Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 10(8): 18156–18182.
DOI: 10.3934/math.2025809
Received: 18 June 2025
Revised: 22 July 2025
Accepted: 31 July 2025
Published: 11 August 2025

Research article

Spectral analysis of reaction-diffusion systems via physics-informed neural networks

Burhan Bezekci*
Department of Electrical-Electronics Engineering, Faculty of Engineering and Architecture, Kilis 7 Aralik University, Kilis, Turkey
* **Correspondence:** Email: burhanbezekci@kilis.edu.tr.

Abstract: Reaction–diffusion systems serve as a powerful mathematical framework for modeling dynamic behaviors and pattern formation, widely used to model physical, chemical, and biological systems, among others. Understanding their stability often involves examining eigenvalue problems derived by linearizing the governing equations around their time-independent solutions. In most practical cases, analytical solutions to these eigenvalue problems are difficult to obtain—especially in multi-component systems or when dealing with non-self-adjoint problems—making both analysis and computation more challenging. As a result, numerical methods remain the primary practical tool for investigating their spectral properties and gaining insight into system behavior. Recently, physics-informed neural networks (PINNs) have emerged as a promising alternative for solving partial differential equations by embedding physical laws and constraints directly into the training process. In this work, we use a PINN-based approach to compute multiple eigenpairs for the eigenvalue problems of reaction–diffusion systems, covering both single- and multi-component cases. This includes challenging scenarios involving non-self-adjoint problems, where traditional numerical methods often become less effective. In this work, we limit our focus to two one-dimensional reaction–diffusion models: the Zeldovich–Frank–Kamenetsky (ZFK) equation, a single-component system with a self-adjoint structure, and the FitzHugh–Nagumo (FHN) system, a two-component model exhibiting non-self-adjoint behavior. By embedding essential physical constraints—such as biorthonormality between left and right eigenfunctions and spectral ordering—directly into the loss function, the method maintains consistency and robustness during training. In both cases, the PINN framework yields accurate eigenvalue and eigenfunction approximations that agree closely with direct numerical simulations. These results highlight the capability of PINNs to serve as a flexible and effective tool for spectral analysis across a broad class of reaction–diffusion problems.

Keywords: PINN; spectral analysis; reaction–diffusion; eigenvalue problems; Zeldovich–Frank–Kamenetsky; FitzHugh–Nagumo; adjoint; non-self-adjoint
Mathematics Subject Classification: 35A05, 65N25, 68T07

1. Introduction

Reaction–diffusion systems (RDS) describe how the concentration of substances changes over time and space due to a combination of local reactions and diffusion. Although these models were first developed to study chemical processes, they are now widely used across many fields—including biology, physics, ecology, and the social sciences—because they help explain how complex patterns and behaviors can emerge from simple rules [1, 2]. RDS can be grouped into two main types. Single-component systems usually describe simpler processes like how a population spreads, how heat moves through a material, or how combustion fronts form. Multi-component systems, on the other hand, can produce more complex behaviors, including traveling waves, spatial patterns, spiral formations, and localized structures such as solitons [3, 4]. Thanks to their flexibility and the wide range of dynamics they can capture, reaction–diffusion systems continue to be an important tool for researchers in many scientific areas.

One important area of research in reaction–diffusion systems is solving eigenvalue problems. These problems matter because they help us understand whether steady states are stable, how patterns start to form, and when a system might go through big changes, like a bifurcation. Unlike basic algebraic eigenvalue problems, the ones in reaction–diffusion models may involve differential equations with spatially varying coefficients, boundary conditions, and sometimes even singularities. Because of these challenges, exact solutions are usually only possible in very simplified situations. That is why researchers often use asymptotic techniques, numerical methods, or a mix of both to study these problems and make progress.

To find approximate eigenvalues and eigenfunctions in complex systems, researchers often use classical numerical methods like finite difference and finite element techniques [5, 6]. Finite difference methods are easy to set up and understand, which makes them a popular choice. But they can sometimes miss important features, like keeping the system self-adjoint, which might lead to problems like eigenvectors that are not orthogonal. Finite element methods usually handle tricky or irregular domains better and tend to give more accurate results. However, setting them up takes more time and usually requires some background knowledge about the domain and boundary conditions. One downside of both methods is that they only give numerical results, not exact formulas. That can be a problem if you need symbolic expressions—for example, to take derivatives or study how the solution changes with different parameters.

The limits of traditional numerical methods—like the ones mentioned earlier—plus issues like scalability and high computational cost, have led researchers to explore data-driven and hybrid modeling approaches. Thanks to recent progress in scientific machine learning, there are now promising ways to handle these challenges by combining machine learning with well-established physics-based models. One of the standout methods in this area is physics-informed neural networks (PINNs). These networks are especially useful because they can build physical laws right into the learning process [7, 8]. Unlike models that rely only on data, PINNs use known physical rules, which means they need much less data-just boundary and initial conditions in many cases. This setup helps them simulate complex systems more efficiently while still respecting important physical principles like conservation laws, symmetries, and other built-in constraints. Since the introduction of PINNs, many studies have focused on improving their performance. Recent works offer complementary methodologies that enhance training stability and accuracy in solving high-order or convection-dominated PDEs. For example, a Fourier-feature induced physics-informed randomized neural network method has been proposed for high-order operators relevant to spectral analysis [9], and a PINN framework combining soft and hard boundary constraints with Fourier expansions has been introduced for advection-diffusion problems [10]. These developments reflect the ongoing efforts to refine and extend the original PINN framework.

Even though PINNs show a lot of promise, using them for eigenvalue problems—whether in self-adjoint Sturm–Liouville cases or in systems with non-self-adjoint linearized operators—is still a developing area of research. In this study, we look at both types of problems and introduce a PINN-based framework to tackle eigenvalue problems that come from reaction–diffusion systems. While PINNs have already been used in several applications, their use in non-self-adjoint problems is still fairly limited, which leaves room for more exploration. Our approach directly approximates both the eigenvalues and eigenfunctions, offering a unified way to study stability while aiming to boost both computational accuracy and efficiency.

What makes our approach distinctive is the way we incorporate the spectral structure of the problem into the training process itself. In particular, we enforce biorthonormality and biorthogonality conditions between left and right eigenfunctions—relationships that naturally arise in non-self-adjoint settings but are often ignored in existing PINN-based methods. We also introduce a strategy to guide the network toward separating the dominant eigenmodes from the rest of the spectrum, helping to stabilize the training and clarify the spectral interpretation of the results. In addition, incorporating a priori knowledge—such as symmetry properties of the eigenfunctions or expected spectral behavior—has the potential to further improve training efficiency and solution quality. These structural insights act as useful inductive biases, steering the model toward more meaningful representations and faster convergence. To the best of our knowledge, this is the first study to integrate biorthogonality constraints and spectral separation into a unified PINN framework for computing multiple eigenpairs in reaction–diffusion systems.

The remainder of this paper is structured as follows: In Section 2, we go over the mathematical setup for the self-adjoint eigenvalue problems that play a key role in reaction–diffusion systems. Section 3 introduces our PINN-based approach, including the neural network design, how we define the loss functions, and how the model is trained. In Section 4, we test the method on two well-known examples—the Zeldovich-Frank–Kamenetsky (ZFK) equation and the FitzHugh–Nagumo (FHN) model—to demonstrate its effectiveness. Finally, Section 5 concludes with a summary of our findings and outlines directions for future research.

2. Problem statement

As stated before, the reaction-diffusion equation is a fundamental mathematical model used to describe various physical, chemical, and biological phenomena. In its general form, the equation is written as

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{D}\frac{\partial^2 \mathbf{u}}{\partial x^2} + \mathbf{f}(\mathbf{u}), \tag{2.1}$$

where $\mathbf{u} : \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}^n$ is an n-component field, with $n \geq 1$, defined over space $x \in \mathbb{R}$ and time $t \in \mathbb{R}^+$. Here, $\mathbf{D} \in \mathbb{R}^{n\times n}$ is the diffusivity matrix, and $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is the nonlinear vector function describing the reaction kinetics. We rewrite equation (2.1) in a moving frame traveling at constant speed c, i.e.,

$\mathbf{u} = \tilde{\mathbf{u}}(\xi, \tau)$, $\xi = x - ct$, $\tau = t$ so that (2.1) becomes

$$\frac{\partial \tilde{\mathbf{u}}}{\partial \tau} = \mathbf{D}\frac{\partial^2 \tilde{\mathbf{u}}}{\partial \xi^2} + c\frac{\partial \tilde{\mathbf{u}}}{\partial \xi} + \mathbf{f}(\tilde{\mathbf{u}}). \tag{2.2}$$

In this moving frame of reference, there exists a special type of solution—called the critical solution—that plays a key role in determining the threshold between propagation and decay in reaction-diffusion systems. This nontrivial solution takes different forms depending on the system. In one-component models, it usually appears as a stationary profile known as *critical nucleus*. In multicomponent systems, it often emerges as a moving structure, such as a *critical pulse* or *critical front* [11, 12]. Mathematically, the critical solution satisfies the following ordinary differential equation

$$\mathbf{D}\frac{d^2 \hat{\mathbf{u}}}{d\xi^2} + c\frac{d\hat{\mathbf{u}}}{d\xi} + \mathbf{f}(\hat{\mathbf{u}}) = \mathbf{0}. \tag{2.3}$$

We now linearize Eq (2.2) around the critical solution $\hat{\mathbf{u}}(\xi)$ by substituting $\tilde{\mathbf{u}}(\xi, \tau) = \hat{\mathbf{u}}(\xi) + \mathbf{v}(\xi, \tau)$ and retaining linear terms in $\mathbf{v}(\xi, \tau)$. This yields the following linearized equation

$$\frac{\partial \mathbf{v}}{\partial \tau} = \mathbf{D}\frac{\partial^2 \mathbf{v}}{\partial \xi^2} + c\frac{\partial \mathbf{v}}{\partial \xi} + \mathbf{J}(\hat{\mathbf{u}})\mathbf{v}, \tag{2.4}$$

where $\mathbf{J}(\hat{\mathbf{u}})$ represents the matrix of partial derivatives of the reaction term, evaluated at $\hat{\mathbf{u}}$, given by $\mathbf{J}(\hat{\mathbf{u}}) = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\Big|_{\mathbf{u}=\hat{\mathbf{u}}}$. We assume the linearized equation (2.4) admits solutions of the form

$$\mathbf{v}(\xi, \tau) = e^{\lambda\tau}\mathbf{V}(\xi), \tag{2.5}$$

where $\mathbf{V}(\xi)$ is an eigenfunction and λ is the corresponding eigenvalue. Substituting this ansatz into (2.4) yields the eigenvalue problem

$$\lambda\mathbf{V} = \mathbf{D}\frac{d^2 \mathbf{V}}{d\xi^2} + c\frac{d\mathbf{V}}{d\xi} + \mathbf{J}(\hat{\mathbf{u}})\mathbf{V}. \tag{2.6}$$

This can be expressed more compactly as

$$\mathcal{L}\mathbf{V} = \lambda\mathbf{V}, \tag{2.7}$$

where the linear differential operator $\mathcal{L}$ is defined by

$$\mathcal{L} = \mathbf{D}\frac{d^2}{d\xi^2} + c\frac{d}{d\xi} + \mathbf{J}(\hat{\mathbf{u}}). \tag{2.8}$$

The properties of $\mathcal{L}$—specifically, whether it is *self-adjoint* or *non-self-adjoint*—play a crucial role in analyzing the eigenvalue problem. An operator is *self-adjoint* if it satisfies

$$\langle \mathcal{L}\mathbf{V}_1, \mathbf{V}_2 \rangle = \langle \mathbf{V}_1, \mathcal{L}\mathbf{V}_2 \rangle \tag{2.9}$$

for all $\mathbf{V}_1, \mathbf{V}_2$ in the appropriate function space, where $\langle \cdot, \cdot \rangle$ denotes the inner product. If this condition is not met, the operator is referred to as *non-self-adjoint*.

