \theta_2)]^T$ ▷ Final neural network solution approximation

Definition 1. *Let a function $u = v_\rho$ with some probability measure (ρ) on the domain $\overline{\Omega}$. We define the Barron norm of u on $\overline{\Omega}$ with the index $p \in [1, +\infty]$, the neural parameters $(w_{12i}, w_{22i}, w_{1i}, b_i)$, and the support radius R by*

$$\begin{aligned}\|u\|_{\mathcal{B}_R^p(\Omega)} &= \inf_\rho \left\{ \left(\int |w_2|^p \rho(dw_2, dw_1, db) \right)^{\frac{1}{p}} : u = \int w_2 \sigma(w_1 x + b) \rho(dw_2, dw_1, db) \text{ on } \overline{\Omega}, \right. \\ &\qquad \left. \rho \text{ is supported on } \mathbb{R} \times \overline{B}_R \times \mathbb{R} \right\},\end{aligned}$$

where $\overline{B}_R = \{x \in \overline{\Omega} : \|x\| \le R\}$. The corresponding Barron space is then defined as $\mathcal{B}_R^p(\overline{\Omega}) = \left\{ u : \|u\|_{\mathcal{B}_R^p(\overline{\Omega})} < \infty \right\}$.

We have provided a clear motivation of the Barron space in Section 4. Basically, our aim is to provide the convergence of the neural network solution for the steady-state version of MHD systems. Since the solution of the MHD equation is continuous and lies in the Barron space, see Lemma 1. In particular, the proof of convergence on Barron space uses cos as an activation function [49], which does not seem to be effective for the boundary layer in the present class of problems. Hence, we have provided the convergence based on sigmoid activation functions, which works well experimentally too.

Assumption 1. *Let σ(sigmoid) be an activation function, defined as $\sigma : \mathbb{R} \to \mathbb{R}$ and note that σ is sufficiently smooth. Let, $M_1 := \sup_{x\in\Omega} |\sigma(x)| < \infty$ and $M_{i+1} := \sup_{x\in\Omega} |\sigma^i(x)| < \infty$, for $i = 1, 2$, where $\sigma^i(x)$ represents the i-th derivative of $\sigma(x)$.*

Lemma 1. *Let $\sigma(x)$ be a sigmoid activation function for the proposed neural network in Algorithm 1 and the neural network parameters θ be independently and identically distributed (i.i.d.) random variables drawn from a distribution. Consider a complex-valued function $\xi_{j0}(x) \in L^2(\mathbb{R}) \cap L^1(\mathbb{R})$, in such a way $\{\xi_{j0}(x) := u_{j0}(x)$ for all $x \in \overline{\Omega}$ and 0 for all $x \notin \overline{\Omega}$, and $j = 1, 2\}$ on a domain $\overline{\Omega}$ of the positive side of the x-axis. We also have $\overline{u}_j(x) = Re(\mathcal{F}^{-1}\xi_{j0}(x)) \in L^2(\mathbb{R}) \cap \mathcal{B}_R^2(\mathbb{R})$ and $u_j(x) = Re(\mathcal{F}^{-1}u_{j0}(x)) \in L^2(\overline{\Omega}) \cap \mathcal{B}_R^2(\overline{\Omega})$ for some $0 < R < \infty$, where $\mathcal{F}^{-1}$ denotes the inverse Fourier transformation, and $j = 1, 2$.*

Proof. Utilizing the technique presented in [49] for elliptic problems, we employ the sigmoid activation function to establish the approximation properties within the Barron space's framework. Let u_{j0} be a complex number and consider it to be $u_{j0}(s) = e^{i\theta(s)} F_j(s)$, where $F_j(s) = |u_{j0}(s)|$. It then holds that $\overline{u}_j(x) = Re((\mathcal{F}^{-1}\xi_{j0})(x))$, $x \in \mathbb{R}$, $j = 1, 2$.

Therefore

$$\begin{aligned}
u_j(x) &= Re((\mathcal{F}^{-1}u_{j0})(x)), \quad x \in \overline{\Omega}, \\
&= Re\left(\tfrac{1}{\sqrt{2\pi}} \int e^{isx} u_{j0}(s) ds \right), \\
&= Re\left(\tfrac{1}{\sqrt{2\pi}} \int e^{i(sx+\theta(s))} F_j(s) ds \right), \\
&= \tfrac{1}{\sqrt{2\pi}} \int \cos(sx + \theta(s)) \mu_j(ds), \\
&\le \tfrac{4}{\sqrt{2\pi}} \int \tfrac{1}{1+e^{-(sx+\theta(s))}} \mu_j ds = \tfrac{4}{\sqrt{2\pi}} \int \sigma(sx + \theta(s)) \mu_j(ds), \\
u_j(x) &= \int \tfrac{4w_{j0}}{\sqrt{2\pi}} \tfrac{1}{1+e^{-(sx+\theta(s))}} \overline{\mu}_j(ds), \quad x \in \overline{\Omega}, \\
u_j(x) &= \int \tfrac{w_{j2}}{1+e^{-(sx+\theta(s))}} \overline{\mu}_j(ds), \quad x \in \overline{\Omega}, \quad j = 1, 2,
\end{aligned}$$

where the measure μ_j is defined as $\mu_j(D) = \int_D F_j(s) ds$, for a domain D, $\overline{\mu}_j = \mu_j / w_{j0}$, and $w_{j2} = \frac{4w_{j0}}{\sqrt{2\pi}}$. Since u_{j0} is compactly supported, there is a $R < \infty$ for which u_{j0} is supported on $\overline{B}_R$. By the

aforementioned definition of Barron space, μ_j and $\overline{\mu}_j$ are supported in $\overline{B}_R$. Therefore, ρ is supported in $\mathbb{R} \times \overline{B}_R \times \mathbb{R}$. We then get the bound of $u_j(x)$ in the following manner

$$\|u_j(x)\|^2_{\mathcal{B}^2_R(\overline{\Omega})} \leq \int |w_{j2}|^2 \rho(dw_{j2}, dw_1, db) < C_b \|u_j(x)\|^2_{\mathcal{B}^2_R(\overline{\Omega})} < \infty, \quad C_b > 0, \ j = 1, 2. \tag{4.3}$$

Hence $u_j(x) \in \mathcal{B}^2_R(\overline{\Omega})$. Since $u_j(x) \leq \mathcal{F}^{-1} u_{j0}(x)$, we write $\|u_j(x)\|_{L^2(\overline{\Omega})} \leq \|\mathcal{F}^{-1} u_{j0}(x)\|_{L^2(\overline{\Omega})}$. Using Parseval's identity $\|u_j(x)\|_{L^2(\overline{\Omega})} \leq \|\mathcal{F}^{-1} u_{j0}(x)\|_{L^2(\overline{\Omega})} = \|u_{j0}(x)\|_{L^2(\overline{\Omega})}$, and hence $u_j(x) \in L^2(\overline{\Omega})$.

Theorem 1. *Consider the activation function for the proposed model to be sigmoid, which satisfies Assumption 1, and let the neural network parameters be i.i.d. random variables drawn from a distribution. Let $u_j(x) \in \mathcal{B}^2_R(\overline{\Omega})$. Then for any open set Ω and for any $\tau \in \mathbb{N}$, $\{w_{j2}, w_1, b\}^2_{j=1}$ exist such that the neural network solution satisfies*

$$\|v_j(x;\theta) - u_j(x)\|^2_{L^2(\Omega)} \leq \frac{2M_1^2 \chi(\Omega)}{\tau} \|u_j(x)\|^2_{\mathcal{B}^2_R(\overline{\Omega})}, \tag{4.4}$$

where M_1 is a constant as mentioned in Assumption 1, $\chi(\Omega)$ is the Lebesgue measure of Ω, and τ denotes the number of neurons in the hidden layer.

