Mathematical Biosciences and Engineering

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

MBE, 22(7): 1598–1633.
DOI: 10.3934/mbe.2025059
Received: 27 February 2025
Revised: 21 April 2025
Accepted: 09 May 2025
Published: 20 May 2025

Research article

Physics-informed neural networks for optimal vaccination plan in SIR epidemic models

Minseok Kim[1,†], **Yeongjong Kim**[2,†] **and Yeoneung Kim**[1,*]

[1] Department of Applied Artificial Intelligence, SeoulTech, Nowon-gu 01811, Republic of Korea

[2] Department of Mathematics, POSTECH, Nam-gu 06974, Republic of Korea

† These authors contributed equally to the work.

* **Correspondence:** Email: yeoneung@seoultech.ac.kr; Tel: +8229709715.

Abstract: This work investigates the minimum eradication time in a controlled susceptible-infectious-recovered model with constant infection and recovery rates. The eradication time is defined as the earliest time the infectious population falls below a prescribed threshold and remains below it. Leveraging the fact that this problem reduces to solving a Hamilton-Jacobi-Bellman (HJB) equation, we propose a mesh-free framework based on a physics-informed neural network to approximate the solution. Moreover, leveraging the well-known structure of the optimal control of the problem, we efficiently obtain the optimal vaccination control from the minimum eradication time using the dynamic programming principle. To improve training stability and accuracy, we incorporate a variable scaling method and provide theoretical justification through a neural tangent kernel analysis. Numerical experiments show that this technique significantly enhances convergence, reducing the mean squared residual error by approximately 80% compared with standard physics-informed approaches. Furthermore, the method accurately identifies the optimal switching time. These results demonstrate the effectiveness of the proposed deep learning framework as a computational tool for solving optimal control problems in epidemic modeling as well as the corresponding HJB equations.

Keywords: physics-informed neural networks; optimal control; Hamilton-Jacobi-Bellman equation; controlled epidemic model; minimum eradication time

1. Introduction

The study of vaccination strategies and eradication times in susceptible-infectious-recovered (SIR) models has a long history, beginning with the seminal work of Kermack and McKendrick [1], and

its variants have received a great deal of attention during and after the outbreak of COVID-19. The controlled SIR model is a cornerstone in mathematical epidemiology and is frequently employed to study the dynamics of disease transmission and control strategies. Various optimization problems based on the SIR model, where the vaccination strategy is treated as a control, have been extensively studied within the framework of optimal control theory [2–5].

The controlled SIR model is given by

$$\begin{cases} \dot{S} = -\beta(t)SI - r(t)S, \\ \dot{I} = \beta(t)SI - \gamma(t)I, \end{cases}$$

for $t > 0$, with the initial conditions $S(0) = x$ and $I(0) = y \geq \mu$ for some $\mu > 0$. Here, $\beta(t)$ and $\gamma(t)$ denote the infection and recovery rates, respectively, and $r(t)$ represents a vaccination control that takes values in $[0, 1]$.

Recently, Bolzoni et al. [4] introduced the notion of minimum eradication time, defined as the first time I falls below a given threshold $\mu > 0$. For mathematical treatments, the eradication time in controlled SIR models with constant infection and recovery rates was first studied as a viscosity solution to a static first-order Hamilton-Jacobi-Bellman (HJB) equation in [6]. Additionally, two critical times in SIR dynamics were studied in [7]: the point at which the infected population begins to decrease and the first time that this population falls below a given threshold. Both studies [6, 7] focused on SIR models with a constant β and γ. To identify the optimal controls, the Pontryagin maximum principle (PMP) [8] was applied, confirming that a bang-bang control (i.e., taking values of 0 or 1) is optimal. The authors of [9] extended the notion of eradication time to cases involving time-inhomogeneous dynamics. However, numerical treatments for solving Hamilton–Jacobi equations and determining the optimal vaccination controls remain relatively unexplored.

Motivated by the fact that the minimum eradication time satisfies an HJB equation in the viscosity sense [6, 10], we introduce a novel framework based on physics-informed neural networks (PINNs) to solve this problem in a mesh-free and scalable manner. Traditional numerical schemes often suffer from discretization errors and are limited in their flexibility, especially when handling high-dimensional or nonlinear systems. In contrast, our approach embeds the HJB's dynamics directly into the training of a neural network, enabling efficient approximation without spatial discretization. Moreover, by incorporating the dynamic programming principle (DPP) into a neural optimization loop, we compute both the value function and the associated optimal bang-bang vaccination control within a unified PINN framework. This integration offers a new perspective for solving optimal control problems in the study of controlled epidemic models using deep learning.

Unlike existing deep learning approaches for epidemic modeling, which typically require retraining for each initial condition, our PINN-based framework computes the minimum eradication time uniformly across a range of initial states. This is made possible by the structural property of the optimal control in our setting. Specifically, the optimal control exhibits a bang-bang form, which is a unique property in our problem's setting. Leveraging this structure, we apply the DPP to recover the optimal vaccination strategy directly from the computed eradication time, without solving the control problem separately.