In the current formulation, the presence of the first-derivative term $c\frac{d}{d\xi}$ and the non-symmetric Jacobian matrix $\mathbf{J}(\hat{\mathbf{u}})$ typically renders $\mathcal{L}$ non-self-adjoint. However, when the system is scalar, i.e., $\mathbf{u} = u$, the wave speed c vanishes due to translational symmetry, and the Jacobian reduces to a scalar function, which is trivially symmetric. In such cases, the operator $\mathcal{L}$ becomes self-adjoint. In general, though, non-self-adjoint operators arise frequently in stability analyses of nonlinear waves and require specialized techniques—such as adjoint eigenfunctions and biorthogonal expansions—to properly characterize their spectral properties and dynamical behavior.

For non-self-adjoint operators, we must consider both the original problem and its adjoint. The corresponding adjoint eigenvalue problem is given by

$$\mathcal{L}^{+} = \mathbf{D}\frac{d^2}{d\xi^2} - c\frac{d}{d\xi} + \mathbf{J}(\hat{\mathbf{u}}), \tag{2.10}$$

with eigenfunctions denoted by $\mathbf{W}(\xi)$, satisfying

$$\mathcal{L}^{+}\mathbf{W} = \lambda\mathbf{W}. \tag{2.11}$$

The eigenfunctions $\mathbf{V}_j(\xi)$ and $\mathbf{W}_j(\xi)$ of $\mathcal{L}$ and $\mathcal{L}^{+}$, respectively, satisfy the biorthogonality condition

$$\left\langle \mathbf{W}_j, \mathbf{V}_k \right\rangle = \int_{-\infty}^{\infty} \mathbf{W}_j^{\top}(\xi)\mathbf{V}_k(\xi)\, d\xi = \begin{cases} 1, & \text{if } j = k, \\ 0, & \text{otherwise.} \end{cases} \tag{2.12}$$

For the sake of clarity and consistency with prior analytical frameworks [13,14], we assume that the eigenvalues λ_j of the operator $\mathcal{L}$ are all real and simple. Furthermore, we order the eigenpairs $(\lambda_j, \mathbf{V}_j)$ such that

$$\lambda_1 > \lambda_2 = 0 > \lambda_3 > \cdots,$$

where λ_1 corresponds to the single unstable mode and $\lambda_2 = 0$ arises due to translational invariance of the critical solution. The corresponding eigenfunctions $\{\mathbf{V}_j\}$ are assumed to form a basis in the appropriate functional space, and the same assumption holds for the adjoint eigenfunctions $\{\mathbf{W}_j\}$ of $\mathcal{L}^{+}$.

In self-adjoint problems, the eigenfunctions are orthogonal, which makes the analysis much easier. In non-self-adjoint systems, we lose that orthogonality, but the eigenfunctions and their adjoints are still biorthogonal. This means we can still project the dynamics onto each mode and better understand how different parts of the system behave. In many cases, the leading eigenpair—associated with the positive eigenvalue λ_1—has the biggest impact on how the system behaves in the vicinity of the critical solution. However, understanding the full evolution of perturbations often requires more than just this dominant mode. To really understand how perturbations evolve over time, especially in multicomponent systems or when transient effects matter, we also need to consider subleading eigenpairs. These additional modes may have a noticeable influence, even if they decay more slowly or act over shorter time scales.

Computing these left and right eigenpairs accurately can be challenging. In practice, obtaining multiple eigenpairs can be computationally expensive, especially for high-dimensional problems or when high accuracy is required. In light of these challenges, several recent studies have proposed alternative strategies to approximate the eigenvalues and eigenfunctions of such systems with improved robustness and efficiency. A recent study [15] introduced neural network approaches called power method neural network (PMNN) and inverse power method neural network (IPMNN) to solve linear eigenvalue problems. These proposed methods combine principles from traditional numerical analysis with modern deep learning techniques. The PMNN approach mimics the conventional power method through a neural network framework, iteratively refining the approximation of the dominant eigenpair. Similarly, IPMNN implements an inverse power scheme to capture the smallest eigenvalue and its corresponding eigenfunction. While the method performs well in computing isolated eigenpairs, it is designed to recover either the dominant (largest) or the smallest eigenvalue and its corresponding eigenfunction, one at a time. Another study [16] introduced the generalized inverse power method neural network and its physics-constrained variant to solve generalized eigenvalue problems with discontinuities, focusing on computing the smallest eigenvalue and its corresponding eigenfunction, and showed that adding interface conditions during training improves both the accuracy and the stability of the solution.

Another recent study [17] explores a different direction by applying supervised machine learning models to inverse eigenvalue problems. Instead of computing eigenfunctions, the authors aim to reconstruct unknown system parameters—such as the potential in a Sturm–Liouville problem or the refractive index in a transmission eigenvalue problem—using a small number of known eigenvalues as input. While the approach shows promising results in recovering these parameters, it does not address the computation of eigenfunctions or multiple eigenpairs directly and is limited to the linear, one-dimensional case. Physics-informed neural networks have also been applied to eigenvalue problems in quantum systems governed by the Schrödinger equation, nonlinear Helmholtz equations in acoustics, and quantum billiards [18–20]. A related study employs PINNs to compute multiple eigenstates of quantum systems by solving the Schrödinger equation, using customized loss functions to enforce normalization, orthogonality, and symmetry constraints during training [21].

Although the studies mentioned above offer valuable tools for analyzing eigenvalue problems with neural networks, most of them come with certain limitations. Many are designed to find only a single eigenpair—either the largest or the smallest—and computing additional modes often requires repeating the training process. In some cases, eigenfunctions are not computed at all, or their orthogonality is not strictly enforced. Moreover, the majority of these studies focus on self-adjoint problems, and do not fully account for the complexities introduced by non-self-adjoint operators or systems with multiple components.

In the context of our problem, which involves analyzing reaction-diffusion systems around critical solutions, it's not enough to compute just one eigenpair. A complete understanding of the behavior near these solutions—particularly in multicomponent cases—requires access to both left and right eigenfunctions, as well as their associated eigenvalues. These eigenpairs offer a useful representation of the system's behavior and are central to understanding its response to perturbations. Our goal, therefore, is to develop a physics-informed neural network (PINN)–based method capable of accurately obtaining multiple eigenpairs simultaneously. To achieve this, the method incorporates both the structure of the spectrum and the appropriate orthogonality or biorthogonality constraints directly into the training process. This makes it particularly suitable for handling more complex cases, including multicomponent systems and non-self-adjoint operators, where traditional approaches often fall short.

3. Physics-informed neural networks (PINNs)

A neural network is a computational framework designed to approximate complex functions by mimicking the structure of neural connections in the human brain [22]. It functions as a universal approximator, meaning it can represent any continuous function given sufficient depth (number of hidden layers) and width (number of neurons). A typical architecture includes an input layer, one or more hidden layers, and an output layer. The input layer receives data, the hidden layers apply nonlinear transformations using activation functions, and the output layer produces the final output. The learning process involves adjusting weights and biases, which control how inputs are combined and thresholds are applied. This flexibility allows neural networks to capture complex patterns in data and adapt to a wide variety of tasks. Collectively, this process defines a function

$$\mathcal{N}^{(L)}(\mathbf{x}) : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}, \tag{3.1}$$

where $\mathcal{N}^{(L)}(\mathbf{x})$ denotes the output of a neural network with L layers applied to an input $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$. The mathematical structure of each of these layers can be written as follows:

- **Input layer:**
$$\mathcal{N}^{(0)}(\mathbf{x}) = \mathbf{x}, \quad \mathbf{x} \in \mathbb{R}^{d_{\text{in}}}. \tag{3.2}$$
- **Hidden layers:**
$$\mathcal{N}^{(l)} = \sigma\left(\mathbf{W}_l \mathcal{N}^{(l-1)} + \mathbf{b}_l\right), \quad \mathbf{W}_l \in \mathbb{R}^{N_l \times N_{l-1}}, \mathbf{b}_l \in \mathbb{R}^{N_l}, \ 1 \le l \le L-1, \tag{3.3}$$
where σ is the activation function applied element-wise.
- **Output layer:**
$$\mathcal{N}^{(L)} = \mathbf{W}_L \mathcal{N}^{(L-1)} + \mathbf{b}_L, \quad \mathbf{W}_L \in \mathbb{R}^{d_{\text{out}} \times N_{L-1}}, \mathbf{b}_L \in \mathbb{R}^{d_{\text{out}}}. \tag{3.4}$$

Here, $\mathbf{W}_l$ denotes the weight matrix associated with layer l, where each entry $W_{ji}^{(l)}$ represents the weight connecting the i-th neuron in layer $l-1$ to the j-th neuron in layer l. Similarly, $\mathbf{b}_l$ is the bias vector for layer l, and the function σ refers to the nonlinear activation function applied element-wise to the output of the affine transformation. The parameters N_l and N_{l-1} indicate the number of neurons in the l-th and $(l-1)$-th layers, respectively. The final layer index L determines the total depth of the network, with d_{in} and d_{out} representing the input and output dimensions of the network, respectively.

The approximate solution to the target function or differential equation is represented by the neural network output. This solution is parameterized by $\theta = \{\mathbf{W}_l, \mathbf{b}_l\}_{l=1}^{L}$ and mathematically the solution can be expressed as

$$\hat{U}(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \theta) = \left(\mathcal{N}^{(L)} \circ \mathcal{N}^{(L-1)} \circ \cdots \circ \mathcal{N}^{(0)}\right)(\mathbf{x}), \tag{3.5}$$

where $\circ$ represents function composition. The parameters θ of the network are aimed to be optimized by minimizing a loss function, which measures the discrepancy between the network's output and the desired target—such as the residual of a differential equation, boundary conditions, or known solution values. Through this optimization process, the neural network learns to approximate the target solution U, effectively capturing the behavior dictated by the underlying equations and constraints.

In the framework of PINNs, the loss function plays a central role by measuring how well the neural network's output aligns with the expected physical behavior of the system. It is typically formulated as a combination of terms that penalize deviations from the governing differential equation, as well as any prescribed boundary or initial conditions. This structure guides the training process, ensuring that the learned solution remains consistent with the underlying mathematical model and captures both the dynamics and mathematical or physical constraints associated with the problem.

The training process involves minimizing the loss function using gradient-based optimization algorithms, such as the Adam optimizer [23]. By leveraging automatic differentiation, PINNs can compute exact derivatives of the neural network output with respect to both inputs and parameters, eliminating the need for numerical differentiation or mesh-based discretization [24]. This enables greater flexibility in selecting training points and supports efficient implementation over complex or unstructured domains [25].