Proof. We denote the set of parameters θ_j as $\theta_j = \{w_{j2i}, w_{1i}, b\}_{1 \leq i \leq \tau}$ with respect to $\rho_j^{\times\tau}$, C_b denotes a positive constant, and $j = 1, 2$. Here, $\rho_j^{\times\tau}$ denotes the neural parameters that occur τ times. By using the Eq (4.3) and using the parameters as i.i.d. random variables, the difference between the neural network solution and the exact solution $\mathcal{E}_{j1}(x;\theta) = v_j(x;\theta) - u_j(x)$ satisfies the following bound:

$$\begin{aligned}
\mathbb{E}_{\rho^{\times\tau}} \|\mathcal{E}_j\|^2_{L^2(\Omega)} &= \mathbb{E}_{\rho^{\times\tau}} \int_\Omega \left(\tfrac{1}{\tau} \textstyle\sum_{i=1}^{\tau} w_{j2i} \frac{1}{1+e^{(w_{1i}x+b_i)}} - u_j(x) \right)^2 dx, \\
&= \tfrac{1}{\tau^2} \int_\Omega \mathbb{E}_{\rho^{\times\tau}} \left(\textstyle\sum_{i=1}^{\tau} (w_{j2i} \frac{1}{1+e^{(w_{1i}x+b_i)}} - u_j(x)) \right)^2 dx, \\
&= \tfrac{1}{\tau^2} \int_\Omega \mathbb{E}_{\rho^{\times\tau}} \textstyle\sum_{i=1}^{\tau} (w_{j2i} \frac{1}{1+e^{(w_{1i}x+b_i)}} - u_j(x))^2 dx, \\
&= \tfrac{1}{\tau} \int_\Omega \mathbb{E}_{\rho} (w_{j2} \frac{1}{e^{(w_1 x+b)}} - u_j(x))^2 dx, \\
&= \tfrac{1}{\tau} \int_\Omega \mathrm{Var}_{\rho} (w_{j2} \frac{1}{1+e^{(w_1 x+b)}}) dx, \\
&\leq \tfrac{1}{\tau} \int_\Omega \mathbb{E}_{\rho} \left[(w_{j2} \frac{1}{1+e^{(w_1 x+b)}})^2 \right] dx, \\
&\leq \tfrac{M_1^2 \chi(\Omega)}{\tau} \mathbb{E}_{\rho} \left[|w_{j2}|^2 \right], \\
&\leq \tfrac{C_b M_1^2 \chi(\Omega)}{\tau} \|u_j(x)\|^2_{\mathcal{B}^2_R(\overline{\Omega})}.
\end{aligned}$$

Remark 1. *Consider $\beta_1, \beta_2 \in [0, 1)$. Let $\mathcal{E}_{ji}$ be a bounded error function defined over the L^2 norm. Let us use the ADAM optimizer in Algorithm 1 for optimal prediction by minimizing the objective loss function Eq (3.2). A general form of the ADAM optimizer is as follows:*

$$\theta_{i+1} = \theta_i - \eta_i \frac{\overline{z}_i}{\sqrt{\overline{l}_i + \delta}}, \tag{4.5}$$

where $\overline{z}_i = z_i/(1-\beta_1^i)$, $z_i = \beta_1 z_{i-1} + (1-\beta_1)\mathcal{E}_{ji}$, $\overline{l}_i = l_i/(1-\beta_2^i)$, $l_i = \beta_2 l_{i-1} + (1-\beta_2)\mathcal{E}^2_{ji}$, and $\mathcal{E}_{ji}$ denotes the error function at the ith iteration and $j = 1, 2$.

From Eq (4.5)*, the updation of the weight function in the mth iteration is defined by*

$$w_{m+1} = w_m - \eta_m \frac{\beta_1 z_{m-1} + (1-\beta_1)\mathcal{E}_{jm}}{(1-\beta_1^m)\sqrt{\overline{l_m} + \delta}}, \quad j = 1, 2.$$

On the Basis of $z_m = \beta_1 z_{m-1} + (1-\beta_1)\mathcal{E}_{jm}$*, by using all the previous values of* z_m *through each iteration, we get the following form for* $j = 1, 2$ *and* $1 \le i \le m$*:*

$$\bar{z}_m = \frac{1}{(1-\beta_1^m)}\left[\beta_1^i z_{m-i} + \beta_1^i(1-\beta_1)\mathcal{E}_{j(m-i)} + \beta_1^{i-1}(1-\beta_1)\mathcal{E}_{j(m-i+1)} + \cdots + (1-\beta_1)\mathcal{E}_{j(i-1)} + (1-\beta_1)\mathcal{E}_{ji}\right],$$

If $i = m$*, then*

$$\bar{z}_m = \frac{1}{(1-\beta_1^m)}\left[\beta_1^m z_0 + \beta_1^m(1-\beta_1)\mathcal{E}_{j0} + \beta_1^{m-1}(1-\beta_1)\mathcal{E}_{j1} + \cdots + (1-\beta_1)\mathcal{E}_{j(m-1)} + (1-\beta_1)\mathcal{E}_{jm}\right], \quad j = 1, 2.$$

Initially, we use $z_0 = 0$ *in the algorithm. We then get*

$$\bar{z}_m = \frac{(1-\beta_1)}{(1-\beta_1^m)}\left[\beta_1^m \mathcal{E}_{j0} + \beta_1^{m-1}\mathcal{E}_{j1} + \cdots + \mathcal{E}_{j(m-1)} + \mathcal{E}_{jm}\right], \quad j = 1, 2,$$

$$\bar{z}_m \le \frac{(1-\beta_1)\mathcal{E}_{jm_1}}{(1-\beta_1^m)}\left[\beta_1^m + \beta_1^{m-1} + \cdots + 1 + 1\right], \quad 1 \le m_1 \le m,$$

$$\bar{z}_m \le \frac{(1-\beta_1)\mathcal{E}_{jm_1}}{(1-\beta_1^m)}\left[\frac{(1-\beta_1)}{(1-\beta_1^m)} + 1\right].$$

here, $\mathcal{E}_{jm_1}$ *denotes the maximum value of the error function in the iterations below.*

Similarly, $\sqrt{\overline{l}_m} = \sqrt{\frac{\sum_{i=1}^m (1-\beta_2)\beta_2^{m-i}\mathcal{E}_{ji}^2}{(1-\beta_2^m)}}$*. Using Young's inequality,* $\sqrt{\overline{l}_m}$ *can be written as* $\sqrt{\overline{l}_m} \le 1 + \frac{1}{2}\sum_{i=1}^m \mathcal{E}_{ji}^2$*. Then*

$$\|w_{m+1} - w_m\|_{L^2} \le 4|\eta_m|\left|\frac{(1-\beta_1)}{(1-\beta_1^m)}\right|\left\|\frac{\mathcal{E}_{jm_1}}{\sum_{i=1}^m \mathcal{E}_{ji}^2}\right\|_{L^2}, 1 \le m_1 \le m.$$

Hence, we can conclude that, if the iterations and neurons of the hidden layer increase, then $\|w_{m+1} - w_m\|_{L^2}$ *goes to zero as the number of iterations m goes to infinity.*

5. Numerical experiments

We begin with Example 1, whose convection coefficient matrix is either M matrix or not M matrix. Although it will not satisfy the standard maximum principle, it will meet the continuous and discrete form of the non-standard maximum principle involving the sign of the solution's derivatives. Based on the available numerical methods for a priori defined piecewise uniform adaptive Shishkin-type meshes $\{x_i\}_0^N$ [23, 67], which are dense inside the boundary layer regions. For the numerical calculation using the finite difference method, we evaluate the errors in the L^2 norm and the corresponding error and order of accuracy are defined by

$$\mathfrak{E}_j^N = \left(\sum_{i=0}^N \left|\mathbf{v}_{j,i}^N - \mathbf{v}_{j,i}^{2N}\right|^2\right)^{1/2}, \quad \boldsymbol{\sigma}_j^N = \log_2(\mathfrak{E}_j^N/\mathfrak{E}_j^{2N}), \tag{5.1}$$

In the definition above, $\mathbf{v}_{j,i}^{N}$ for $j = 1, 2$ represents the components of the numerical solution obtained with N partitions, while $\mathbf{v}_{j,i}^{2N}$ is the solution computed on the same mesh $\{x_i\}_0^N$, including its midpoints. Although $\mathbf{v}_{j,i}^{2N}$ is computed at the midpoints, the error comparison between $\mathbf{v}_{j,i}^{N}$ and $\mathbf{v}_{j,i}^{2N}$ is performed only at the mesh points $\{x_i\}_0^N$. Moreover, $\mathfrak{E}_j^N$ and $\boldsymbol{\sigma}_j^N$ represent the uniform error and order over all values of ε_1 and ε_2 considered for the numerical experiments. Here, we have taken $\varepsilon_1, \varepsilon_2 \in \{1, 10^{-1}, 10^{-2}, 10^{-3}\}$.