In addition, we provide rigorous theoretical support to explain why it works. While PINNs have shown success in various applications, their training is often unstable, particularly when applied to stiff or degenerate partial differential equations (PDEs). To address this issue, we leverage a variable

scaling (VS) technique originally proposed in [11] and develop a theoretical foundation for solving the HJB equation arising in the minimum eradication time problem. Specifically, using neural tangent kernel (NTK) theory [12], we quantify how VS affects the convergence rate during the training of PINNs. This constitutes the first NTK-based convergence analysis associated with HJB equations arising in the controlled SIR problem and provides new insights into the theoretical tractability of PINN-based optimal control. We further analyze the impact of applying distinct scaling factors to each variable. This variable-specific scaling effectively captures the anisotropic behavior inherent in the HJB equation.

1.1. Related works

PINNs [13] have gained significant attention as a powerful and flexible framework for solving differential equations, and have been widely adopted in various fields, including epidemic modeling [14, 15], fluid mechanics [13, 16–18], finance [19, 20], and biomedical engineering [21, 22], where understanding the underlying physical models is crucial.

In the framework of PINNs, we solve differential equations by training a neural network. The loss function consists of initial and boundary conditions, along with residual terms derived from the governing equations. However, the training results are highly sensitive to the choice of boundary condition settings, requiring the introduction of a penalty coefficient to balance the boundary loss term. Heuristic adjustments to the penalty coefficient can accelerate convergence. However, if not properly set, these values may lead to inaccurate solutions. To address these challenges, various adaptive methods have been proposed. One example is a learning rate annealing algorithm [23]. This algorithm dynamically adjusts the weights assigned to each term in the loss function. PINNs with adaptive weighted loss functions have been introduced for the efficient training of Hamilton-Jacobi (HJ) equations [24]. To further enhance the stability of PINNs, the authors of [25] proposed an adaptive training strategy that ensures stable convergence through the lens of NTK theory [12]. Recently, the failure of PINNs in stiff ordinary differential equation (ODE) systems was observed [26], and stiff-PINN was proposed as an improvement. Subsequently, various methods have been introduced, such as self-adaptive PINNs [27] and variable scaling PINNs [11]. Among these methods, we employ the variable scaling technique [11], as it is simple and effective.

While PINNs have been successfully applied to a wide range of differential equation problems, their application to optimal control, particularly in solving HJB equations, remains relatively underexplored. The key challenge lies in ensuring stability and accuracy when approximating the value functions and control policies. This has motivated recent studies investigating the interplay between deep learning and optimal control, aiming to develop computationally efficient methods that leverage the advantages of PINNs for solving partial differential equation (PDE) and optimal control problems.

There is a rich body of literature exploring the interplay between PINNs and optimal control. By leveraging the ability of PINNs to solve partial differential equations (PDEs) and the scalability of deep neural networks, researchers have developed computationally efficient methods for solving optimal control problems. For instance, a training procedure for obtaining optimal control in PDE-constrained problems was presented in [28]. Similarly, the authors of [29] utilized a Lyapunov-type PDE for efficient policy iteration in control-affine problems. Slightly later, a deep operator learning framework was introduced to solve high-dimensional optimal control problems [30], building on the policy iteration scheme developed in [31]. Most recently, [32]

demonstrated the application of deep learning in controlled epidemic models.

In parallel, recent studies have explored the integration of artificial intelligence (AI) with mechanistic epidemiological models to enhance predictive capabilities and inform public health interventions [33, 34]. These approaches highlight the potential of AI to address complexities in disease dynamics that traditional models may not fully capture.

Building on these advancements, our work focuses on leveraging PINNs for solving optimal control problems, specifically in the context of SIR models with vaccination strategies. The proposed approach not only provides an effective approximation of the minimum eradication time but also facilitates the synthesis of optimal control policies in a computationally efficient manner.

1.2. Contributions

This paper makes the following contributions:

- We propose a PINN-based framework to approximate the minimum eradication time in the controlled SIR model by solving the associated HJB equation in a mesh-free manner. Our approach enables the simultaneous computation of eradication times across all initial conditions, eliminating the need for retraining.
- By leveraging the known structural property that the optimal control in the controlled SIR model is a bang-bang control with at most one switching, we recover the optimal vaccination strategy efficiently via the DPP, thereby avoiding the complex control synthesis procedures commonly required in general optimal control problems.
- We provide a theoretical analysis based on NTK theory, which explains the effectiveness of the variable scaling method in training PINNs for solving the HJB equation. Furthermore, we introduce variable-specific (nonuniform) scaling to address the anisotropic behavior inherent in the PDE, improving both training stability and convergence.

1.3. Organization of the paper

The remainder of this paper is organized as follows: Section 2 presents preliminary results on the minimum eradication time in the context of HJB equations. Section 3 reviews variable scaling PINNs and includes an error analysis specific to our HJB equation. Section 4 details the training procedure, while Section 5 presents the experimental results. Finally, Section 6 concludes the paper by summarizing the key findings, discussing the limitations of the approach, and providing directions for future research.