The optimization process in the PINN framework is formulated as

$$\theta^* = \arg\min_{\theta} \mathscr{L}(\theta),$$

where θ^* denotes the optimal set of network parameters that minimize the total loss $\mathscr{L}$. During training, the weights and biases of the network are iteratively updated to reduce the residuals of the PDE and satisfy the given boundary or initial conditions. Through this procedure, the network progressively learns to approximate the solution with increasing accuracy. After training, the predictions can be validated against analytical results or benchmark numerical results. If the accuracy does not meet expectations, improvements can be made by adjusting the network architecture, refining the loss function, or selecting alternative optimization strategies.

In some cases, such as our eigenvalue problem, certain parameters that appear in the governing equations—like the eigenvalue itself—are not known beforehand. In these situations, the framework can be extended using inverse physics-informed neural networks (inverse PINNs), which treat such unknowns as additional trainable parameters. This allows the network to learn both the solution and the underlying parameter simultaneously by minimizing a loss function that incorporates the differential equation and available constraints. This approach is particularly effective when the parameter of interest is not directly observable but indirectly governs the system's behavior, a scenario frequently encountered in spectral problems [26, 27].

4. PINN-based methodology

In this section, we introduce a PINN framework designed to solve the class of eigenvalue problems considered in this paper, including both self-adjoint and non-self-adjoint cases, as will be demonstrated in the two test problems presented in the next section. The eigenvalue problems for the operator $\mathcal{L}$ and its adjoint $\mathcal{L}^+$ are defined as

$$\mathcal{L}\mathbf{V} = \lambda\mathbf{V}, \quad \mathcal{L}^+\mathbf{W} = \lambda\mathbf{W}, \quad \text{in } \Omega, \tag{4.1}$$

where $\mathcal{L}$ and $\mathcal{L}^+$ are the linearized and adjoint linearized operators, respectively, $\mathbf{V}(\xi)$ and $\mathbf{W}(\xi)$ are the eigenfunctions, and λ is the shared eigenvalue. The shared boundary conditions for the eigenvalue problems of $\mathcal{L}$ and $\mathcal{L}^+$ are given as

$$\mathscr{B}\mathbf{V} = \mathscr{B}\mathbf{W} = g, \quad \text{on } \partial\Omega, \tag{4.2}$$

where $\mathcal{B}$ is the boundary operator, and g defines the required boundary behavior. The domain $\Omega \subset \mathbb{R}$ denotes the spatial interval over which the eigenvalue problem is defined, and $\partial\Omega$ refers to the boundary of this domain, where the boundary conditions are applied.

We denote the neural network by $\mathcal{N}(\mathbf{x}; \lambda, \theta)$, where θ represents the network parameters, and λ is treated as an additional unknown parameter to be learned during training. In this framework, we use the same neural network to approximate both the left and right eigenfunctions. These two networks are trained and optimized simultaneously, with their outputs used in the same loss function. Consequently, we denote the approximated eigenfunctions by $\hat{\mathbf{V}}(\mathbf{x})$ and $\hat{\mathbf{W}}(\mathbf{x})$, while the predicted eigenvalue $\hat{\lambda}$ is learned simultaneously during training as part of the network's output.

Since the eigenvalue λ is treated as a trainable parameter alongside the neural network weights and biases, special attention must be given to the formulation of the loss function. An effective loss must ensure that the network not only satisfies the linearized and adjoint linearized eigenvalue problems but also enforces essential properties such as normalization, biorthonormality between the left and right eigenfunctions, and consistency in the eigenvalue estimate. These components work together to guide the optimization process and ensure that the learned eigenpair accurately reflects the spectral behavior of the operators.

The primary components of the loss function are derived from the eigenvalue problems for the equations in (4.1). These terms ensure that the neural network approximations $\hat{\mathbf{V}}(\mathbf{x})$ and $\hat{\mathbf{W}}(\mathbf{x})$ satisfy their respective eigenvalue problems. The combined loss is defined as

$$\mathscr{L}_{\text{ODE}} = \frac{1}{N_f} \sum_{\mathbf{x} \in \mathcal{D}_f} \left(\left\| \mathcal{L}\hat{\mathbf{V}}(\mathbf{x}) - \hat{\lambda}\hat{\mathbf{V}}(\mathbf{x}) \right\|^2 + \left\| \mathcal{L}^{+}\hat{\mathbf{W}}(\mathbf{x}) - \hat{\lambda}\hat{\mathbf{W}}(\mathbf{x}) \right\|^2 \right), \tag{4.3}$$

where $|| \cdot ||$ denotes the L^2-norm, defined as

$$||f||^2 = \int_{\Omega} |f(x)|^2 \, dx$$

for a continuous function $f(x)$, or equivalently, as the sum of squared values over discrete points for numerical implementations. In this context, the L^2-norm quantifies the discrepancy between the predicted eigenfunctions $\hat{\mathbf{V}}(\mathbf{x})$ and $\hat{\mathbf{W}}(\mathbf{x})$ and their respective eigenvalue problems. Additionally, $\mathcal{D}_f$ refers to N_f collocation points sampled inside the domain Ω. These collocation points are used to evaluate the residuals of the eigenvalue problems for both the approximated left eigenfunction $\hat{\mathbf{V}}(\mathbf{x})$ and right eigenfunction $\hat{\mathbf{W}}(\mathbf{x})$.

To ensure the stability of the neural network's approximation and to prevent the solutions from collapsing into trivial or null solutions, we need to enforce the biorthonormality conditions required for the eigenvalue problem. To guarantee that the right eigenfunctions $\hat{\mathbf{V}}_k^i$ and left eigenfunctions $\hat{\mathbf{W}}_k^i$ satisfy biorthonormality, we define a loss function that enforces the following conditions:

- **Biorthonormality:** $\langle \hat{\mathbf{V}}_k^i, \hat{\mathbf{W}}_k^i \rangle = 1$, ensuring that each right eigenfunction $\hat{\mathbf{V}}_k^i$ and its corresponding left (adjoint) eigenfunction $\hat{\mathbf{W}}_k^i$ are normalized with respect to one another.
- **Biorthogonality:** $\langle \hat{\mathbf{V}}_k^i, \hat{\mathbf{W}}_m^i \rangle = 0$ for $k \neq m$, enforcing mutual orthogonality between non-matching pairs of right and left eigenfunctions.

The biorthonormality loss function, ensuring normalization of eigenfunction pairs and orthogonality of non-paired combinations, is defined as

$$\mathscr{L}_{\mathrm{BO}} = \sum_{k=1}^{n} \left(\langle \hat{\mathbf{V}}_k^i, \hat{\mathbf{W}}_k^i \rangle - 1 \right)^2 + \sum_{k \neq m} \left(\langle \hat{\mathbf{V}}_k^i, \hat{\mathbf{W}}_m^i \rangle \right)^2 . \tag{4.4}$$

This part of our loss function helps the model find solutions that do two important things at once: follow the physics we are modeling (the governing equations) and maintain the special mathematical relationship we need between our functions (biorthonormality). Without this term, we would risk getting solutions that are either physically meaningless or numerically unstable. Hence, minimizing above term helps prevent degenerate or ill-conditioned solutions and enhances numerical stability in the learned representations.

To encourage the predicted eigenvalues $\hat{\lambda}_k$ to follow the expected spectral structure $\hat{\lambda}_1 > \hat{\lambda}_2 = 0 > \hat{\lambda}_3 > \ldots$, we introduce an eigenvalue ordering loss defined as

$$\mathscr{L}_{\mathrm{EIG}} = \sum_{k=1}^{n-1} \max(0, \hat{\lambda}_{k+1} - \hat{\lambda}_k)^2 + (\hat{\lambda}_2)^2. \tag{4.5}$$

The first term penalizes any violations of the strict decreasing order of the eigenvalues by contributing a nonzero value only when $\hat{\lambda}_{k+1} > \hat{\lambda}_k$, while the second term enforces that the second eigenvalue tends to zero, as expected from the translational invariance of the underlying system. This part of the loss makes sure the predicted eigenvalues stay physically realistic and match what theory expects, helping avoid overlapping modes and keeping the overall spectral structure intact during training.

In a one-dimensional domain, the boundary $\partial\Omega$ consists of just two points—the left and right ends of the interval. Accordingly, the boundary dataset $\mathcal{D}_{\mathrm{BC}}$ includes $N_{\mathrm{BC}} = 2$ collocation points, which are used to enforce the boundary conditions for both the right and left eigenfunctions. The corresponding boundary loss is defined using the L^2-norm evaluated at these endpoints as

$$\mathscr{L}_{\mathrm{BC}} = \frac{1}{2} \sum_{\mathbf{x} \in \mathcal{D}_{\mathrm{BC}}} \left(\|\mathcal{B}\hat{\mathbf{V}}(\mathbf{x}) - g(\mathbf{x})\|^2 + \|\mathcal{B}\hat{\mathbf{W}}(\mathbf{x}) - g(\mathbf{x})\|^2 \right), \tag{4.6}$$

where $\mathcal{B}$ denotes the boundary operator and $g(\mathbf{x})$ is the prescribed boundary condition. This soft constraint approach allows the boundary conditions to be enforced approximately by adding a penalty term to the loss function. It is simple to implement and offers flexibility, especially for more complex or irregular domains.

To further improve accuracy and training stability, we also consider a hard constraint formulation that embeds the boundary conditions directly into the architecture of the neural network. In this approach, the network's output is transformed to inherently satisfy the boundary constraints by construction, following the method originally proposed in [28]. Specifically, we define the transformed outputs as

$$\tilde{\mathbf{V}}(\xi) = \mathbf{V}_c(\xi) + \psi(\xi)\hat{\mathbf{V}}(\xi), \quad \tilde{\mathbf{W}}(\xi) = \mathbf{W}_c(\xi) + \psi(\xi)\hat{\mathbf{W}}(\xi), \tag{4.7}$$

where $\mathbf{V}_c(\xi)$ and $\mathbf{W}_c(\xi)$ are functions that satisfy the boundary conditions at the domain boundaries, i.e., $\mathcal{B}[\mathbf{V}_c] = \mathcal{G}(\xi)$ and $\mathcal{B}[\mathbf{W}_c] = \mathcal{G}(\xi)$ for $\xi \in \partial\Omega$, and $\mathcal{G}(\xi)$ denotes the prescribed boundary data. The function $\psi(\xi)$ is a boundary basis function that vanishes on $\partial\Omega$, ensuring that $\tilde{\mathbf{V}}(\xi)$ and $\tilde{\mathbf{W}}(\xi)$ inherently satisfy the boundary constraints. By using this transformation, the predicted solutions are guaranteed to respect the boundary conditions, leading to physically consistent and accurate outputs throughout the domain.

The use of hard constraints offers several advantages: it eliminates the need for explicit boundary penalty terms, simplifies the loss function, improves numerical stability, and enforces physical consistency. Although harder to implement for complex or irregular geometries, this approach is well suited to the one-dimensional domain considered in this study. By embedding the boundary conditions directly into the neural network output, we enhance the convergence properties of the model and reduce computational cost while preserving accuracy, as also noted in previous works such as [25].