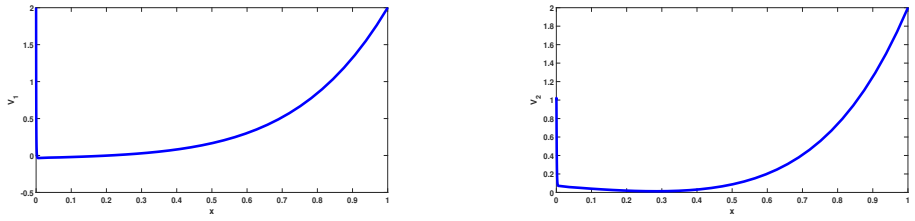
Example 1. *Consider the multiple-scale coupled system (1.8) with the following coupled convection and reaction matrices with an unknown solution:*

$$\mathbf{P}(x) = \begin{bmatrix} p_{11} & -1-2x \\ -1-x & p_{22} \end{bmatrix}, \ \mathbf{Q}(x) = \begin{bmatrix} 10+x^2 & -2 \\ -1 & 4+x \end{bmatrix}, \ \mathbf{F}(x) = \begin{bmatrix} 1+x+3x^2 \\ 2x-1, \end{bmatrix} \quad \text{for } x \in \Omega \equiv (0,1), \tag{5.2}$$

together with the following boundary data

$$\boldsymbol{B}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \ \boldsymbol{B}_2 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}. \tag{5.3}$$

We begin with the convection coefficient matrix is diagonally dominant. Numerically, we have observed that the finite difference approach can capture the non-oscillatory behavior of the boundary layer's adaptive approximation for coupled systems. In addition, we also demonstrate that the numerical method analyzed in [54] performs well (see the L^2 error in Table 2 and Figure 4) in this case.



(a) Numerical velocity $\mathbf{v}_1$ **(b)** Numerical magnetic field $\mathbf{v}_2$

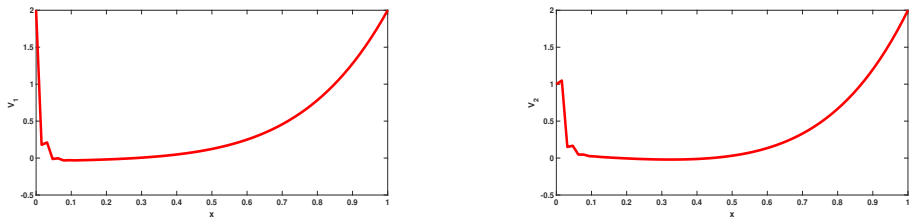
Figure 4. Oscillation-free boundary layer solutions obtained from the numerical scheme for Ex 1, where the diagonal values of the convection matrix are $p_{11} = 4 + xe^x$, $p_{22} = 2 + x^2$, and $\varepsilon_1 = \varepsilon_2 = 10^{-3}, N = 256$ partitions

Table 2. L^2-norm based errors and orders for the numerical approximations of $\mathbf{v}_1$, and $\mathbf{v}_2$, where the diagonal values of the convection matrix are $p_{11} = 4 + xe^x$, and $p_{22} = 2 + x^2$ for Example 1.

N		32	64	128	256	512	1024
$\mathbf{v}_1$	$\mathfrak{E}_1^N$	3.678×10^{-1}	3.489×10^{-1}	3.194×10^{-1}	2.916×10^{-1}	2.809×10^{-1}	2.568×10^{-1}
	$\boldsymbol{\sigma}_1^N$	7.592×10^{-2}	1.274×10^{-1}	1.315×10^{-1}	5.356×10^{-2}	1.294×10^{-1}	—
$\mathbf{v}_2$	$\mathfrak{E}_2^N$	9.235×10^{-1}	9.195×10^{-1}	9.007×10^{-1}	8.594×10^{-1}	7.889×10^{-1}	6.835×10^{-1}
	$\boldsymbol{\sigma}_2^N$	6.187×10^{-3}	2.987×10^{-2}	6.757×10^{-2}	1.235×10^{-1}	2.070×10^{-1}	—
$\max\{\mathbf{v}_1, \mathbf{v}_2\}$	$\mathfrak{E}^N$	9.007×10^{-1}	8.595×10^{-1}	7.890×10^{-1}	6.835×10^{-1}	5.536×10^{-1}	4.310×10^{-1}
	$\boldsymbol{\sigma}^N$	6.757×10^{-2}	1.235×10^{-1}	2.070×10^{-1}	3.040×10^{-1}	3.611×10^{-1}	—

Our next aim is to consider a strongly coupled system where the convection matrix is not an M-matrix, by considering the Example 1 with $p_{ii} = 0$ for $i = 1, 2$ in $\mathbf{P}(x)$ in Eq (5.2). We first show that the present

numerical approach (which works well for Example 1) does not produce a convergent solution for this problem with a nondiagonally dominant convection matrix. Here, one can see that when the scheme proposed in [54] is applied to our problem of interest, it does not converge at all (see Table 3) and fails to provide an oscillation-free solution (see Figure 5), even on boundary layer fitted woith Shishkin-type meshes. However, this numerical method is able to maintain the layer behavior in both component of the solution without spurious oscillations when the convection matrix is an M-matrix.



(a) Numerical velocity $\mathbf{v}_1$ **(b)** Numerical magnetic field $\mathbf{v}_2$

Figure 5. Oscillation occurs through the numerical scheme in the boundary region when the diagonal entries of the convection matrix are $p_{ii} = 0$ for $i = 1, 2$ in $\mathbf{P}(x)$ from Eq (5.2) and $\varepsilon_1 = \varepsilon_2 = 10^{-3}$, with $N = 256$.

Table 3. L^2-norm errors and corresponding orders for the numerical approximations of $\mathbf{v}_1$, and $\mathbf{v}_2$, where the diagonal values of the convection matrix are $p_{ii} = 0$ for $i = 1, 2$ in $\mathbf{P}(x)$ from Eq 5.2.

N		32	64	128	256	512	1024
$\mathbf{v}_1$	$\mathfrak{E}_1^N$	1.255×10^1	9.212×10^1	3.426	6.791	2.385×10^2	1.429×10^1
	$\boldsymbol{\sigma}_1^N$	-2.874	4.748	-9.868×10^{-1}	-5.134	4.060	—
$\mathbf{v}_2$	$\mathfrak{E}_2^N$	1.041×10^1	1.099×10^2	2.867	4.889	2.905×10^2	1.730×10^1
	$\boldsymbol{\sigma}_2^N$	-3.399	5.260	-7.700×10^{-1}	-5.893	4.069	—
$\max\{v_1, v_2\}$	$\mathfrak{E}^N$	1.256×10^1	1.099×10^2	3.427	6.792	2.906×10^2	1.731×10^1
	$\boldsymbol{\sigma}^N$	-3.129	5.0030	-9.868×10^{-1}	-5.419	4.069	—

Now we implement our proposed advanced PINN algorithm to show that this algorithm accurately detects boundary layers for problems where the convection matrix is an M matrix in Figure 6. Observe the convergence of the mean square error (MSE) of the advanced PINN solution in Table 4 as the number of iterations increases.

Table 4. Mean square error of the proposed MI-PINN algorithm over $N = 256$ uniform partitions for Example 1, when the diagonal entries of the convection matrix is $p_{11} = 4 + xe^x$, $p_{22} = 2 + x^2$ in $\mathbf{P}(x)$ and $\epsilon_1 = \epsilon_2 = 10^{-3}$

Iteration (r) :	14	15	16	17
MSE:	1.972×10^{-1}	3.190×10^{-2}	1.334×10^{-2}	6.824×10^{-3}

Hence, the present algorithm works well for those problems for which numerical approximations are already available. We want to note that the errors will decrease up to a certain level once the iterations increase for the present iterative PINN algorithm over uniform partitions. Hence, it is necessary to

propose a boundary layer adaptive algorithm via a PINN, which can identify the layer regions without using prior information about the layers. Therefore, this work not only provides a set of problems that is not trivial to solve numerically; but also proposes an alternative deep learning-based algorithm for these problems.

(a) MI-PINN velocity $\mathbf{v}_1$ **(b)** MI-PINN magnetic field $\mathbf{v}_2$

Figure 6. Oscillation-free boundary layer solutions obtained from Algorithm 1 for Example 1, where the diagonal values of the convection matrix are $p_{11} = 4 + xe^x$, $p_{22} = 2 + x^2$ in $\mathbf{P}(x)$, and $\varepsilon_1 = \varepsilon_2 = 10^{-3}$, with $N = 256$.