2. HJB equation for the minimum eradication time

2.1. Minimum eradication time problem

Throughout the paper, we consider a time-homogeneous controlled SIR model where the infection and recovery rates are constant:

$$\beta(t) \equiv \beta \quad \text{and} \quad \gamma(t) \equiv \gamma.$$

We consider a threshold $\mu > 0$ and the initial conditions $x \geq 0$ and $y \geq \mu$. Let $r(t) \in \mathcal{U} = \{r : [0, \infty) \to [0, 1]\}$ be a vaccination control. Under these settings, we define the eradication time as

$$u^r(x, y) := \min\{t > 0 : I(t) = \mu\},$$

where S^r and I^r satisfy

$$\begin{cases} \dot{S}^r = -\beta S^r I^r - rS^r, \\ \dot{I}^r = \beta S^r I^r - \gamma I^r, \end{cases} \tag{2.1}$$

with $(S^r(0), I^r(0)) = (x, y)$.

A crucial property of u^r is that for each $t \in [0, u^r(x, y)]$,

$$u^r(x, y) = t + u^r(S^r(t), I^r(t)), \tag{2.2}$$

which is known as the DPP. This relationship can be interpreted as follows: at time t, the remaining eradication time from the state $(S^r(t), I^r(t))$ is $u^r(x, y)$.

Finally, the minimum eradication time is defined as

$$u(x, y) := \min_{r \in \mathcal{U}} u^r(x, y). \tag{2.3}$$

The mathematical properties of this value function have been extensively studied in [6]. For the convenience of readers, we summarize the theoretical results provided in [6].

Thanks to (2.2), it is known that u is the unique viscosity solution to the following HJB equation.

Theorem 1 (Theorem 1.2 of [6])**.** *For $\mu > 0$, the value function u defined in* (2.3) *is the unique viscosity solution to*

$$\beta xy\partial_x u + x(\partial_x u)^+ + (\gamma - \beta x)y\partial_y u = 1 \quad in \quad (0, \infty) \times (\mu, \infty), \tag{2.4}$$

with the boundary conditions

$$u(0, y) = \frac{1}{\gamma} \ln\left(\frac{y}{\mu}\right) \quad for \quad y \geq \mu,$$

and

$$u(x, \mu) = 0 \quad for \quad 0 \leq x \leq \frac{\gamma}{\beta}.$$

In the next section, we proceed by identifying the optimal control that minimizes the eradication time.

2.2. Optimal bang-bang control

With the value function $u(x, y)$, characterized as the unique viscosity solution to the HJB equation (2.4), we now turn our attention to identifying the corresponding optimal control. In the HJB framework, the optimal cost is encoded in the value function, and the structure of the optimal control is identified through minimization of the Hamiltonian. In our setting, the dynamics are affine in the control variable, which implies that the optimal control takes the form of bang-bang [35].

To make this connection more explicit, we invoke the PMP [8], which provides a necessary condition for optimality in terms of the dynamics of adjoint variables. In our model, the PMP reveals

that any optimal vaccination strategy $r(t) \in [0,1]$ must satisfy a bang-bang structure, meaning that $r(t) \in \{0,1\}$ for almost every $t \in [0, u^r]$. The switching behavior is determined by the sign of the adjoint variable, which evolves according to a differential equation coupled with the state dynamics [4, 6, 35].

Moreover, in the time-homogeneous controlled SIR model with constant infection and recovery rates, it is known [4] that the optimal control switches at most once from $r(t) = 0$ to $r(t) = 1$. This result follows from a careful analysis of the monotonicity properties of the adjoint state. Consequently, we restrict our attention to switching controls of the form

$$r_\tau(t) = \begin{cases} 0, & t < \tau, \\ 1, & t \geq \tau. \end{cases} \tag{2.5}$$

Here, r_τ and τ are referred to as the switching control and switching time, respectively.

We now recall some well-known results on the minimum eradication time and the optimal switching time.

Theorem 2 (Theorem 1.4 of [6])**.** *Let $\mu > 0$, $x \geq 0$, and $y \geq \mu$. Then,*

$$u(x,y) = \min_{\tau \geq 0} \{\tau + u^{r_0}(S(\tau), I(\tau))\}, \tag{2.6}$$

where $S(t)$ and $I(t)$ satisfy the uncontrolled SIR model:

$$\begin{cases} \dot{S} = -\beta S I, \\ \dot{I} = \beta S I - \gamma I, \end{cases} \tag{2.7}$$

with $(S(0), I(0)) = (x, y)$. Moreover, any τ for which the minimum in (2.6) *is achieved is the switching time of the optimal switching control.*

To solve for the optimal switching time τ from (2.6), it is necessary to compute u^{r_0}. Recalling the DPP (2.2), when $r \equiv 1$, i.e., $r = r_0$, we have the following identity:

$$u^{r_0}(x,y) = t + u^{r_0}(S^{r_0}(t), I^{r_0}(t)), \tag{2.8}$$

where S^{r_0} and I^{r_0} satisfy

$$\begin{cases} \dot{S}^{r_0} &= -\beta S^{r_0} I^{r_0} - S^{r_0}, \\ \dot{I}^{r_0} &= \beta S^{r_0} I^{r_0} - \gamma I^{r_0}, \end{cases} \tag{2.9}$$

with $(S^{r_0}(0), I^{r_0}(0)) = (x, y)$. Taking the time derivative of both sides of (2.8), we deduce that

$$\begin{aligned} 0 &= 1 + \frac{d}{dt} u^{r_0}(S^{r_0}(t), I^{r_0}(t)) \\ &= 1 + \dot{S}^{r_0}(t) \partial_x u^{r_0} + \dot{I}^{r_0}(t) \partial_y u^{r_0} \\ &= 1 + (-\beta S^{r_0}(t) I^{r_0}(t) - S^{r_0}(t)) \partial_x u^{r_0} + (\beta S^{r_0}(t) I^{r_0}(t) - \gamma I^{r_0}(t)) \partial_y u^{r_0}. \end{aligned}$$