To improve the consistency between the predicted eigenfunctions and their associated eigenvalue, we incorporate an additional loss term based on the original time-dependent reaction-diffusion equation. In addition to solving the eigenvalue problems, we would also like to ensure that the predicted eigenvalue and eigenfunction pairs respect the dynamics of the original time-dependent PDE. Specifically, we define the transformed variables

$$\hat{\mathbf{v}}(\xi,\tau) = e^{\hat{\lambda}\tau}\hat{\mathbf{V}}(\xi), \quad \hat{\mathbf{w}}(\xi,\tau) = e^{\hat{\lambda}\tau}\hat{\mathbf{W}}(\xi), \tag{4.8}$$

and substitute them into the original linearized and adjoint linearized PDEs. This allows us to construct a residual loss that directly penalizes discrepancies in the full spatiotemporal dynamics. The resulting PDE-based residual loss is given by

$$\mathscr{L}_{\text{PDE}} = \frac{1}{N_t}\sum_{j=1}^{N_t}\left(\left\|\frac{\partial\hat{\mathbf{v}}}{\partial\tau_j} - \mathbf{D}\frac{\partial^2\hat{\mathbf{v}}}{\partial\xi^2} - c\frac{\partial\hat{\mathbf{v}}}{\partial\xi} - \mathbf{J}(\hat{\mathbf{u}})\hat{\mathbf{v}}\right\|^2 + \left\|\frac{\partial\hat{\mathbf{w}}}{\partial\tau_j} - \mathbf{D}\frac{\partial^2\hat{\mathbf{w}}}{\partial\xi^2} + c\frac{\partial\hat{\mathbf{w}}}{\partial\xi} - \mathbf{J}^{\top}(\hat{\mathbf{u}})\hat{\mathbf{w}}\right\|^2\right), \tag{4.9}$$

where $\hat{\mathbf{v}}(\xi,\tau_j)$ and $\hat{\mathbf{w}}(\xi,\tau_j)$ are evaluated at discrete time values $\{\tau_j\}_{j=1}^{N_t}$ sampled over a finite interval $[\tau_{\min},\tau_{\max}]$. This loss encourages the learned eigenvalue–eigenfunction pairs to remain consistent not only with the eigenvalue problem itself but also with the broader temporal dynamics from which the spectral problem was derived.

The total loss function is formulated as a weighted sum of individual components, each enforcing a specific constraint of the eigenvalue problem. These include residuals from the eigenvalue problems, boundary conditions, biorthonormality constraints, eigenvalue ordering, and the consistency of the learned eigenpairs with the original time-dependent equations. The total loss is then defined as

$$\mathscr{L}_{\text{total}} = \omega_{\text{ODE}}\,\mathscr{L}_{\text{ODE}} + \omega_{\text{PDE}}\,\mathscr{L}_{\text{PDE}} + \omega_{\text{BC}}\,\mathscr{L}_{\text{BC}} + \omega_{\text{BO}}\,\mathscr{L}_{\text{BO}} + \omega_{\text{EIG}}\,\mathscr{L}_{\text{EIG}}, \tag{4.10}$$

where $\omega_{\text{ODE}}, \omega_{\text{PDE}}, \omega_{\text{BC}}, \omega_{\text{BO}}, \omega_{\text{EIG}}$ are tunable weights that control the relative influence of each term during training. It is important to note that this formulation assumes the use of soft constraints for enforcing boundary conditions, where $\mathscr{L}_{\text{BC}}$ is explicitly minimized to ensure compliance. However, if we use hard constraints with output transformation, the boundary conditions are inherently satisfied by the structure of the network output. In such cases, the boundary loss term $\mathscr{L}_{\text{BC}}$ is automatically zero and can be excluded from the total loss calculation.

Minimizing the total loss $\mathscr{L}_{\text{total}}$ guides the neural network to satisfy multiple requirements at once: the differential and adjoint eigenvalue equations, boundary conditions, biorthonormality constraints, expected eigenvalue structure, and consistency with the full time-dependent PDE. The eigenvalue $\hat{\lambda}$ and the associated eigenfunctions $\hat{\mathbf{V}}(\mathbf{x})$ and $\hat{\mathbf{W}}(\mathbf{x})$ are learned through this optimization process. Automatic differentiation is used to compute spatial and temporal derivatives, enabling the enforcement of physical and mathematical constraints without relying on numerical discretization.

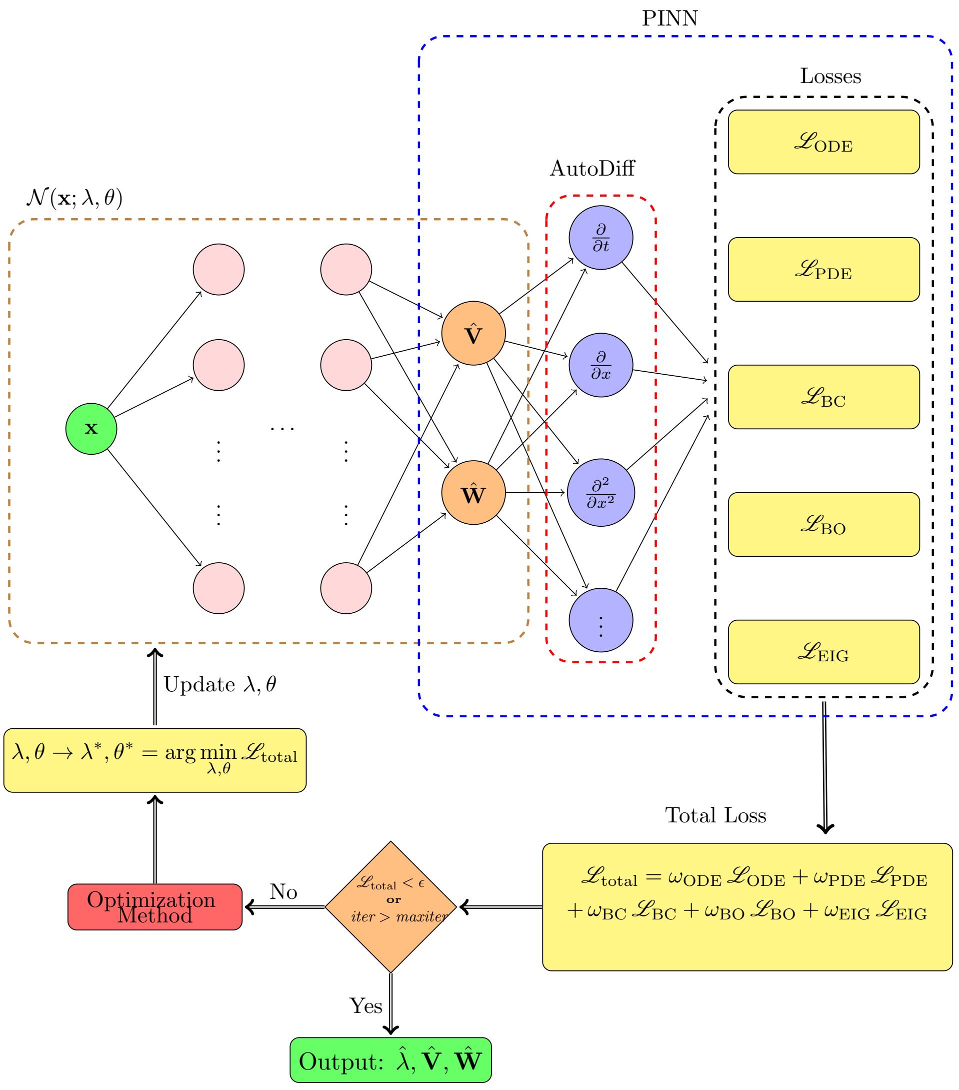
PINN
Losses
$\mathscr{L}_{\text{ODE}}$
$\mathscr{L}_{\text{PDE}}$
$\mathscr{L}_{\text{BC}}$
$\mathscr{L}_{\text{BO}}$
$\mathscr{L}_{\text{EIG}}$
AutoDiff
$\frac{\partial}{\partial t}$
$\frac{\partial}{\partial x}$
$\frac{\partial^2}{\partial x^2}$
$\mathcal{N}(\mathbf{x};\lambda,\theta)$
$\mathbf{x}$
$\hat{\mathbf{V}}$
$\hat{\mathbf{W}}$
Update λ,θ
$\lambda,\theta \to \lambda^*,\theta^* = \arg\min_{\lambda,\theta} \mathscr{L}_{\text{total}}$
Total Loss
$\mathscr{L}_{\text{total}} = \omega_{\text{ODE}}\mathscr{L}_{\text{ODE}} + \omega_{\text{PDE}}\mathscr{L}_{\text{PDE}} + \omega_{\text{BC}}\mathscr{L}_{\text{BC}} + \omega_{\text{BO}}\mathscr{L}_{\text{BO}} + \omega_{\text{EIG}}\mathscr{L}_{\text{EIG}}$
Optimization Method
No
$\mathscr{L}_{\text{total}} < \epsilon$ or $iter > maxiter$
Yes
Output: $\hat{\lambda}, \hat{\mathbf{V}}, \hat{\mathbf{W}}$

Figure 1. Schematic representation of a PINN framework tailored for solving eigenvalue problems.

Figure 1 illustrates the overall structure of the PINN framework designed to solve eigenvalue problems. The process begins with the spatial input $\mathbf{x}$, which is passed through a neural network parameterized by θ and augmented with the eigenvalue λ as a trainable variable. The network generates predictions for both the right and left eigenfunctions, denoted $\hat{\mathbf{V}}$ and $\hat{\mathbf{W}}$, respectively. Using automatic differentiation, spatial and temporal derivatives of the predicted eigenfunctions are computed to form the following residuals: $\mathscr{L}_{\text{ODE}}$ for the eigenvalue problems, $\mathscr{L}_{\text{PDE}}$ for consistency with the time-dependent problems, $\mathscr{L}_{\text{BO}}$ to enforce biorthonormality, and $\mathscr{L}_{\text{EIG}}$ to maintain the spectral ordering. When soft constraints are employed, an additional term $\mathscr{L}_{\text{BC}}$ enforces the boundary conditions.

All components are combined into the total loss $\mathscr{L}_{\text{total}}$, which is minimized using a gradient-based optimization algorithm. The optimization proceeds iteratively, updating both the network parameters and the eigenvalue until convergence criteria are met. These criteria include either the reduction of the total loss below a specified threshold or the completion of a maximum number of training iterations. Upon convergence, the trained network outputs the approximated eigenvalues and left and rigth eigenfunctions $(\hat{\lambda}, \hat{\mathbf{V}}, \hat{\mathbf{W}})$, which satisfy the spectral properties and physical constraints of the underlying problem.

5. Results

We apply the proposed PINN-based eigenvalue solver to two distinct test problems. The first test case is the Zeldovich-Frank-Kamenetsky (ZFK) equation, a one-dimensional self-adjoint problem, and the second test case is the FitzHugh–Nagumo (FHN) system, a two-component non-self-adjoint problem. We use the same neural network architecture for both problems, despite their differing mathematical structures. We use a neural network with five hidden layers, each containing 80 neurons. We use the `swish` activation function in all hidden layers for its smooth and non-monotonic behavior, which enhances convergence, especially in stiff or highly nonlinear regimes [29]. The network parameters are initialized using the Glorot uniform scheme [30].