In this context, we demonstrate that our proposed PINN approach is effective for Example 1 in both cases, and compute the errors on the basis of the loss function in Eq 3.2. Specifically, one can refer to Table 4 and Figure 6 for the results when the convection matrix is an M matrix. Similarly, Tables 5–7 and Figures 7 and 8 present the results when the convection matrix is not the M matrix, considering the same Example 1 with $p_{ii} = 0$ for $i = 1, 2$ in $\mathbf{P}(x)$ from Eq 5.2. This shows that the present PINN algorithm reduces the errors as the number of iterations increases. One can also see the non-oscillatory behavior with the modified PINN algorithm, which ensures the validity of the proposed algorithm and PINN over numerical approaches.

(a) MI-PINN velocity $\mathbf{v}_1$ **(b)** MI-PINN magnetic field $\mathbf{v}_2$

Figure 7. Solution components of Example 1 through MI-PINN, when the diagonal entries of the convection matrix are $p_{ii} = 0$ for $i = 1, 2$ in $\mathbf{P}(x)$ and $\epsilon_1 = 10^{-1}$, $\epsilon_2 = 10^{-1}$, $N = 64$ uniform partitions.

Table 5. Mean square error of the proposed MI-PINN algorithm over $N = 64$ uniform partitions, when $\epsilon_1 = 10^{-1}$, $\epsilon_2 = 10^{-1}$ where the diagonal entries of the convection matrix are $p_{ii} = 0$ for $i = 1, 2$ in $\mathbf{P}(x)$ for Example 1 and the plot is shown in Figure 7.

Iteration (r) :	1	2	3	4
MSE :	9.476×10^{-1}	3.880×10^{-2}	1.055×10^{-2}	9.661×10^{-3}

Table 6. Divergence behavior of the N1 structure where the diagonal entries of convection matrix are $p_{ii} = 0$ for $i = 1, 2$ in $\mathbf{P}(x)$ for Example 1, $\epsilon_1 = 10^{-4}$ and $\epsilon_2 = 10^{-3}$.

$N1$ Iteration (r)	30	32	34	36
MSE	3.966	3.966	3.966	3.967

(a) MI-PINN velocity $\mathbf{v}_1$ **(b)** MI-PINN magnetic field $\mathbf{v}_2$

Figure 8. Boundary layer adopted solution components through MI-PINN having the diagonal entries of convection matrix is $p_{ii} = 0$ for $i = 1, 2$ in $\mathbf{P}(x)$ for Example 1 and $\epsilon_1 = 10^{-4}$ and $\epsilon_2 = 10^{-3}$.

Table 7. Mean square error of MI-PINN over $N = 256$ uniform partitions, when $\epsilon_1 = 10^{-4}$, and $\epsilon_2 = 10^{-3}$ the diagonal entries of convection matrix are $p_{ii} = 0$ for $i = 1, 2$ in $\mathbf{P}(x)$ for Example 1 and the plot is shown in Figure 8.

Iteration (r) :	15	16	17	18
MSE :	1.485×10^{-1}	5.153×10^{-2}	8.817×10^{-3}	7.736×10^{-3}

We have tested Example 1 on the basis of the MSE error of tanh and the swish action function versus iterations in Table 8, MSE error versus the number of neurons in Table 9, and MSE error versus the number of hidden layer in Table 10.

Table 8. Swish and tanh activation-based MSE by the proposed MI-PINN algorithm over $N = 256$ uniform partitions, when $\epsilon_1 = 10^{-4}$, and $\epsilon_2 = 10^{-3}$ with the diagonal entries of the convection matrix are $p_{ii} = 0$ for $i = 1, 2$ in $\mathbf{P}(x)$ for Example 1, and the plot is shown in Figure 8.

Iteration (r) :	15	16	17	18
MSE (swish):	3.966	3.066	3.466	3.266
MSE (tanh):	1.948×10^{-1}	5.066×10^{-2}	9.587×10^{-3}	9.163×10^{-3}

Table 9. Mean square error versus number of neurons by the proposed MI-PINN algorithm over $N = 256$ uniform partitions, when $\epsilon_1 = 10^{-4}$, and $\epsilon_2 = 10^{-3}$, where the diagonal entries of convection matrix is $p_{ii} = 0$ for $i = 1, 2$ in $\mathbf{P}(x)$ for Example 1.

Neurons (τ) :	50	100	150	200
MSE :	3.946×10^{-1}	1.485×10^{-1}	1.183×10^{-1}	1.008×10^{-1}

Table 10. Mean square error versus number of hidden layer with fix neurons ($\tau = 100$) by the proposed MI-PINN algorithm over $N = 256$ uniform partitions through MI-PINN, when $\epsilon_1 = 10^{-4}$, and $\epsilon_2 = 10^{-3}$, where the diagonal entries of the convection matrix are $p_{ii} = 0$ for $i = 1, 2$ in $\mathbf{P}(x)$ for Example 1.

Hidden layer:	1	2	3	4
MSE :	1.485×10^{-1}	1.338×10^{-1}	1.089×10^{-1}	9.946×10^{-2}

We also want to note that the simple PINN-based architecture $N1$ is not able to detect the boundary layers, as depicted in Figure 9. Hence, modifications are necessary, which is the main aim of this work.

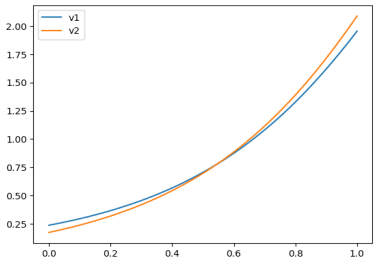


Figure 9. Solution components, velocity ($\mathbf{v}_1$), and magnetic field ($\mathbf{v}_2$) through the $N1$ architecture, when the diagonal entries of the convection matrix are $p_{ii} = 0$ for $i = 1, 2$ in $\mathbf{P}(x)$ in Example 1, and $\epsilon_1 = 10^{-4}$ and $\epsilon_2 = 10^{-3}$. It is clear that the $N1$ structure is unable to predict the boundary layer and boundary conditions accurately when $N = 256$ is considered. It provides the initial guesses of the neural parameters and guesses regarding the location of the boundary layer for our present algorithm. Note that the boundary conditions are not matched through N1.

The remaining examples in this section computationally validate our theoretical justification in Section 3 and demonstrate that the proposed method applies to both relevant and practical scenarios,

including fully nonlinear problems, as provided in Example 2.

Example 2. *Now consider the following nonlinear coupled system, with an unknown solution. The physical representation of* $\mathbf{v}_1$ *denotes the velocity, and* $\mathbf{v}_2$ *represents the magnetic field. Additionally,* ϵ_1 *and* ϵ_2 *control the thickness of the boundary layers:*

$$\begin{cases} -\epsilon_1 \mathbf{v}_1'' + \mathbf{v}_1' - 2\,\mathbf{v}_2' + \mathbf{v}_1^3 - \mathbf{v}_2 = \sin \pi x, & x \in \Omega \equiv (0,1), \\ -\epsilon_2 \mathbf{v}_2'' - 2\,\mathbf{v}_1' + \mathbf{v}_2' - \mathbf{v}_1 + \mathbf{v}_2^3 = e^x, \\ \mathbf{v}_1(x) = 0, \quad \mathbf{v}_2(x) = 0, & x \in \partial\Omega = \{0,1\}, \end{cases}$$

We test the performance of this Algorithm 1 on a non linear problem as the MHD flow is also a non linear problem. Note that the simple architecture based on the PINNs $N1$ is unable to detect the boundary layer and does not satisfy the boundary condition for the nonlinear problem as depicted in Figure 10.

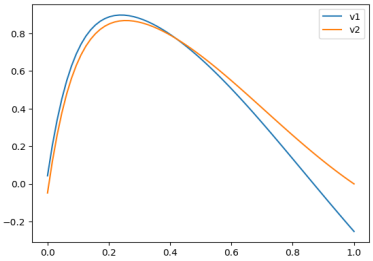


Figure 10. Solution components of the velocity ($\mathbf{v}_1$), and magnetic field ($\mathbf{v}_2$) for the $N1$ architecture, for the coupled nonlinear system in Example 2 with $\epsilon_1 = 10^{-1}$ and $\epsilon_2 = 10^{-1}$ with $N = 64$. Note that the simple PINN is unable to detect the boundary layer accurately. It provides the initial guesses of the neural parameters and guesses about the boundary layer only for the present algorithm.

The non-oscillatory behavior of its solution and the MSE based on the user-defined tolerance while accurately capturing the layer phenomena are provided in Figures 11 and 12 and Tables 11 and 12.