Setting $t = 0$ and using the dynamics (2.9), we obtain:

$$\beta x y \partial_x u^{r_0} + x \partial_x u^{r_0} + (\gamma - \beta x) y \partial_y u^{r_0} = 1 \quad \text{in} \quad (0, \infty) \times (\mu, \infty),$$

where u^{r_0} satisfies the same boundary conditions as u.

We finish this section with well-established properties of the optimal switching time τ^* such that $u(x, y) = u^{\tau^*}(x, y)$. We define the following:

$$\mathcal{S} := \{(x, y) : u(x, y) = u^{r_0}(x, y)\},$$

Corollary 6.2 in [6] yields

$$\begin{cases} \partial_x u^{r_0}(x, y) \geq 0 & \text{for} \quad (x, y) \in \mathcal{S}, \\ \partial_x u^{r_0}(S(\tau^*), I(\tau^*)) = 0 & \text{for} \quad (x, y) \in \mathcal{S}^C, \end{cases} \tag{2.10}$$

where (S, I) satisfies (2.7) with $(S(0), I(0)) = (x, y)$.

Furthermore, by Corollary 6.4 in [6], we have

$$\begin{cases} \partial_x u(x, y) \geq 0 & \text{for} \quad (x, y) \in \mathcal{S}, \\ \partial_x u(x, y) \leq 0 & \text{for} \quad (x, y) \in \mathcal{S}^C. \end{cases} \tag{2.11}$$

On the basis of the theoretical properties of the minimum eradication time, we propose a training procedure to solve (2.4) and compute the optimal switching time through a PINN framework, which does not require spatial discretization.

3. Variable scaling physics-informed neural networks

In this section, we explain the variable scaling physics-informed neural network (VS-PINN), which is a crucial component of our method. For completeness, we begin with an overview of PINNs.

3.1. Physics-informed neural networks

PINNs are trained using a loss function that enables the neural network to approximate a solution satisfying both the differential equation and the initial or boundary conditions. Specifically, we focus on solving a PDE with a boundary condition, as the problem we address falls into this category. Suppose we have a bounded open domain $\Omega \subset \mathbb{R}^d$ and the following equations:

$$\begin{aligned} \mathcal{D}[u](\mathbf{x}) &= f(\mathbf{x}) \quad \text{in} \quad \Omega, \\ u(\mathbf{x}) &= g(\mathbf{x}) \quad \text{on} \quad \partial\Omega, \end{aligned}$$

where $\mathcal{D}$ is a differential operator.

We train a neural network $u(\mathbf{x}; \theta)$ using the loss function

$$\mathcal{L} = \lambda_r \mathcal{L}_r + \lambda_b \mathcal{L}_b, \tag{3.1}$$

where the residual loss $\mathcal{L}_r$ and the boundary loss $\mathcal{L}_b$ are defined as

$$\mathcal{L}_r = \frac{1}{N_r} \sum_{i=1}^{N_r} |\mathcal{D}[u](\mathbf{x}_r^i) - f(\mathbf{x}_r^i)|^2, \quad \mathcal{L}_b = \frac{1}{N_b} \sum_{j=1}^{N_b} |u(\mathbf{x}_b^j) - g(\mathbf{x}_b^j)|^2.$$

Here, the residual data points $\mathbf{x}_r^i \in \Omega$ and the boundary data points $\mathbf{x}_b^j \in \partial\Omega$ are typically sampled randomly from uniform distributions. The weights λ_r, λ_b, and the number of data points N_r, N_b are tunable parameters.

In our problem, we consider $\Omega = (0, \infty) \times (\mu, \infty)$ for $\mu > 0$ and define the operators $\mathcal{D}$ and $\mathcal{D}^0$ as

$$\begin{aligned} \mathcal{D} &= \beta x y \partial_x + x(\partial_x)^+ + (\gamma - \beta x) y \partial_y, \\ \mathcal{D}^0 &= \beta x y \partial_x + x \partial_x + (\gamma - \beta x) y \partial_y, \end{aligned} \tag{3.2}$$

where $(\partial_x)^+ u = (\partial_x u)^+$. We solve for u and u^{r_0}, satisfying

$$\mathcal{D}[u] = 1 \quad \text{and} \quad \mathcal{D}^0[u^{r_0}] = 1 \quad \text{in} \quad \Omega, \tag{3.3}$$

with the boundary conditions

$$\begin{cases} u(0, y) = u^{r_0}(0, y) = \frac{1}{\gamma} \ln\left(\frac{y}{\mu}\right) & \text{for } y \geq \mu, \\ u(x, \mu) = u^{r_0}(x, \mu) = 0 & \text{for } 0 \leq x \leq \frac{\gamma}{\beta}. \end{cases} \tag{3.4}$$

A schematic diagram of the framework is presented in Figure 1.