Training is conducted using a staged optimization procedure. In the initial phases, the Adam optimizer is applied with a learning rate schedule that progressively decreases across training stages. The learning rate starts at 8×10^{-3} for the first stage and is reduced to 1×10^{-4} in the second stage. Each stage is run for a total of 20000 iterations. We apply %10 dropout after every hidden layer to prevent overfitting and encourage the network to learn more robust features. This simple regularization not only improves generalization when training data are limited or noisy, but also introduces slight variability into the network outputs, providing a practical estimate of solution uncertainty—a strategy shown to be effective in both forward and inverse PINN frameworks [31]. In our case, we use this variability to get a sense of how stable the predicted quantities (such as eigenvalues) are under small perturbations to the network. Running the model multiple times with dropout active yields a distribution of outputs, which offers insight into the reliability of the results. The optimization process is also stopped early using an EarlyStopping callback to ensure the model converges quickly.

In the later stages of training, the limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) optimizer with a total of 30000 iterations is employed for finer optimization. This quasi-Newton algorithm is known for its effectiveness in minimizing physics-informed loss functions, particularly in landscapes that are stiff or constrained [32]. Similar to the Adam stage, the %10 dropout and EarlyStopping callback is used to halt training when further progress becomes negligible, ensuring efficient convergence. This two-step process helps balance stability and accuracy: the first optimizer gets the model to converge quickly, and then the L-BFGS optimizer fine-tunes it for better precision and alignment with the physics.

Uniformly spaced collocation points are distributed throughout the domain, excluding the boundary regions. In both cases we consider below, we employ no-flux boundary conditions, and these boundary conditions are enforced via hard constraints through output transformations, which ensure compliance with the prescribed boundary behavior by construction, without the need for additional penalty terms. Furthermore, the output transformation is designed to impose additional structural properties such as symmetry or anti-symmetry, depending on the spectral characteristics of the specific eigenfunctions being learned. Enforcing the correct parity focuses the model on the relevant solution subspace, which sharpens its search, avoids unphysical artifacts, and leads to faster, more stable convergence and higher-quality eigenfunction estimates [21,33].

To enforce the required even- or odd-symmetry, we apply a simple transform to the network's raw output that mirrors the prediction appropriately, ensuring that every computed mode inherently follows the chosen symmetry before any loss is evaluated. For an even eigenfunction $V(x)$ (same idea is valid for $W(x)$), we set

$$\hat{V}(x) \;=\; \mathcal{N}(|x|), \tag{5.1}$$

which by construction yields

$$\hat{V}(-x) \;=\; \mathcal{N}(|-x|) = \mathcal{N}(|x|) = \hat{V}(x).$$

For an odd eigenfunction, we instead use

$$\hat{V}(x) \;=\; \operatorname{sgn}(x)\,\mathcal{N}(|x|), \tag{5.2}$$

so that

$$\hat{V}(-x) = \operatorname{sgn}(-x)\,\mathcal{N}(|-x|) = -\,\operatorname{sgn}(x)\,\mathcal{N}(|x|) = -\,\hat{V}(x).$$

In practice, ensuring convergence to the correct eigenvalue is a critical aspect of training PINN-based eigenvalue solvers. Without proper initialization or guidance, PINNs can converge to undesired eigenvalues—often higher excited states—even when the initial guesses for the eigenvalue or eigenfunction are reasonable. To mitigate this risk in our implementation, we adopt a structured initialization approach that improves stability and alignment with theoretical expectations.

Specifically, we initialize the first eigenfunction using a randomly generated function that satisfies the prescribed boundary conditions and is normalized. For the second eigenfunction, we use the derivative of the first eigenfunction as the initial guess. This not only provides a nontrivial starting point but also promotes orthogonality between the first and second modes. This process is iteratively extended to higher modes, where each new eigenfunction is initialized in a way that ensures orthogonality to the previously computed ones.

Moreover, we leverage theoretical insights about the spectrum of the system: for example, in the non-self-adjoint case, we know that the second eigenvalue should lie close to zero due to translational symmetry. Based on this, we target the second eigenpair first and progressively compute the remaining ones, enforcing orthogonality constraints to guide the network toward the correct spectral components. This targeted strategy helps avoid convergence to spurious eigenmodes and improves the overall robustness of the method.

We aim to compare our PINN-based approach for estimating multiple eigenpairs with direct numerical simulations obtained via the modified power iteration scheme and Gram–Schmidt orthogonalization [34]. We begin by choosing any linearly independent initial guesses $\{V_k^{(0)}, W_k^{(0)}\}_{k=1}^{n}$, setting $\lambda_k^{(0)} = 0$ for all k, and then proceed with the following updates at each iteration i for $k = 1, \ldots, n$:

$$\widetilde{V}_k^{(i)} \;=\; \exp(\mathcal{L}T)\,V_k^{(i-1)}, \qquad \widetilde{W}_k^{(i)} \;=\; \exp(\mathcal{L}^{+}T)\,W_k^{(i-1)}. \tag{5.3}$$

To enforce biorthogonality against the previously computed modes $m < k$, we subtract their projections:

$$\widetilde{V}_k^{(i)} \leftarrow \widetilde{V}_k^{(i)} - \sum_{m=1}^{k-1} \langle W_m^{(i)}, \widetilde{V}_k^{(i)} \rangle V_m^{(i)},$$

$$\widetilde{W}_k^{(i)} \leftarrow \widetilde{W}_k^{(i)} - \sum_{m=1}^{k-1} \langle \widetilde{W}_k^{(i)}, V_m^{(i)} \rangle W_m^{(i)}. \tag{5.4}$$

We then normalize each mode to unit length:

$$V_k^{(i)} = \frac{\widetilde{V}_k^{(i)}}{\|\widetilde{V}_k^{(i)}\|}, \qquad W_k^{(i)} = \frac{\widetilde{W}_k^{(i)}}{\|\widetilde{W}_k^{(i)}\|}. \tag{5.5}$$

Finally, the eigenvalue is updated via the logarithmic stretch:

$$\lambda_k^{(i)} = \frac{1}{T} \ln \langle W_k^{(i)}, \widetilde{V}_k^{(i)} \rangle. \tag{5.6}$$

We consider the method converged once the approximated eigenvalues stabilize within a chosen tolerance. A common criterion involves monitoring the spectral shifts—changes in successive eigenvalue estimates—and ensuring they fall below a specified threshold:

$$\max_{1 \le k \le n} \left| \lambda_k^{(i)} - \lambda_k^{(i-1)} \right| < \varepsilon,$$

for some tolerance ε. However, one can also monitor the change in the eigenfunctions themselves via their L^2-norms, e.g.,

$$\max_{1 \le k \le n} \left\| V_k^{(i)} - V_k^{(i-1)} \right\|_2 < \varepsilon \quad \text{and} \quad \max_{1 \le k \le n} \left\| W_k^{(i)} - W_k^{(i-1)} \right\|_2 < \varepsilon.$$

For even greater robustness, one may combine both eigenvalue and eigenfunction criteria into a single stopping condition. While this hybrid approach can guard against premature termination, it incurs additional cost from computing and comparing multiple norms at each iteration. Once the selected convergence criterion is met, the algorithm terminates and

$$\{(\lambda_k^{(i)}, V_k^{(i)}, W_k^{(i)})\}_{k=1}^{n} \tag{5.7}$$

are accepted as the final eigenpairs.

In the self-adjoint case the left and right eigenfunctions coincide, so we only need a single sequence of vectors $\{w_k^{(i)}\}$. Once each mode has been propagated via the operator exponential, we enforce orthogonality against all previously accepted modes via a modified Gram–Schmidt step. In practice, for each k we subtract from $w_k^{(i)}$ its projections onto all $m < k$ modes:

$$w_k^{(i)} \leftarrow w_k^{(i)} - \sum_{m=1}^{k-1} \langle w_k^{(i)} \mid w_m^{(i)} \rangle w_m^{(i)}. \tag{5.8}$$

This guarantees $\langle w_k^{(i)}, w_m^{(i)} \rangle = 0$ for $m < k$. We then rescale each mode to unit length,

$$w_k^{(i)} \leftarrow \frac{w_k^{(i)}}{\|w_k^{(i)}\|}, \tag{5.9}$$

so that $\langle w_k^{(i)}, w_k^{(i)} \rangle = 1$. This single orthonormalization step replaces the biorthogonal projections and separate normalization used in the non-self-adjoint algorithm while preserving all the essential spectral properties.

5.1. Case I: Self-adjoint one-component problem

Here, we consider a one-dimensional reaction-diffusion system governed by a self-adjoint linear operator, $\mathcal{L} = \mathcal{L}^+$. In this case, the left and right eigenfunctions are identical, which simplifies the problem. Instead of predicting separate solutions for each, we only need to predict one set of eigenfunctions, $\hat{V}$, along with the corresponding eigenvalues, $\hat{\lambda}$. In this case, the original biorthonormality condition is no longer required. Instead, the eigenfunctions must satisfy standard orthonormality when appropriately normalized. This effectively reduces the biorthonormality constraint to a simpler orthonormality condition.

The selected test problem is Zeldovich-Frank-Kamenetsky (ZFK) equation [35], which has been extensively used to model wave propagation phenomena in various engineering applications. This equation, also known as the Schlögl model [36] in chemical kinetics or the Nagumo equation [37] in neuroscience, provides a benchmark problem for studying reaction-diffusion systems. Mathematically, the ZFK equation is defined in the following form

$$u_t = u_{xx} + u(u-\theta)(1-u), \tag{5.10}$$

where $\theta \in (0, 1/2)$ is a threshold parameter. In the following simulation, the parameter θ is fixed at 0.15. The critical nucleus $\hat{u}(x)$ associated with this problem is known in closed form and given by [13,34]

$$\hat{u}(x) = \frac{3\theta\sqrt{2}}{(1+\theta)\sqrt{2} + \cosh(x\sqrt{\theta})\sqrt{2 - 5\theta + 2\theta^2}}. \tag{5.11}$$

We now consider the eigenvalue problem obtained by linearizing the system around $\hat{u}$. Since the linearized operator $\mathcal{L}$ is self-adjoint and $c = 0$, the eigenvalue problem reduces to finding $\lambda \in \mathbb{R}$ and eigenfunction $V(x)$ pairs satisfying

$$\frac{d^2V}{d\xi^2} + \left(-3\hat{u}^2 + 2(\theta+1)\hat{u} - \theta\right)V = \lambda V, \quad V(\pm\infty) = 0. \tag{5.12}$$

In fact, one can show by direct differentiation that the spatial derivative of the critical nucleus,

$$V_2(x) = \frac{d\hat{u}}{dx}(x),$$

satisfies the homogeneous problem $\mathcal{L}V_2 = 0$ and therefore corresponds to the zero eigenvalue $\lambda_2 = 0$. This property reflects the translational invariance of the solution and provides a built-in verification of our numerical scheme. Furthermore, when the threshold parameter θ is small, the cubic nonlinearity

$u(u-\theta)(1-u)$ may be approximated by the simpler quadratic form $u(u-\theta)$, allowing one to derive closed-form expressions for the leading spectral quantities. In this limit the critical nucleus and the dominant eigenpair reduces to written explicitly as [38]

$$\hat{u}(x) \approx \frac{3}{2}\,\theta\,\mathrm{sech}^2\!\Big(\tfrac{x\sqrt{\theta}}{2}\Big), \quad \lambda_1 \approx \frac{5\,\theta}{4}, \quad V_1(x) \approx \mathrm{sech}^3\!\Big(\tfrac{x\sqrt{\theta}}{2}\Big). \tag{5.13}$$

These analytical approximations provide valuable benchmarks for our PINN-based solver, ensuring that the learned eigenvalues and eigenfunctions converge to the correct asymptotic forms in the regime $\theta \ll 1$.