(a) MI-PINN velocity $\mathbf{v}_1$ **(b)** MI-PINN magnetic field $\mathbf{v}_2$

Figure 11. Solution components of the iterative PINN for the Example 2 with $\epsilon_1 = 10^{-1}$ and $\epsilon_2 = 10^{-1}$ over $N = 64$ uniform partitions.

Table 11. Decreasing behavior of MSE by the proposed MI-PINN algorithm for Example 2 when $\epsilon_1 = 10^{-1}$, and $\epsilon_2 = 10^{-1}$ over $N = 64$ uniform partitions.

Iteration (r) :	1	2	3	4
MSE:	1.526×10^{-2}	1.513×10^{-2}	1.477×10^{-3}	9.053×10^{-5}

(a) MI-PINN velocity $\mathbf{v}_1$ **(b)** MI-PINN magnetic field $\mathbf{v}_2$

Figure 12. Boundary layers' adaptive solution components for the iterative PINN for Example 2, when $\epsilon_1 = 10^{-2}$ and $\epsilon_2 = 10^{-3}$ over $N = 256$.

Table 12. Decreasing behavior of MSE of MI-PINN for Example 2 when $\epsilon_1 = 10^{-2}$, and $\epsilon_2 = 10^{-3}$ over $N = 256$.

Iteration (r) :	17	18	19	20
MSE :	6.614×10^{-1}	3.427×10^{-1}	1.038×10^{-1}	7.803×10^{-2}

We test our proposed Algorithm 1 on a 2D steady-state problem with a nondiagonal convection matrix, as demonstrated in Example 3.

Example 3. *We now present a steady-state version of a 2D MHD flow problem with a strongly coupled convection coefficient $\mathcal{P}_1$ that is not an M matrix, $\mathcal{P}_2$ is a reaction matrix, and* $\mathbf{F}$ *is a source function. Here, the velocity and magnetic fields can be considered as the variables* $\mathbf{v}_1$ *and* $\mathbf{v}_2$*. For* $\boldsymbol{x} \in \Omega \equiv (-1, 1)$*, consider*

$$\beta_\epsilon = diag\{\epsilon_1, \epsilon_2\}, \quad \mathcal{P}_1(\boldsymbol{x}) = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}, \quad \mathcal{P}_2(\boldsymbol{x}) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{F}(\boldsymbol{x}) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

along with the boundary condition $\mathbf{v} = 0$*, on* $\partial\Omega$*.*

To analyze the solution, we utilize the contour plots in Figures 13–15 to examine the layer behavior of the velocity and magnetic fields obtained from the neural network. The clustering of contour lines near the boundaries indicates the presence of layer structures in the solution, while the colors in the contour plots represent the predicted output values of the neural network. In particular, one can see that the MSE decreases as the number of iterations increases; see, Tables 13–15. The proposed algorithm has been validated for the two-dimensional MHD flow problem on a uniform partition, where N_x, N_y, and N_t denote the number of divisions in the x-, y-, and t-directions, respectively. The approximated solutions are compared with the corresponding exact solutions and their L^2 error.

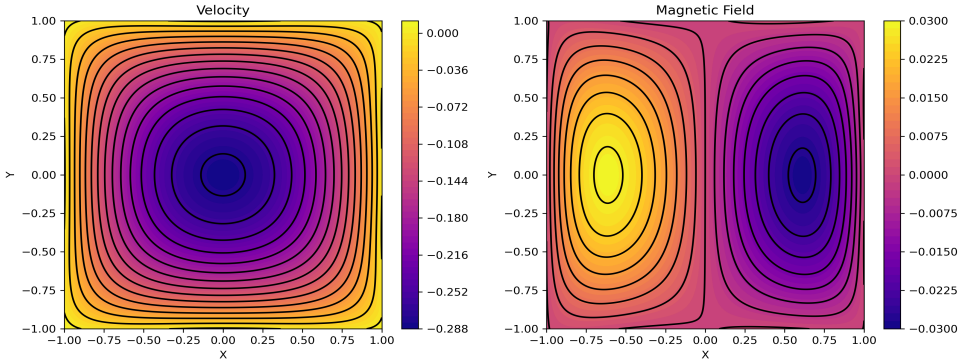


Figure 13. Contour plot of velocity and magnetic field of the MI-PINN when $\epsilon_1 = \epsilon_2 = 1$ for Example 3.

Table 13. Decreasing behavior of MSE for Example 3 when $\epsilon_1 = 1,$ and $\epsilon_2 = 1$ over uniform partitions with $N_x = 16,$ and $N_y = 16$, respectively.

Iteration (r) :	1	10	20	30
MSE :	8.589×10^{-4}	2.529×10^{-4}	2.053×10^{-4}	1.574×10^{-4}

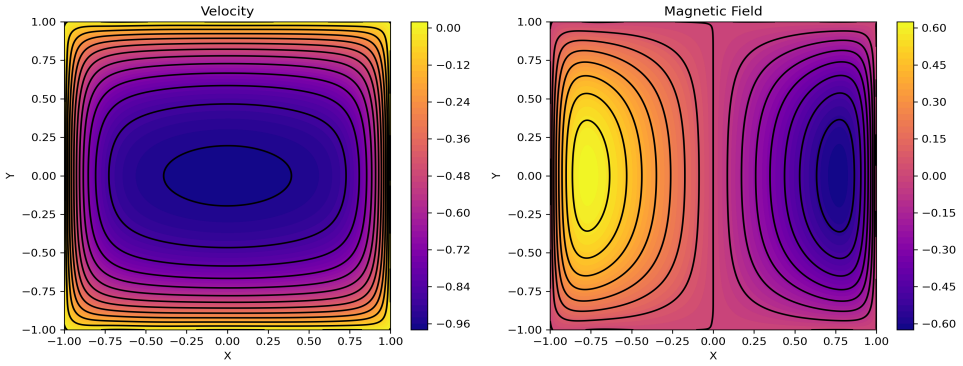


Figure 14. Contour plot of velocity and magnetic field for the MI-PINN when $\epsilon_1 = \epsilon_2 = 10^{-1}$ for Example 3.

Table 14. Decreasing behavior of MSE for Example 3 when $\epsilon_1 = 10^{-1},$ and $\epsilon_2 = 10^{-1}$ over the uniform partitions with $N_x = 16,$ and $N_y = 16$, respectively.

Iteration (r) :	1	10	20	30
MSE :	4.443×10^{-1}	3.104×10^{-3}	1.156×10^{-3}	3.875×10^{-4}

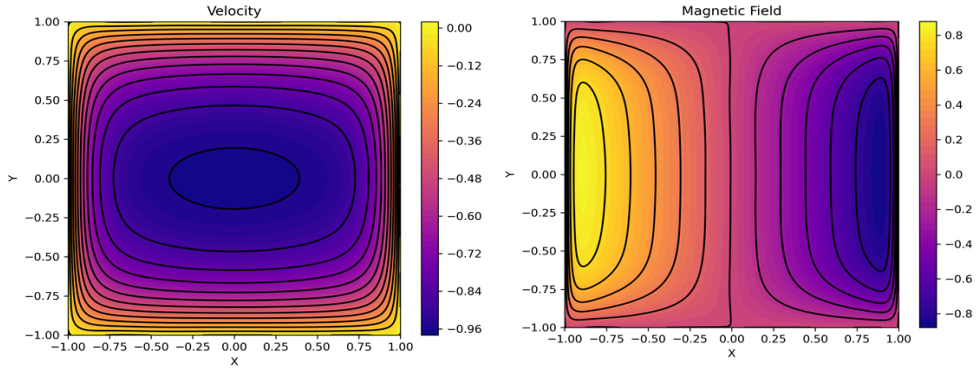


Figure 15. Contour plot of velocity and magnetic field for the MI-PINN when $\epsilon_1 = 10^{-1}$, and $\epsilon_2 = 10^{-2}$ for Example 3.

Table 15. Decreasing behavior of MSE for Example 3 when $\epsilon_1 = 10^{-1}$, and $\epsilon_2 = 10^{-2}$over the uniform partitions with $N_x = 16, N_y = 16$, respectively.