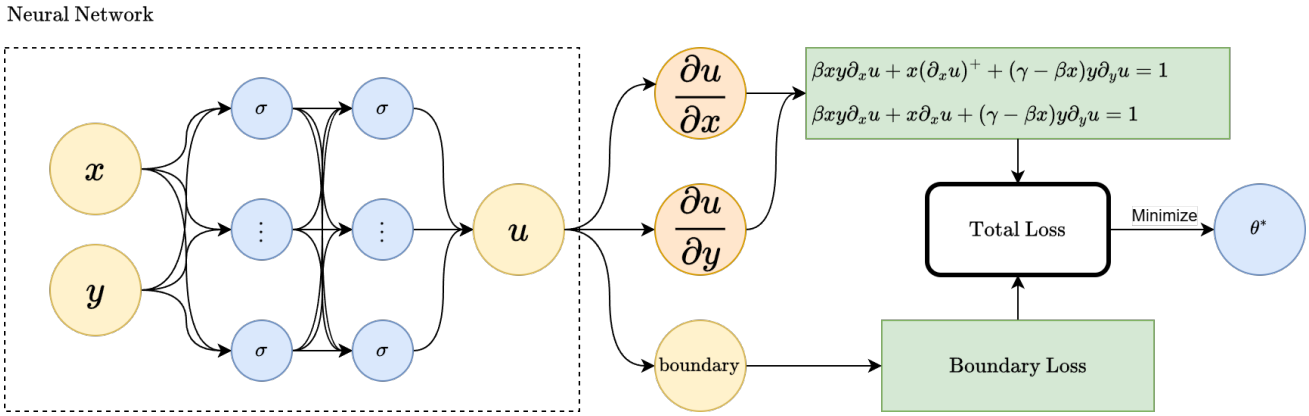


Figure 1. Training u and u^{r_0} under the PINN framework.

3.2. Variable scaling

In [11], Ko and Park proposed a simple method that improves the performance of PINNs. The idea is to scale the variables so that the domain of the target function is magnified, making the function less stiff.

Let us begin with a change of variable $\mathbf{x} = \hat{\mathbf{x}}/N$ and setting

$$\hat{u}(\hat{\mathbf{x}}) := u(\hat{\mathbf{x}}/N).$$

The function $\hat{u}$ is a scaled version of the function u, with its domain expanded by a factor of N, causing its stiffness to decrease by a factor of N compared with u. Thus, our goal is to train $\hat{u}$ instead of directly training u. After training $\hat{u}$, we can simply recover our original target u by substituting

$$u(\mathbf{x}) = \hat{u}(N\mathbf{x}).$$

Unlike [11], we use different scaling parameters N_x, N_y for each component x, y and also apply translations, which means we use

$$\hat{x} = N_x x + b_x, \quad \hat{y} = N_y y + b_y.$$

We employ this approach to address the anisotropic behavior in PDEs such as our HJB equation, where the dynamics exhibit asymmetry across different variables x and y. By doing so, we effectively mitigate the imbalance in gradient magnitudes during training and improve the optimization's stability. Thus, after training $\hat{u}$, we recover u by

$$u(x, y) = \hat{u}(N_x x + b_x, N_y y + b_y).$$

Recall that the differential operator of our problem in (3.2) takes the form of $\mathcal{D} = F(x, y, \partial_x, \partial_y)$. The modified version of our problem has the domain $\hat{\Omega} = (b_x, \infty) \times (N_y\mu + b_y, \infty)$.

By the chain rule,

$$\begin{aligned}\partial_x u(x, y) &= \partial_{\hat{x}} u(x, y) \cdot \partial_x \hat{x} = \partial_{\hat{x}} \hat{u}(\hat{x}, \hat{y}) \cdot N_x,\\ \partial_y u(x, y) &= \partial_{\hat{y}} u(x, y) \cdot \partial_y \hat{y} = \partial_{\hat{y}} \hat{u}(\hat{x}, \hat{y}) \cdot N_y,\end{aligned}$$

we define the operators $\hat{\mathcal{D}}$ and $\hat{\mathcal{D}}^0$ as

$$\hat{\mathcal{D}} = \beta \frac{(\hat{x} - b_x)(\hat{y} - b_y)}{N_y} \partial_{\hat{x}} + (\hat{x} - b_x)(\partial_{\hat{x}})^+ + (\gamma - \beta \frac{\hat{x} - b_x}{N_x})(\hat{y} - b_y)\partial_{\hat{y}},$$

and

$$\hat{\mathcal{D}}^0 = \beta \frac{(\hat{x} - b_x)(\hat{y} - b_y)}{N_y} \partial_{\hat{x}} + (\hat{x} - b_x)\partial_{\hat{x}} + (\gamma - \beta \frac{\hat{x} - b_x}{N_x})(\hat{y} - b_y)\partial_{\hat{y}}.$$