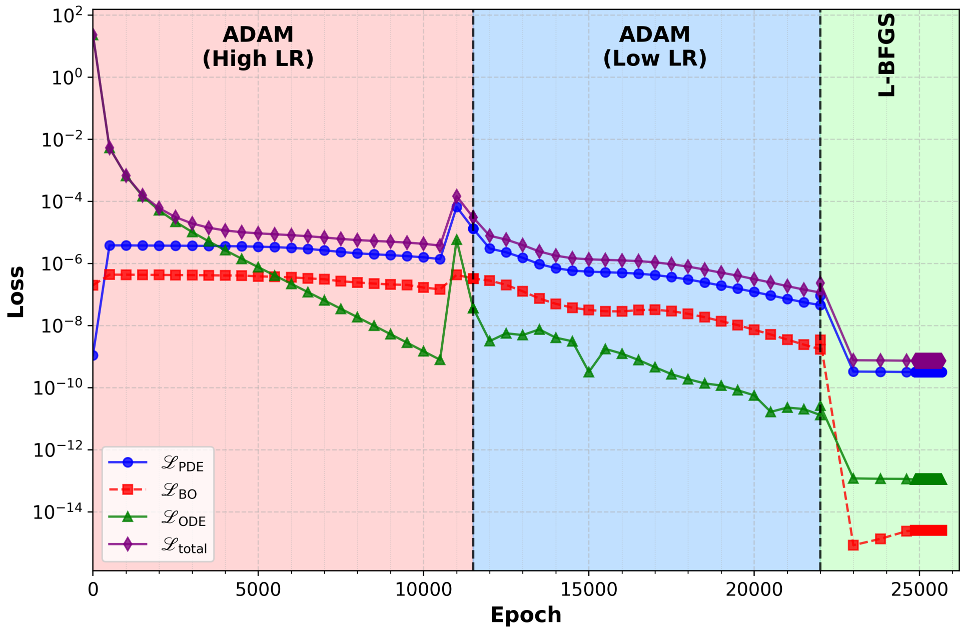

Figure 2. Illustration of the loss computation for the ZFK equation.

Since boundary conditions are imposed through hard constraints, the associated loss term is satisfied exactly and therefore omitted from the visualization. The eigenvalue loss is also excluded, as it is constructed to align with the spectral structure and converges quickly during training, making its contribution negligible beyond the initial iterations. Figure 2 focuses instead on the remaining loss components, shown across three distinct optimization phases: two stages using the Adam optimizer (with the learning rates specified earlier), followed by a final stage using L-BFGS. The plot is color-coded to indicate where stopping criteria are met for each regime. As the figure shows, the overall loss decreases steadily and stabilizes after approximately 25000 iterations, marking the point at which the solution converges and the end of training.

Notably, the ODE-related loss $\mathscr{L}_{\mathrm{ODE}}$ dominates the early convergence phase, especially during high learning rate training. As optimization progresses, $\mathscr{L}_{\mathrm{PDE}}$ and $\mathscr{L}_{\mathrm{BO}}$ become more influential. In the final L-BFGS phase, both $\mathscr{L}_{\mathrm{BO}}$ and $\mathscr{L}_{\mathrm{ODE}}$ contribute less to the total loss, while $\mathscr{L}_{\mathrm{PDE}}$ increasingly governs the optimization dynamics. These trends underscore the complementary and evolving contributions of each term in guiding the optimization toward convergence.

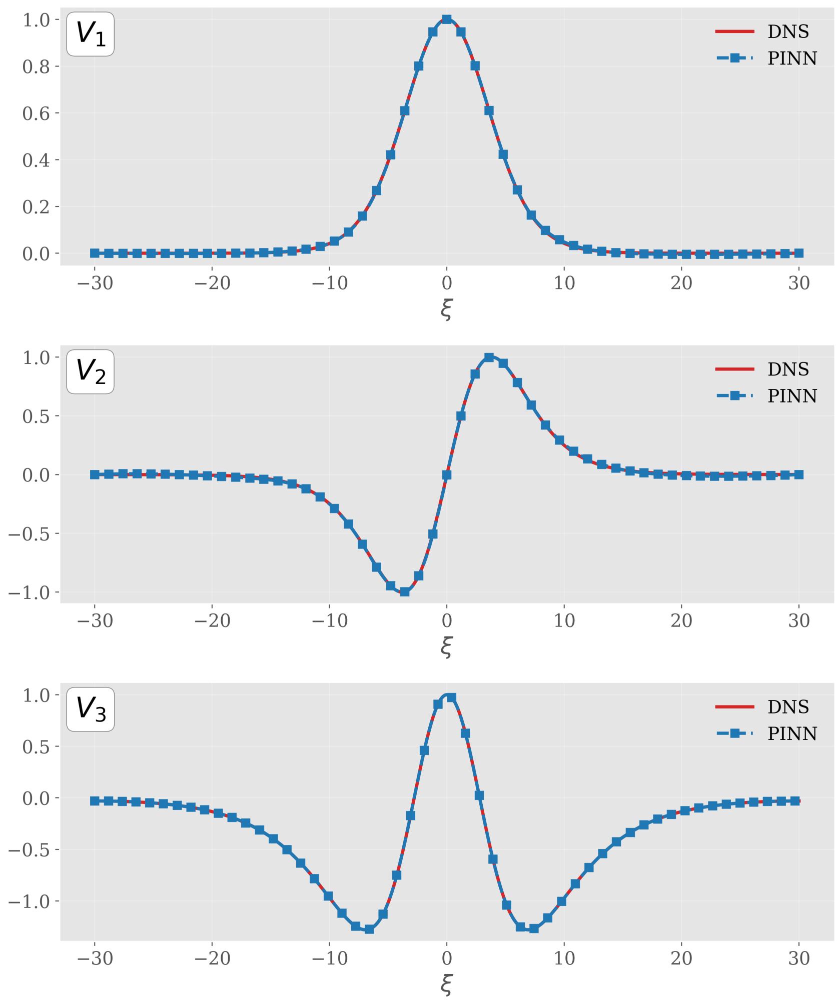

Figure 3. PINN-based approximation of the first three eigenfunctions of the ZFK equation compared with those obtained with DNS.

To numerically solve the eigenvalue problem in (5.12) using our PINN framework, we construct a loss function that penalizes violations of the differential equation and the orthonormality constraints. Specifically, the differential operator $\mathcal{L} = \frac{d^2}{d\xi^2} + \left(-3\hat{u}^2 + 2(\theta + 1)\hat{u} - \theta\right)$ is applied to the neural network output $V(x)$, and the residual $\mathcal{L}V - \lambda V$ is evaluated at a set of collocation points in the domain. The corresponding loss term integrates the squared residuals over these points, effectively enforcing the eigenvalue equation. Since the system is self-adjoint, we impose standard orthonormality conditions between the eigenfunctions via an additional loss term based on inner products. These combined constraints guide the network toward learning eigenpairs that satisfy both the governing PDE and the spectral structure of the solution space.

We compute the first three eigenfunctions of the ZFK equation using our PINN framework and compare them to direct numerical simulations (DNS), obtained via the modified power iteration scheme with Gram–Schmidt orthogonalization described above. To ensure a meaningful comparison, both methods are applied to the same spatial domain with the same discretization in the spatial coordinate. As shown in Figure 3, the eigenfunctions produced by the PINN closely match those obtained from the direct numerical method, demonstrating strong agreement between the two approaches.

Since the second eigenfunction of the ZFK problem coincides exactly with the spatial derivative of the critical nucleus, it provides a valuable benchmark against a known closed-form expression. To quantify the accuracy of our PINN and DNS approximations, we introduce the relative L^2 error

$$E_{\mathrm{rel}}(V_2) = \frac{\|V_2 - V_2^{\mathrm{exact}}\|_2}{\|V_2^{\mathrm{exact}}\|_2} \tag{5.14}$$

where $V_2^{\mathrm{exact}}(x) = \frac{d\hat{u}}{dx}(x)$ is the analytic translational mode. By computing E_{rel} for both the PINN prediction and the DNS result against this exact profile, we obtain a direct measure of how accurately each method resolves the zero mode. As shown in Table 1, the PINN-based reconstruction of the translational mode achieves a relative L^2 error of 0.0035, compared with 0.0042 for the direct numerical simulation. These similarly low error levels indicate that the PINN accurately captures the fine spatial structure of the derivative eigenfunction while enforcing the boundary conditions effectively. Such close agreement builds confidence in applying the PINN framework to compute higher-order eigenmodes for which no analytic solutions are available.

Table 1. Relative L^2 error for the second eigenfunction V_2.

Method	**Relative $E_{\mathrm{rel}}(V_2)$ error**
PINN-based approximation	0.0035
Direct numerical simulation	0.0042

Table 2 presents a comparison of the first three eigenvalues computed by direct numerical simulation (DNS), our PINN framework, and the available analytical estimate for the dominant mode. Both DNS and PINN accurately recover the neutral second eigenvalue at $\lambda_2 \approx 0$. The leading eigenvalue λ_1 shows a modest deviation from its asymptotic analytical value $\lambda_1 \approx 5\theta/4 = 0.1875$, reflecting that $\theta = 0.15$ is only marginally small; despite this, DNS and PINN agree closely at approximately 0.1424. For the second eigenvalue, the analytical value is exactly zero for both quadratic and cubic kinetics and for all θ, making it a critical benchmark for comparison. Our PINN estimate of λ_2 matches this expectation almost perfectly, and the DNS result likewise lies within numerical noise of zero, confirming excellent agreement across analytical, numerical, and PINN approaches. The third eigenvalue lacks a simple closed-form expression, yet the PINN prediction remains within a few percent of the DNS result. These findings demonstrate that, even beyond the strict asymptotic limit, our PINN approach reliably reproduces the self-adjoint ZFK spectrum with high fidelity.

Table 2. Comparison of the first three eigenvalue approximations for the ZFK model using DNS, PINN, and analytical estimates.