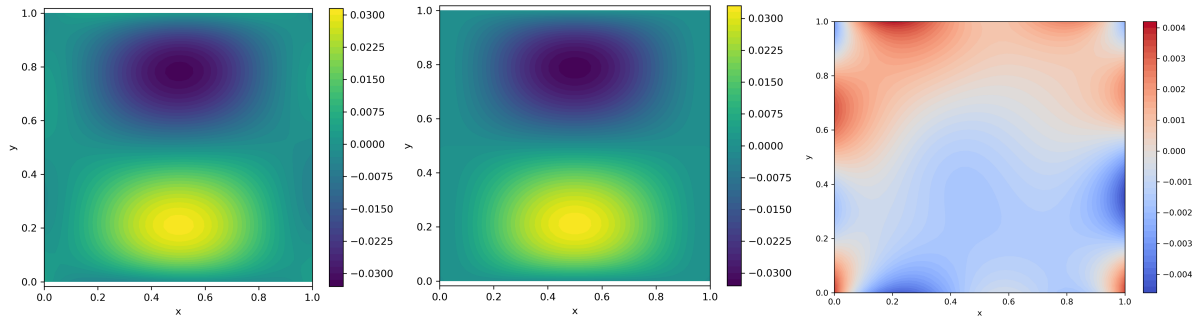
Iteration (r) :	1	10	20	30
MSE :	4.733×10^{-1}	1.167×10^{-2}	4.445×10^{-3}	3.071×10^{-3}

Example 4. *We now consider the original MHD flow equation Eq* (1.11) *that has velocity* $u = [u_1, u_2]^T$ *and magnetic field* $B = [b_1, b_2]^T$.

In this case, we assume $\Omega \in [0, 1] \times [0, 1]$, $T = 1$, *and* $\mu = \epsilon = \sigma = 1$. *Then the exact solution of the problem above can be given as follows:*

$$\begin{cases} u_1(x, y, t) = 10x^2(x-1)^2y(y-1)(2y-1)\cos t, \\ u_2(x, y, t) = -10x(x-1)(2x-1)y^2(y-1)^2\cos t, \\ b_1(x, y, t) = \sin(\pi x)\cos(\pi y)\cos t, \\ b_2(x, y, t) = -\sin(\pi y)\cos(\pi x)\cos t, \\ p(x, y, t) = 10(2x-1)(2y-1)\cos t. \end{cases} \tag{5.4}$$

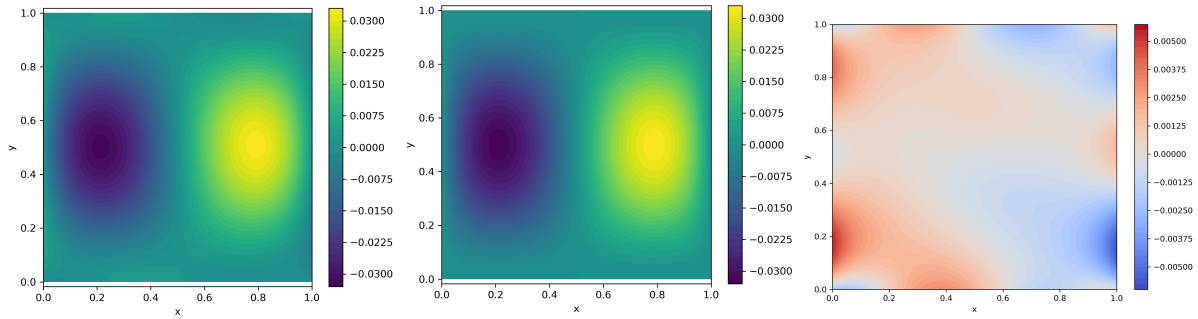
The source functions f, g *and the initial conditions* u_0, b_0 *can be computed using the exact solution.*



(a) Approximate solution of u_1 **(b)** Exact solution of u_1 **(c)** L^2 based error plot of u_1

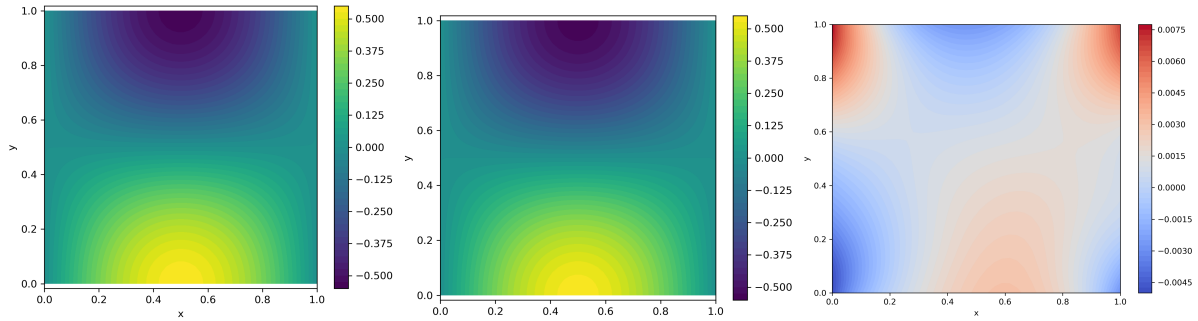
Figure 16. Contour plot of the approximate solution, exact solution, and L^2 error for the MI-PINN at t=1 over uniform partitions with $N_x = 20, N_y = 20$, and $N_t = 10$, respectively.

The contour plot of the approximated solutions, the exact solutions, and L^2 errors are plotted in Figures 16–19. The decreasing behavior of the MSE can be observed in Table 16, based on the number of iterations.



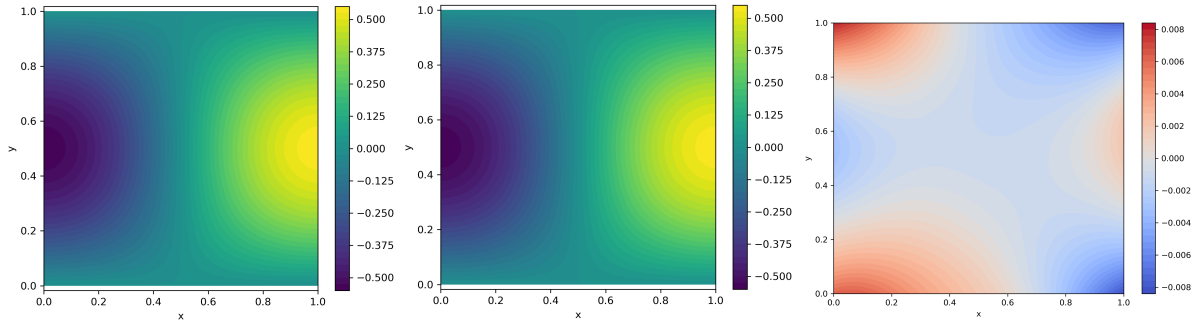
(a) Approximate solution of u_2 **(b)** Exact solution of u_2 **(c)** L^2-based error plot of u_2

Figure 17. Contour plot of the approximate solution, exact solution, and L^2 error for the MI-PINN at t=1 over uniform partitions with $N_x = 20, N_y = 20,$ and $N_t = 10$, respectively.



(a) Approximate solution of b_1 **(b)** Exact solution of b_1 **(c)** L^2-based error plot of b_1

Figure 18. Contour plot of the approximate solution, exact solution, and L^2 error for the MI-PINN at $t = 1$ over uniform partitions having $N_x = 20, N_y = 20,$ and $N_t = 10$, respectively.



(a) Approximate solution of b_2 **(b)** Exact solution of b_2 **(c)** L^2-based error plot of b_2

Figure 19. Contour plot of the approximate solution, exact solution, and L^2 error for the MI-PINN at $t = 1$ over uniform partitions having $N_x = 20, N_y = 20,$ and $N_t = 10$, respectively.

Table 16. Decreasing behavior of MSE for Example 4 when $\epsilon = 1$ over uniform partitions with $N_x = 20, N_y = 20,$ and $N_t = 10$, respectively.

Iteration (r) :	5	10	15	20
MSE :	4.672×10^{-2}	1.587×10^{-2}	4.656×10^{-3}	2.419×10^{-3}

6. Conclusions

We have developed a modified iterative PINN-based deep learning algorithm to solve strongly coupled systems of singularly perturbed boundary value problems originating from the boundary layer. Our work addresses the lack of theoretical analysis and numerical challenges, mainly when the convection matrix does not satisfy the M matrix condition, as frequently encountered in MHD flows. Furthermore, we have explored the limitations of standard PINN models and highlighted the numerical difficulties in detecting boundary layers. We have proved that the exact solution of the defined problems lies in the Barron space with some compact support property under the probability measure and theoretical convergence of the two-layer neural network within the Barron space. Several numerical experiments are performed to verify that the proposed deep learning-based algorithm detects the boundary layers efficiently. The present algorithm performs better than simple PINNs and several existing numerical approaches.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgements

Pratibhamoy Das, wants to acknowledge the Ministry of Education and the Science and Engineering Research Board (SERB), Government of India, for supporting the present research under research grant No. MTR/2021/000797. Arihant Patawari, gratefully acknowledges his institute, Indian Institute of Technology Patna, and is thankful to Joint Council of Scientific and Industrial Research-University Grants Commission (CSIR-UGC), 2021, National Testing Agency (NTA) Reference number 211610137817, New Delhi, India, for their financial support. Shridhar Kumar, gratefully acknowledges his institute, Indian Institute of Science Education and Research, Thiruvananthapuram for providing an institute fellowship.