We then solve for $\hat{u}$ and $\hat{u}^{r_0}$, satisfying

$$\hat{\mathcal{D}}[\hat{u}] = 1 \quad \text{and} \quad \hat{\mathcal{D}}^0[\hat{u}^{r_0}] = 1 \quad \text{in} \quad \hat{\Omega},$$

with

$$\begin{cases} \hat{u}(0, \hat{y}) = \hat{u}^{r_0}(0, \hat{y}) = \frac{1}{\gamma} \ln\left(\frac{\hat{y} - b_y}{N_y \mu}\right) & \text{for} \quad \hat{y} \geq N_y\mu + b_y, \\ \hat{u}(\hat{x}, \mu) = \hat{u}^{r_0}(\hat{x}, \mu) = 0 & \text{for} \quad b_x \leq \hat{x} \leq \frac{N_x \gamma}{\beta} + b_x. \end{cases} \tag{3.5}$$

As a result, we should use the loss function in (3.1) modified by replacing $\mathcal{D}$ (or $\mathcal{D}^0$) and $f, g, u(\mathbf{x}; \theta)$ (or $u^{r_0}(\mathbf{x}; \theta)$) with $\hat{\mathcal{D}}$ (or $\hat{\mathcal{D}}^0$) and $\hat{f}, \hat{g}, \hat{u}(\hat{\mathbf{x}}; \theta)$ (or $\hat{u}^{r_0}(\hat{\mathbf{x}}; \theta)$) in order to train $\hat{u}(\hat{\mathbf{x}}; \theta)$ (or $\hat{u}^{r_0}(\hat{\mathbf{x}}; \theta)$), where $f(x) = \hat{f}(\hat{x}) = 1$, g and $\hat{g}$ are given in (3.4) and (3.5), respectively, that is

$$\begin{cases} g(0, y) = \frac{1}{\gamma} \ln\left(\frac{y}{\mu}\right) & \text{for } y \geq \mu, \\ g(x, \mu) = 0 & \text{for } 0 \leq x \leq \frac{\gamma}{\beta}, \end{cases}$$

and

$$\begin{cases} \hat{g}(0, \hat{y}) = \frac{1}{\gamma} \ln\left(\frac{\hat{y} - b_y}{N_y \mu}\right) & \text{for} \quad \hat{y} \geq N_y\mu + b_y, \\ \hat{g}(\hat{x}, \mu) = 0 & \text{for} \quad b_x \leq \hat{x} \leq \frac{N_x \gamma}{\beta} + b_x. \end{cases}$$

In the following section, we analyze the effect of the scaling factors.

3.3. Theoretical support via neural tangent kernel

We now establish the theoretical foundation for the efficiency of the variable scaling method in our problem through the lens of NTK [12], a widely used framework for analyzing the training dynamics of deep neural networks. In [25], the NTK theory was extended to PINNs.

Applying this theory, the authors of [11] demonstrated that the variable scaling method in PINNs applied to a simple one-dimensional Poisson equation enhances training efficiency. In this section, we establish a proof for Eq (2.4), which is more complex than the one-dimensional Poisson equation. To avoid redundancy, we focus on solving for u.

3.3.1. NTK

Given that the parameter θ of a PINN $u(\mathbf{x};\theta)$ is trained through the gradient flow

$$\frac{d\theta}{dt} = -\nabla_\theta \mathcal{L}$$

with respect to the loss (3.1) with $\lambda_r = \lambda_b = 1/2$, it is proved in [25] that the evolution of u and $\mathcal{D}[u]$ follows

$$\begin{bmatrix} \frac{du(\mathbf{x}_b;\theta(t))}{dt} \\ \frac{d\mathcal{D}[u](\mathbf{x}_r;\theta(t))}{dt} \end{bmatrix} = -\begin{bmatrix} K_{uu}(t) & K_{ur}(t) \\ K_{ru}(t) & K_{rr}(t) \end{bmatrix} \begin{bmatrix} u(\mathbf{x}_b;\theta(t)) - g(\mathbf{x}_b) \\ \mathcal{D}[u](\mathbf{x}_r;\theta(t)) - f(\mathbf{x}_r) \end{bmatrix},$$

where $K_{uu}(t) \in \mathbb{R}^{N_b \times N_b}$, $K_{rr}(t) \in \mathbb{R}^{N_r \times N_r}$, and $K_{ru}(t) = [K_{ur}(t)]^\top \in \mathbb{R}^{N_r \times N_b}$, whose (i, j)th entries are given by

$$\begin{aligned} (K_{uu})_{ij}(t) &= \left\langle \frac{du(\mathbf{x}_b^i;\theta(t))}{d\theta}, \frac{du(\mathbf{x}_b^j;\theta(t))}{d\theta} \right\rangle, \\ (K_{ru})_{ij}(t) &= \left\langle \frac{d\mathcal{D}[u](\mathbf{x}_r^i;\theta(t))}{d\theta}, \frac{du(\mathbf{x}_b^j;\theta(t))}{d\theta} \right\rangle, \\ (K_{rr})_{ij}(t) &= \left\langle \frac{d\mathcal{D}[u](\mathbf{x}_r^i;\theta(t))}{d\theta}, \frac{d\mathcal{D}[u](\mathbf{x}_r^j;\theta(t))}{d\theta} \right\rangle. \end{aligned}$$

We call the matrix $K(t) = \begin{bmatrix} K_{uu}(t) & K_{ur}(t) \\ K_{ru}(t) & K_{rr}(t) \end{bmatrix}$ the NTK of the training dynamics of $u(\mathbf{x};\theta)$ via PINN.