Eig. Val.	**DNS**	**PINN**	**Analytical**
λ_1	0.14237	0.14242	0.1875
λ_2	0.00009	0.00007	0
λ_3	−0.10881	−0.11680	*unknown*

5.2. *Case II: Non-self-adjoint two-component problem*

Our second test problem is the FitzHugh–Nagumo (FHN) system, which extends the ZFK model by introducing a slow recovery variable. This modification leads to a two-component reaction–diffusion system commonly used to model excitable media, such as neuronal dynamics and cardiac tissue [37, 39, 40]. In contrast to the previous case, we now have the non-self-adjoint case, meaning $\mathcal{L} \neq \mathcal{L}^{+}$, and therefore both left and right eigenpairs must be computed to accurately characterize the system's spectral properties. We consider the equation in the following form

$$\begin{aligned} u_t &= u_{xx} + u(u-\beta)(1-u) - v, \\ v_t &= \gamma(\alpha u - v), \end{aligned} \tag{5.15}$$

where $u(x,t)$ and $v(x,t)$ denote the fast and slow variables, respectively. The parameters are fixed as $\beta = 0.05$, $\alpha = 0.37$, and $\gamma = 0.01$, which govern the excitation threshold and the timescale separation between the fast and slow dynamics.

An explicit analytical form of the critical pulse solution $(\hat{u}, \hat{v})$ is not available for this system. For a detailed numerical derivation, as well as the computation of the left and right eigenfunctions and their eigenvalues, the reader is referred to [34]. We linearize the system around this stationary critical pulse solution. The right eigenvalue problem is in the following form

$$\lambda \mathbf{V} = \mathcal{L}\mathbf{V}, \tag{5.16}$$

where

$$\mathcal{L} = \begin{pmatrix} \partial_\xi^2 - c\partial_\xi - 3\hat{u}^2 + 2(1+\beta)\hat{u} - \beta & -1 \\ \gamma\alpha & -c\partial_\xi - \gamma \end{pmatrix}, \quad \mathbf{V} = \begin{pmatrix} \mathcal{U} \\ \mathcal{V} \end{pmatrix}. \tag{5.17}$$

and the corresponding adjoint (left) eigenvalue problem

$$\lambda \mathbf{W} = \mathcal{L}^{+}\mathbf{W}, \tag{5.18}$$

where

$$\mathcal{L}^{+} = \begin{pmatrix} \partial_\xi^2 + c\partial_\xi - 3\hat{u}^2 + 2(1+\beta)\hat{u} - \beta & \gamma\alpha \\ -1 & c\partial_\xi - \gamma \end{pmatrix}, \quad W = \begin{pmatrix} \mathcal{U}^{*} \\ \mathcal{V}^{*} \end{pmatrix}. \tag{5.19}$$

In this non-self-adjoint setting, left and right eigenfunctions are not identical and must be computed separately. Consequently, biorthonormality conditions must be imposed during training to enforce consistency between these eigenfunction pairs. This structure introduces additional complexity to the learning process.

Since no closed-form expressions are available for the stationary pulse or its spectrum in the FHN system, we validate our PINN-computed eigenpairs by direct comparison with those obtained via classical numerical simulation. As with the ZFK case, we omit the boundary condition and eigenvalue loss components from the visualization, since the boundary conditions are enforced via hard constraints and the eigenvalue loss converges rapidly. Figure 4 presents the remaining loss terms during training for the FHN model, split across three optimization phases: two stages using the Adam optimizer with previously specified learning rates, followed by a final stage using the L-BFGS optimizer. The colored regions mark transitions between phases based on the stopping criteria. Training for the FHN model takes slightly longer, with convergence reached after approximately

30000 iterations. This increase is expected, given the two-component structure of the system and the additional requirement to learn both left and right eigenfunctions. Nonetheless, we observe convergence of the learned eigenpair, at which point training is terminated.

As seen from the figure for the FHN equation, the loss components exhibit distinct contributions across training phases. During the initial ADAM phase with a high learning rate, both $\mathscr{L}_{\text{ODE}}$ and $\mathscr{L}_{\text{BO}}$ dominate the total loss. As training progresses into the second ADAM phase with a reduced learning rate, $\mathscr{L}_{\text{PDE}}$ becomes increasingly significant, contributing to solution refinement. From the end of the second ADAM phase through the entire L-BFGS stage, $\mathscr{L}_{\text{PDE}}$ becomes the dominant term. These transitions highlight the complementary contributions of each component across different stages of optimization.

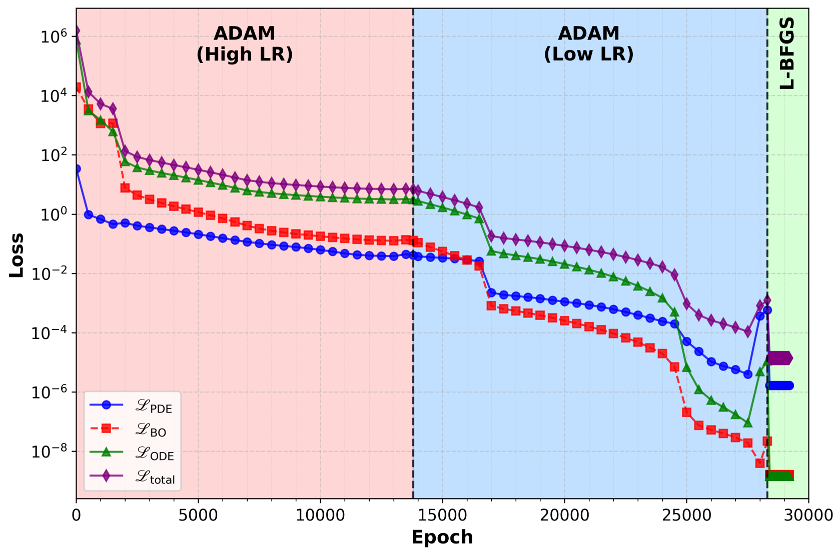

Figure 4. Illustration of the loss computation for the FHN equation.

For the FHN system, we compute the first two left and right eigenpairs and compare the outcomes of the PINN-based method with those from direct numerical simulation. As shown in Figure 5, the results exhibit strong alignment, confirming the reliability of the PINN framework in this more complex, non-self-adjoint case.

Finally, Table 3 presents the eigenvalues obtained using both the PINN-based method and direct numerical simulation for the FHN model. The values show close agreement across the two approaches, further demonstrating the accuracy of the PINN framework in capturing the spectral properties of this non-self-adjoint system.

The eigenvalues in Tables 2 and 3 are listed according to their theoretical significance within the system, rather than by strict numerical magnitude. Specifically, λ_1 corresponds to the dominant eigenvalue that characterizes the system's primary spectral behavior, while λ_2 arises due to translational symmetry and is theoretically expected to be zero. Subsequent eigenvalues represent additional spectral modes. This ordering reflects the expected spectral structure based on analytical and numerical studies of the associated linearized operator. The eigenfunctions shown in the corresponding figures follow the same labeling: each is plotted in relation to the eigenvalue it belongs to, making it easier to interpret their qualitative features and link them to the underlying dynamics.

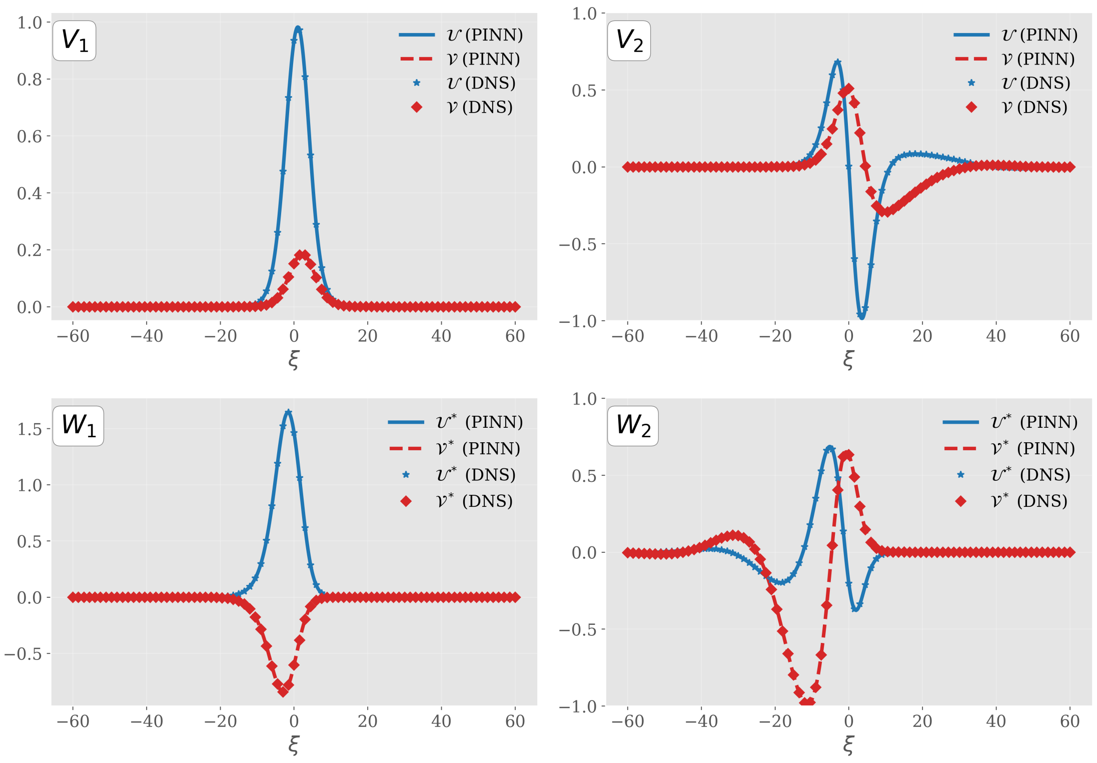

Figure 5. PINN-based approximation of the first two left and right eigenfunctions of the FHN equation, compared against results obtained from direct numerical simulation.

Table 3. Comparison of the first two eigenvalue approximations for the FHN model using DNS and PINN.

Eig. Val.	**DNS**	**PINN**
λ_1	0.17228	0.17245
λ_2	−0.00007	0.00013

To complement the accuracy evaluations above, we also assess the computational efficiency of the proposed PINN-based framework. Specifically, we compare its runtime performance against the numerical procedure previously described for computing eigenpairs. In terms of computational cost, the modified power iteration scheme based on finite difference discretization required approximately 60.29 seconds to compute three eigenpairs for the self-adjoint ZFK system, while the PINN-based method took around 22 times longer to deliver comparable results. For the non-self-adjoint FHN system, computing the first two left and right eigenpairs using this scheme took 752.81 seconds, whereas the corresponding PINN approach resulted in a roughly 153-times increase in runtime.

Although these findings highlight the higher computational demand of neural network-based solvers, their flexibility in integrating physical structure, solving inverse problems, and learning from sparse or noisy data makes them a compelling alternative to traditional numerical methods. With ongoing progress in neural network architectures and training algorithms, their efficiency and scalability are expected to improve significantly in the near future.

6. Conclusion

In this work, we developed a physics-informed neural network (PINN) framework to solve eigenvalue problems that arise in reaction–diffusion systems, focusing on both self-adjoint and non-self-adjoint cases. Our method incorporates the governing physics directly into the network's structure and training process, enabling it to approximate multiple eigenpairs—both eigenvalues and their corresponding left and right eigenfunctions—at once. Unlike conventional numerical methods, it achieves this without the need for predefined grids or mesh-based solvers.

The key innovation of this work lies in how spectral properties are embedded directly into the learning process. By enforcing biorthonormality between left and right eigenfunctions and introducing a spectral separation strategy, the method is able to distinguish leading eigenmodes and improve training stability—particularly in non-self-adjoint settings where traditional approaches often struggle. These additions make the framework not only more robust but also more interpretable. This appears to be the first use of such structural constraints within a PINN-based approach for spectral problems.