Conflict of interest

Pratibhamoy Das is a guest editor for NHM and was not involved in the editorial review or the decision to publish this article. All authors declare that there are no competing interests.

Author contributions

All authors have contributed equally to this research.

Data availability statement

The data sets generated during and/or analyzed for the current study are available in this manuscript. No extra data were used for this research.

References

1. T. Chen, Z. Ren, G. Lin, Z. Wu, B. L. Ye, Real-time computational optimal control of an MHD flow system with parameter uncertainty quantification, *J. Frankl. Inst.*, **357** (2020), 2830–2850. https://doi.org/10.1016/j.jfranklin.2019.12.013
2. S. Wu, B. Peng, Z. Tian, Exponential compact ADI method for a coupled system of convection-diffusion equations arising from the 2D unsteady magnetohydrodynamic MHD flows, *Appl. Numer. Math.*, **146** (2019), 89–122. https://doi.org/10.1016/j.apnum.2019.07.003
3. T. Chen, Z. Ren, G. Lin, C. Xu, Learning-PDE-Based approximate optimal control for an MHD system with uncertainty quantification, *IEEE Trans. Syst. Man Cybern. Syst.*, **52** (2022), 7185–7192. https://doi.org/10.1109/TSMC.2022.3152505
4. C. G. Campbell, Magnetohydrodynamics in binary stars, in *Astrophysics and Space Science Library* Springer Cham, (2018), XVII, 474. https://doi.org/10.1007/978-3-319-97646-4 .
5. E. N. Parker, *Cosmical Magnetic Fields: Their Origin and Their Activity*, Oxford University Press, Oxford, UK, 1979. Available from: https://books.google.co.in/books?id=rgCvDwAAQBAJ.
6. P. A. Davidson, An introduction to magnetohydrodynamics, in *Cambridge Texts in Applied Mathematics*, Cambridge University Press, Cambridge, UK, (2001). https://doi.org/10.1017/CBO9780511626333
7. R. Moreau, Magnetohydrodynamics, in *Fluid Mechanics and Its Applications*, Springer Dordrecht, The Netherlands, (1990), XIV, 320. https://doi.org/10.1007/978-94-015-7883-7
8. P. W. Hsieh, S. Y. Yang, Two new upwind difference schemes for a coupled system of convection–diffusion equations arising from the steady MHD duct flow problems, *J. Comput. Phys.*, **229** (2010), 9216–9234. https://doi.org/10.1016/j.jcp.2010.08.034
9. P. Das, S. Rana, J. Vigo-Aguiar, Higher order accurate approximations on equidistributed meshes for boundary layer originated mixed type reaction diffusion systems with multiple scale nature, *Appl. Numer. Math.*, **148** (2020), 79–97. https://doi.org/10.1016/j.apnum.2019.08.028
10. T. Linß, Layer-Adapted meshes for reaction-convection-diffusion problems, in *Lecture Notes in Mathematics*, Springer Berlin, Heidelberg, Germany, (2010), XII, 326. https://doi.org/10.1007/978-3-642-05134-0
11. M. K. Kadalbajoo, K. C. Patidar, Singularly perturbed problems in partial differential equations: A survey, *Appl. Math. Comput.*, **134** (2003), 371–429. https://doi.org/10.1016/S0096-3003(01)00291-0
12. S. Saini, P. Das, S. Kumar, Parameter uniform higher order numerical treatment for singularly perturbed robin type parabolic reaction-diffusion multiple scale problems with large delay in time, *Appl. Numer. Math.*, **196** (2024), 1–21. https://doi.org/10.1016/j.apnum.2023.10.003

13. S. Arslan, M. Tezer-Sezgin, Exact and FDM solutions of 1D MHD flow between parallel electrically conducting and slipping plates, *Adv. Comput. Math.*, **45** (2019), 1923–1938. https://doi.org/10.1007/s10444-019-09669-x

14. A. Ayet, B. Chapron, The dynamical coupling of wind-waves and atmospheric turbulence: A review of theoretical and phenomenological models, *Bound. Layer Meteorol.*, **183** (2022), 1–33. https://doi.org/10.1007/s10546-021-00666-6

15. N. Madden, M. Stynes, G. Thomas, On the application of robust numerical methods to a complete flow wave current model. Boundary and Interior Layers ONERA, in *Proceedings of BAIL*, 2004. https://doi.org/10.13025/15121

16. M. S. Gockenbach, Understanding and implementing the finite element method, in *Other Titles in Applied Mathematics*, SIAM, 2006. https://doi.org/10.1137/1.9780898717846

17. V. Saw, P. Das, H. M. Srivastava, Investigations of a class of liouville-caputo fractional order pennes bioheat flow partial differential equations through orthogonal polynomials on collocation points, *Bull. Sci. Math.*, **201** (2025), 103637. https://doi.org/10.1016/j.bulsci.2025.103637

18. K. Zhuang, Z. Du, X. Lin, Solitary waves solutions of singularly perturbed higher-order kdV equation via geometric singular perturbation method, *Nonlinear Dyn.*, **80** (2015), 629–635. https://doi.org/10.1007/s11071-015-1894-7

19. Aakansha, S. Kumar, P. Das, Analysis of an efficient parameter uniform domain decomposition approach for singularly perturbed gierer-meinhardt type nonlinear coupled systems of parabolic problems, *Int. J. Numer. Anal. Model.*, **22** (2025), 459–482. https://doi.org/10.4208/ijnam2025-1020

20. T. Apel, G. Lube, Anisotropic mesh refinement for a singularly perturbed reaction diffusion model problem, *Appl. Numer. Math.*, **26** (1998), 415–433. https://doi.org/10.1016/S0168-9274(97)00106-2

21. C. Kuehn, Multiple time scale dynamics, in *Applied Mathematical Sciences*, Springer Cham, (2015), XIII, 814. https://doi.org/10.1007/978-3-319-12316-5

22. S. Kumar, P. Das, K. Kumar, Adaptive mesh based efficient approximations for darcy scale precipitation–dissolution models in porous media, *Int. J. Numer. Methods Fluids*, **96** (2024), 1415–1444. https://doi.org/10.1002/fld.5294

23. S. Kumar, I. Rosey, P. Das, Impact of mixed boundary conditions and non-smooth data on layer originated non-premixed combustion problems: Higher order convergence analysis, *Stud. Appl. Math.*, **96** (2024), e12763. https://doi.org/10.1111/sapm.12763

24. M. Raissi, P. Perdikaris, G. Em Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.*, **378** (2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

25. A. Arzani, K. W. Cassel, R. M. D'Souza, Theory-guided physics-informed neural networks for boundary layer problems with singular perturbation, *J. Comput. Phys.*, **473** (2023), 111768. https://doi.org/10.1016/j.jcp.2022.111768

26. S. Rezaei, A. Harandi, A. Moeineddin, B. X. Xu, S. Reese, A mixed formulation for physics-informed neural networks as a potential solver for engineering problems in heterogeneous domains: Comparison with finite element method, *Comput. Methods Appl. Mech. Engrg.*, **401** (2022), 115616. https://doi.org/10.1016/j.cma.2022.115616

27. S. Yadav, S. Ganesan, Artificial neural network-augmented stabilized finite element method, *J. Comput. Phys.*, **499** (2024), 112702. https://doi.org/10.1016/j.jcp.2023.112702

28. A. Patawari, P. Das, From traditional to computationally efficient scientific computing algorithms in option pricing: Current progresses with future directions, *Arch. Comput. Methods Eng.*, (2025).

29. S. Kumar, P. Das, A uniformly convergent analysis for multiple scale parabolic singularly perturbed convection-diffusion coupled systems: Optimal accuracy with less computational time, *Appl. Numer. Math.*, **207** (2025), 534–557. https://doi.org/10.1016/j.apnum.2024.09.020

30. M. Raissi, P. Perdikaris, G. Em Karniadakis, Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, preprint, 2017. https://doi.org/10.48550/arXiv.1711.10561

31. M. Raissi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, *J. Mach. Learn. Res.*, **19** (2018), 1–24. Available from: http://jmlr.org/papers/v19/18-046.html.