In [25], it is proved that the NTK of the PINN at initialization, i.e., the kernel at $t = 0$, converges to a deterministic kernel and remains the same in the infinite-width limit when $\theta(0)$ is assumed to follow the standard normal distribution. In [11], it is shown that variable scaling can enhance the performance of PINNs by analyzing how the initial NTK of the one-dimensional Poisson equation evolves with variable scaling. We follow their argument to deduce a similar result for Eq (2.4).

By definition, $K(t)$ is positive semi-definite. The eigenvalues of $K(t)$ are related to the convergence rate of the training dynamics. Since it is difficult to directly compute all eigenvalues of $K(t)$, we instead consider the average of the eigenvalues $\frac{\mathrm{Tr}(K(t))}{N_r+N_b}$. It is a weighted average of $\frac{\mathrm{Tr}(K_{uu}(t))}{N_b}$ and $\frac{\mathrm{Tr}(K_{rr}(t))}{N_r}$ as follows:

$$\frac{\mathrm{Tr}(K(t))}{N_r + N_b} = \frac{N_b}{N_r + N_b}\frac{\mathrm{Tr}(K_{uu}(t))}{N_b} + \frac{N_r}{N_r + N_b}\frac{\mathrm{Tr}(K_{rr}(t))}{N_r}.$$

For notational simplicity, we set the translation parameters $b_x = b_y = 0$ and omit the hat ($\hat{}$) notation in this subsection as follows:

$$\mathcal{D} = \beta\frac{xy}{N_y}\partial_x + x(\partial_x)^+ + (\gamma - \beta\frac{x}{N_x})y\partial_y,$$

and let our neural network be

$$u(x, y; \theta) = \frac{1}{\sqrt{d_1}}\sum_{k=1}^{d_1} W_2^k\sigma(W_1^{1k}x + W_1^{2k}y + b_1^k) + b_2, \tag{3.6}$$

with $\sigma = \max\{0, x^3\}$, since the activation function is required to be twice differentiable in NTK theory [12, 25] to ensure stationarity of NTK. Among the twice differentiable activation functions, we focused on the cubic rectified linear unit (ReLU) activation function in our theoretical analysis, primarily for reasons of analytical tractability. Specifically, for commonly used activation functions σ such as a sigmoid or hyperbolic tangent, computing expectations like $\mathbb{E}[\sigma(X)]$ when $X \sim N(0, \delta^2)$ becomes complex and often intractable in closed form. Although our theoretical analysis is based on a strong assumption, our experimental results indicate that the variable scaling method remains effective for a broader class of activation functions. We initialize the parameters $W_1^{ik}, W_2^k, b_1^k, b_2$ from the standard normal distribution $\mathcal{N}(0, 1)$.

We focus on training a PINN within the rectangular domain

$$[0, N_x\ell_x] \times [N_y\mu, N_y(\mu + \ell_y)], \tag{3.7}$$

and introduce an augmented boundary given by

$$[0, N_x\ell_x] \times \{N_y\mu\} \cup \{0\} \times [N_y\mu, N_y(\mu + \ell_y)] \cup [0, N_x\ell_x] \times \{N_y(\mu + \ell_y)\}. \tag{3.8}$$

Although this approach necessitates additional simulations to obtain ground-truth data for the augmented boundary given by (3.8), the associated computational cost is significantly lower compared with simulating the entire two-dimensional domain. To proceed, we sample the boundary points $\{x_b^j, y_b^j\}_{j=1}^{N_b}$ uniformly from the augmented boundary (3.8) and sample the residual points $\{x_r^i, y_r^i\}_{i=1}^{N_r}$ uniformly from the restricted domain (3.7). Note that the domain (3.7) is obtained by scaling the domain $[0, \ell_x] \times [\mu, \mu + \ell_y]$ ($\ell_x, \ell_y > 0$).

3.3.2. Convergence of NTK for HJB

We now present the main result, which demonstrates that the average of eigenvalues of the deterministic kernel grows in the scaling factors.

Theorem 3. *Under the sampling regime of data points and the differential operator as described above, the following convergences hold as the width d_1 of* (3.6) *goes to infinity:*

$$\frac{\mathrm{Tr}(K_{uu}(0))}{N_b} \xrightarrow{\mathcal{P}} O\left((N_x^2 + N_y^2 + 1)^3\right) \quad \textit{and} \quad \frac{\mathrm{Tr}(K_{rr}(0))}{N_r} \xrightarrow{\mathcal{P}} O\left(P(N_x, N_y)\right),$$

where $\xrightarrow{\mathcal{P}}$ represents the convergence in probability and $P(x, y)$ is a degree 3 homogeneous polynomial in x^2 and y^2 with positive coefficients. Moreover, if $N_x = N_y = N$, then

$$\frac{\mathrm{Tr}(K(0))}{N_r + N_b} \xrightarrow{\mathcal{P}} O(N^6)^*.$$

*$f(x) = O(g(x))$ implies that constant $C > 0$ and x_0 exist such that $|f(x)| \le C|g(x)|$ for all $x \ge x_0$.