We applied our method to two well-known reaction–diffusion systems—one single-component (ZFK) and one multi-component (FHN)—to demonstrate its effectiveness across different types of models. For the ZFK equation, which involves a self-adjoint operator and a single variable, we computed the first three leading eigenvalues along with their corresponding eigenfunctions. We also validated the PINN's second eigenfunction against its known closed-form expression, obtaining very small relative L^2 error. Finally, we compared these results with those from direct numerical simulation and found excellent agreement across all modes, confirming the high fidelity of our PINN approach. The FHN system, being a two-component model with a non-self-adjoint operator, posed a more challenging problem that required computing both left and right eigenfunctions. Even though the FHN system was more complex, the PINN method still worked well. By adding a few extra constraints to keep the left and right eigenfunctions aligned and the eigenvalues in the right order, we were able to get accurate and consistent results. These two examples highlight the flexibility of our approach: it can handle both simpler one-component systems where standard spectral properties apply and more complex multi-component systems where additional structure needs to be learned. This suggests that the method has the potential to be generalized to a broad class of reaction–diffusion systems, regardless of whether they involve self-adjoint or non-self-adjoint operators or single- or multi-component dynamics.

One of the key parts of our work is how we built the loss function. We did not just focus on the eigenvalue problem—we also included terms that come from the original time-dependent system. This helped the model learn spectral modes that not only solve the math correctly but also make sense in terms of how the system actually behaves over time.

While this study focuses on one-dimensional reaction–diffusion systems, the proposed framework can, in principle, be extended to higher-dimensional problems and systems with additional components. However, such extensions come with increased computational demands. As the number of spatial dimensions or solution components grows, the number of collocation points and trainable parameters must also increase to maintain accuracy. This, in turn, raises memory requirements and training times. Similarly, computing more eigenpairs requires enlarging the neural network's output layer and introducing additional orthogonality constraints, which may slow convergence. Addressing these scalability issues—especially in two- or three-dimensional domains—would benefit from strategies such as adaptive sampling, parallel training, or more efficient loss formulations, and is an important direction for future work.

Since the eigenvalue problem is formulated as an inverse PINN task, another promising direction would be to explore the use of transfer learning. One possible approach is to first train the model to learn the network parameters for a known eigenpair. Once trained, these parameters could be fixed and reused to identify additional eigenpairs. This strategy has the potential to reduce computational time significantly, especially when solving sequences of related spectral problems.

In this work, we focused on problems where the considered eigenvalues and their corresponding eigenfunctions are real. This assumption simplifies both the theoretical formulation and the training procedure, allowing us to validate the core framework in settings where accurate comparisons with analytical or numerical benchmarks are feasible. However, many non-self-adjoint operators encountered in physical and biological systems can give rise to complex eigenvalues. Extending the current method to handle such cases would require reformulating the framework to support complex-valued outputs and associated loss functions. This presents a technically meaningful direction for future research, and exploring it could significantly expand the range of systems to which our PINN-based approach can be applied.

Use of Generative-AI tools declaration

The author declares Artificial Intelligence (AI) tools were not used in the creation of this article.

Data availability

The datasets analysed during the current study are not publicly available but are available from the corresponding author on reasonable request.

Conflict of interest

The author has no conflict of interest to declare.

References

1. M. Ghergu, V. Radulescu, *Nonlinear PDEs: Mathematical models in biology, chemistry and population genetics*, Berlin, Heidelberg: Springer, 2011.
2. V. Volpert, S. Petrovskii, Reaction–diffusion waves in biology, *Phys. Life Rev.*, **6** (2009), 267–310. http://doi.org/10.1016/j.plrev.2009.10.002
3. A. C. Newell, J. A. Whitehead, Finite bandwidth, finite amplitude convection, *J. Fluid Mech.*, **38** (1969), 279–303. http://doi.org/10.1017/S0022112069000176
4. A. M. Turing, The chemical basis of morphogenesis, *Bltn. Mathcal. Biology*, **52** (1990), 153–197. http://doi.org/10.1007/BF02459572

5. B. Bezekci, I. Idris, R. D. Simitev, V. N. Biktashev, Semianalytical approach to criteria for ignition of excitation waves, *Phys. Rev. E*, **92** (2015), 042917. http://doi.org/10.1103/PhysRevE.92.042917

6. A. L. Andrew, J. W. Paine, Correction of finite element estimates for Sturm-Liouville eigenvalues, *Numer. Math.*, **50** (1986), 205–215. https://doi.org/10.1007/BF01390430

7. I. E. Lagaris, A. Likas, D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE Trans. Neural Netw.*, **9** (1998), 987–1000. http://doi.org/10.1109/72.712178

8. M. Raissi, P. Perdikaris, G. E. 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. http://doi.org/10.1016/j.jcp.2018.10.045

9. X. Li, J. Zhang, Z. Wang, Y. Liu, Fourier-feature induced physics-informed randomized neural network method for solving the biharmonic equation, *J. Comput. Appl. Math.*, **468** (2025), 116635. http://doi.org/10.1016/j.cam.2024.116635

10. X. Li, J. Zhang, Z. Wang, A hybrid PINN framework with soft and hard constraints using Fourier expansions for advection-diffusion equations, *Comput. Math. Appl.*, **159** (2024), 60–75.

11. B. Bezekci, V. N. Biktashev, Strength-duration relationship in an excitable medium, *Commun. Nonlinear Sci. Numer. Simul.*, **80** (2020), 104954. http://doi.org/10.1016/j.cnsns.2019.104954

12. B. Bezekci, Deep Learning-Enhanced Regularization of Irregular Traveling Pulses in the FitzHugh-Nagumo Model, *SN Comput. Sci.*, **6** (2025), 206. https://doi.org/10.1007/s42979-025-03752-5

13. G. Flores, The stable manifold of the standing wave of the Nagumo equation, *J. Differential Equations*, **80** (1989), 306–314. https://doi.org/10.1016/0022-0396(89)90086-7

14. G. Flores, Stability analysis for the slow travelling pulse of the FitzHugh–Nagumo system, *SIAM J. Math. Anal.*, **22** (1991), 392–399. https://doi.org/10.1137/0522025

15. Q. Yang, Y. Deng, Y. Yang, Q. He, S. Zhang, Neural networks based on power method and inverse power method for solving linear eigenvalue problems, *Comput. Math. Appl.*, **147** (2023), 14–24. https://doi.org/10.1016/j.camwa.2023.07.013

16. Q.-H. Yang, Y. Yang, Y.-T. Deng, Q.-L. He, H.-L. Gong, S.-Q. Zhang, Physics-constrained neural network for solving discontinuous interface K-eigenvalue problem with application to reactor physics, *Nucl. Sci. Tech.*, **34** (2023), 161. https://doi.org/10.1007/s41365-023-01313-0

17. N. Pallikarakis, A. Ntargaras, Application of machine learning regression models to inverse eigenvalue problems, *Comput. Math. Appl.*, **154** (2024), 162–174. https://doi.org/10.1016/j.camwa.2023.11.0384

18. M. Mattheakis, G. R. Schleder, D. T. Larson, E. Kaxiras, First principles physics-informed neural network for quantum wavefunctions and eigenvalue surfaces, 2022. Available from: `https://arxiv.org/abs/2211.04607`.

19. E. G. Holliday, J. F. Lindner, W. L. Ditto, Solving quantum billiard eigenvalue problems with physics-informed machine learning, *AIP Adv.*, **13** (2023), 085315. https://doi.org/10.1063/5.0161067

20. L. Harcombe, Q. Deng, Physics-informed neural networks for discovering localised eigenstates in disordered media, *J. Comput. Sci.*, **73** (2023), 102136. http://doi.org/10.1016/j.jocs.2023.102136

21. H. Jin, M. Mattheakis, P. Protopapas, Physics-informed neural networks for quantum eigenvalue problems, *2022 International Joint Conference on Neural Networks (IJCNN)*, 2022, 1–8. http://doi.org/10.1109/IJCNN55064.2022.9892705

22. A. D. Dongare, R. R. Kharde, A. D. Kachare, Introduction to artificial neural network, *Int. J. Eng. Innov. Technol. (IJEIT)*, **2** (2012), 189–194.

23. D. Kingma, J. L. Ba. Adam, Adam: A method for stochastic optimization, *International Conference on Learning Representations (ICLR) 2015*, 2015.

24. A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: A survey, *J. Mach. Learn. Res.*, **18** (2018), 1–43.

25. L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, S. G. Johnson, Physics-informed neural networks with hard constraints for inverse design, *SIAM J. Sci. Comput.*, **43** (2021), B1105–B1132. http://doi.org/10.1137/21M1397908

26. S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs, *IMA J. Numer. Anal.*, **42** (2022), 981–1022. https://doi.org/10.1093/imanum/drab032

27. L. Yuan, Y.-Q. Ni, X.-Y. Deng, S. Hao, A-PINN: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations, *J. Comput. Phys.*, **462** (2022), 111260. http://doi.org/10.1016/j.jcp.2022.111260

28. L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, *SIAM Review*, **63** (2021), 208–228. http://doi.org/10.1137/19M1274067

29. P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, 2017). Available from: `https://arxiv.org/abs/1710.05941`.

30. X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, 249–256.

31. D. Zhang, L. Lu, L. Guo, G. E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, *J. Comput. Phys.*, **397** (2019), 108850. http://doi.org/10.1016/j.jcp.2019.07.048

32. D. C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, *Math. Program.*, **45** (1989), 503–528. http://doi.org/10.1007/BF01589116

33. M. Mattheakis, P. Protopapas, D. Sondak, M. Di Giovanni, E. Kaxiras, Physical symmetries embedded in neural networks, 2019, Available from: `https://arxiv.org/abs/1904.08991`.

34. B. Bezekci, *Analytical and numerical approaches to initiation of excitation waves*, University of Exeter, PhD thesis, 2016.

35. Ya. B. Zel'dovich, D. A. Frank-Kamenetsky, Towards the theory of uniformly propagating flames, *Doklady AN SSSR*, **19** (1938), 693–697.

36. F. Schlögl, Chemical reaction models for non-equilibrium phase transitions, *Z. Phys.*, **253** (1972), 147–161. http://doi.org/10.1007/BF01379769

37. J. Nagumo, S. Arimoto, S. Yoshizawa, An active pulse transmission line simulating nerve axon, *Proc. IRE*, **50** (1962), 2061–2070. http://doi.org/10.1109/JRPROC.1962.288235

38. I. Idris, V. N. Biktashev, Analytical approach to initiation of propagating fronts, *Phys. Rev. Lett.*, **101** (2008), 244101. http://doi.org/10.1103/PhysRevLett.101.244101

39. R. FitzHugh, Impulses and physiological states in theoretical models of nerve membrane, *Biophys. J.*, **1** (1961), 445–466. http://doi.org/10.1016/S0006-3495(61)86902-6

40. J. C. Neu, R. S. Preissig Jr., W. Krassowska, Initiation of propagation in a one-dimensional excitable medium, *Physica D*, **102** (1997), 285–299. https://doi.org/10.1016/S0167-2789(96)00203-5