32. S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics–informed neural networks: Where we are and what's next, *J. Sci. Comput.*, **92** (2022), 88. https://doi.org/10.1007/s10915-022-01939-z

33. A. Beguinet, V. Ehrlacher, R. Flenghi, M. Fuente, O. Mula, A. Somacal, Deep learning-based schemes for singularly perturbed convection-diffusion problems, *ESAIM: Proc. Surv.*, **73** (2023), 48–67. https://doi.org/10.1051/proc/202373048

34. C. Banerjee, K. Nguyen, C. Fookes, K. George, Physics-informed computer vision: A review and perspectives, *ACM Comput. Surv.*, **57** (2024), 1–38. https://doi.org/10.1145/3689037

35. G. Cybenko, Approximation by superpositions of a sigmoidal function, *Math. Control Signals Syst.*, **2** (1989), 303–314. https://doi.org/10.1007/BF02551274

36. Y. Shin, J. Darbon, G. Em Karniadakis, On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs, *Commun. Comput. Phys.*, **28** (2020), 2042–2074. https://doi.org/10.4208/cicp.OA-2020-0193

37. S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating PDEs, *IMA J. Numer. Anal.*, **43** (2023), 1–43. https://doi.org/10.1093/imanum/drab093

38. T. De Ryck, A. D. Jagtap, S. Mishra, Error estimates for physics-informed neural networks approximating the Navier–Stokes equations, *IMA J. Numer. Anal.*, **44** (2024), 83–119. https://doi.org/10.1093/imanum/drac085

39. A. D. Jagtap, K. Kawaguchi, G. Em Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, *J. Comput. Phys.*, **404** (2020), 109136. https://doi.org/10.1016/j.jcp.2019.109136

40. J. Linghu, W. Gao, H. Dong, Y. Nie, Higher-order multi-scale physics-informed neural network (HOMS-PINN) method and its convergence analysis for solving elastic problems of authentic composite materials, *J. Comput. Appl. Math.*, **456** (2025), 116223. https://doi.org/10.1016/j.cam.2024.116223

41. Y. L. Cun, A theoretical framework for back-propagation, *Phy. Eng.*, **1**, (1988), 21–28. Avaiable from: https://api.semanticscholar.org/CorpusID:16775098.

42. A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, preprint, 2021. https://doi.org/10.48550/arXiv.2109.01050

43. G. M. Gie, Y. Hong, C. Y. Jung, Semi-analytic PINN methods for singularly perturbed boundary value problems, *Appl. Anal.*, **103** (2024), 2554–2571. https://doi.org/10.1080/00036811.2024.2302405

44. E. Kharazmi, Z. Zhang, G. Em Karniadakis, hp-VPINNs: Variational physics-informed neural networks with domain decomposition, *Comput. Methods Appl. Mech. Engrg.*, **374** (2021), 113547. https://doi.org/10.1016/j.cma.2020.113547

45. L. Wang, L. Zhang, G. He, Chien-physics-informed neural networks for solving singularly perturbed boundary-layer problems, *Appl. Math. Mech.*, **45** (2024), 1467–1480. https://doi.org/10.1007/s10483-024-3149-8

46. P. D. Miller, Applied asymptotic analysis, *Amer. Math. Soc.*, **75** (2006). http://doi.org/10.1090/gsm/075

47. E. Kharazmi, Z. Zhang, G. Em Karniadakis, Variational physics-informed neural networks for solving partial differential equations, preprint, 2019. https://doi.org/10.48550/arXiv.1912.00873

48. R. Khodayi-Mehr, M. Zavlanos, VarNet: Variational neural networks for the solution of partial differential equations, *Proc. Machine Learning Res.*, **120** (2020), 1–10. Available from: http://proceedings.mlr.press/v120/khodayi-mehr20a/khodayi-mehr20a.pdf.

49. W. N. E, C. Ma, L. Wu, The Barron space and the flow-induced function spaces for neural network models, *Constr. Approximation*, **55** (2022), 369–406. https://doi.org/10.1007/s00365-021-09549-y

50. G. M. Gie, Y. Hong, C. Y. Jung, D. Lee, Singular Layer physics-informed neural network for convection-dominated boundary layer problems in 2D, preprint, 2023. https://doi.org/10.48550/arXiv.2312.03295

51. X. Chu, X. Shi, D. Shi, Unconditional superconvergence analysis of low-order conforming mixed finite element method for time-dependent incompressible MHD equations, *Commun. Nonlinear Sci. Numer. Simul.*, **143** (2025), 108627. https://doi.org/10.1016/j.cnsns.2025.108627

52. C. Clavero, J. Jorge, A splitting uniformly convergent method for one-dimensional parabolic singularly perturbed convection-diffusion systems, *Appl. Numer. Math.*, **183** (2023), 317–332. https://doi.org/10.1016/j.apnum.2022.09.012

53. M. Brdar, S. Franz, L. Ludwig, H. G. Roos, Numerical analysis of a singularly perturbed convection-diffusion problem with shift in space, *Appl. Numer. Math.*, **186** (2023), 129–142. https://doi.org/10.1016/j.apnum.2023.01.003

54. E. O'Riordan, M. Stynes, Numerical analysis of a strongly coupled system of two singularly perturbed convection-diffusion problems, *Adv. Comput. Math.*, **30** (2009), 101–121. https://doi.org/10.1007/s10444-007-9058-z

55. H. G. Roos, M. Stynes, L. Tobiska, *Robust Numerical Methods for Singularly Perturbed Differential Equations: Convection-Diffusion-Reaction and Flow Problems*, Springer Berlin, Heidelberg, 2008. https://doi.org/10.1007/978-3-540-34467-4

56. T. Linß, Analysis of an upwind finite-difference scheme for a system of coupled singularly perturbed convection-diffusion equations, *Computing*, **79** (2007), 23–32.https://doi.org/10.1007/s00607-006-0215-x
57. T. Linß, M. Stynes, Numerical solution of systems of singularly perturbed differential equations, *Comput. Methods Appl. Math.*, **9** (2009), 165–191. https://doi.org/10.2478/cmam-2009-0010
58. G. Beckett, J. Mackenzie, On a uniformly accurate finite difference approximation of a singularly perturbed reaction-diffusion problem using grid equidistribution, *Appl. Numer. Math.*, **35** (2001), 87–109. https://doi.org/10.1016/S0377-0427(00)00260-0
59. G. I. Shishkin, Mesh approximation of singularly perturbed boundary-value problems for systems of elliptic and parabolic equations, *Comp. Math. Math. Phys.*, **35** (1995), 429–446. https://dl.acm.org/doi/abs/10.5555/214478.214484
60. P. Das, J. Vigo-Aguiar, Parameter uniform optimal order numerical approximation of a class of singularly perturbed system of reaction-diffusion problems involving a small perturbation parameter, *J. Comput. Appl. Math.*, **354** (2019), 533–544. https://doi.org/10.1016/j.cam.2017.11.026
61. K. Anukiruthika, P. Muthukumar, P. Das, Semigroup-based theoretical prospects on optimal control of stochastic second-order gurtin–pipkin integro-differential equations, *Optimization*, (2025), 1–18. https://doi.org/10.1080/02331934.2025.2534119
62. H. G. Roos, Special features of strongly coupled systems of convection-diffusion equations with two small parameters, *Appl. Math. Lett.*, **25** (2012), 1127–1130. https://doi.org/10.1016/j.aml.2012.02.018
63. T. Linß, N. Madden, Analysis of an alternating direction method applied to singularly perturbed reaction-diffusion problems, *Int. J. Numer. Anal. Model*, **7** (2010).
64. H. G. Roos, C. Reibiger, Analysis of a strongly coupled system of two convection-diffusion equations with full layer interaction, *J. Appl. Math. Mech.*, **91** (2011), 537–543. https://doi.org/10.1002/zamm.201000153
65. N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, et al., On the spectral bias of neural networks, preprint, 2019. https://doi.org/10.48550/arXiv.1806.08734
66. A. R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, *IEEE Trans. Inform. Theory*, **39** (2002), 930–945. https://doi.org/10.1109/18.256500
67. D. Sarkar, S. Kumar, P. Das, H. Ramos, Higher order convergence analysis for interior and boundary layers in a semi-linear reaction-diffusion system networked by a k-star graph with non-smooth source terms, *Networks Heterog. Media*, **19** (2024), 1085–1115. https://doi.org/10.3934/nhm.2024048

AIMS Press