Before presenting the proof, we state a useful lemma regarding the moment bounds of Gaussian distributions, which will be used repeatedly throughout the proof. The proof of the lemma is omitted and can be found in [36].

Lemma 1. *If $X \sim \mathcal{N}(0, \delta^2)$, then*

$$\mathbb{E}[X^{2n}] = \frac{(2n)!}{2^n n!}\delta^{2n} \quad \textit{and} \quad \mathbb{E}[X^{2n-1}] = 0 \quad \forall n \in \mathbb{N}.$$

Proof of Theorem 3. We first compute the limit of $\mathrm{Tr}(K_{uu}(0))/N_b$ as the width d_1 goes to infinity. From the definition, we see that

$$\frac{\mathrm{Tr}(K_{uu}(0))}{N_b} = \frac{1}{N_b}\sum_{j=1}^{N_b}\left\langle \frac{du(x_b^j, y_b^j; \theta)}{d\theta}, \frac{du(x_b^j, y_b^j; \theta)}{d\theta}\right\rangle,$$

and each summand is decomposed as

$$\begin{aligned}\left\langle \frac{du(x_b^j, y_b^j; \theta)}{d\theta}, \frac{du(x_b^j, y_b^j; \theta)}{d\theta}\right\rangle &= \left\langle \frac{du(x_b^j, y_b^j; \theta)}{dW_1^1}, \frac{du(x_b^j, y_b^j; \theta)}{dW_1^1}\right\rangle + \left\langle \frac{du(x_b^j, y_b^j; \theta)}{dW_1^2}, \frac{du(x_b^j, y_b^j; \theta)}{dW_1^2}\right\rangle \\ &+ \left\langle \frac{du(x_b^j, y_b^j; \theta)}{dW_2}, \frac{du(x_b^j, y_b^j; \theta)}{dW_2}\right\rangle + \left\langle \frac{du(x_b^j, y_b^j; \theta)}{db_1}, \frac{du(x_b^j, y_b^j; \theta)}{db_1}\right\rangle + \left\langle \frac{du(x_b^j, y_b^j; \theta)}{db_2}, \frac{du(x_b^j, y_b^j; \theta)}{db_2}\right\rangle.\end{aligned} \tag{3.9}$$

For notational convenience, we write

$$X_1 := W_1^{1k} \sim \mathcal{N}(0,1), \quad Y_1 := W_1^{2k} \sim \mathcal{N}(0,1), \quad X_2 := W_2^k \sim \mathcal{N}(0,1), \quad Z := b_1^k \sim \mathcal{N}(0,1).$$

Since X_1, Y_1, and Z are independent, we have

$$S := X_1 x_b^j + Y_1 y_b^j + Z \sim \mathcal{N}(0, (x_b^j)^2 + (y_b^j)^2 + 1).$$

The first term of (3.9) is then expressed as

$$\left\langle \frac{du(x_b^j, y_b^j; \theta)}{d\theta}, \frac{du(x_b^j, y_b^j; \theta)}{d\theta}\right\rangle = \frac{1}{d_1}\sum_{k=1}^{d_1}\left(W_2^k \sigma'(W_1^{1k} x_b^j + W_1^{2k} y_b^j + b_1^k) x_b^j\right)^2.$$

By the law of large numbers, this term converges in probability to

$$(x_b^j)^2 \mathbb{E}[X_2^2 \sigma'(S)^2] = (x_b^j)^2 \mathbb{E}[X_2^2]\mathbb{E}[\sigma'(S)^2] = (x_b^j)^2 \mathbb{E}[\sigma'(S)^2]$$

as d_1 goes to infinity, where the first equality holds, since X_2 and S are independent. As

$$\sigma'(S) = \begin{cases} 3S^2, & \text{if } S \geq 0, \\ 0, & \text{if } S < 0, \end{cases}$$

we have

$$\mathbb{E}[\sigma'(S)^2] = \frac{9}{2}\mathbb{E}[S^4] = O\left(((x_b^j)^2 + (y_b^j)^2 + 1)^2\right), \tag{3.10}$$

where the last equality follows from Lemma 1 and $S \sim \mathcal{N}(0, (x_b^j)^2 + (y_b^j)^2 + 1)$. Thus, the first term of (3.9) is the order of

$$(x_b^j)^2 O(((x_b^j)^2 + (y_b^j)^2 + 1)^2).$$

The second term of (3.9) is

$$\left\langle \frac{du(x_b^j, y_b^j; \theta)}{dW_1^2}, \frac{du(x_b^j, y_b^j; \theta)}{dW_1^2} \right\rangle = \frac{1}{d_1} \sum_{k=1}^{d_1} \left(W_2^k \sigma(W_1^{1k} x_b^j + W_1^{2k} y_b^j + b_1^k) y_b^j \right)^2.$$

By the law of large numbers and (3.10), this term converges in probability to

$$(y_b^j)^2 \mathbb{E}[X_2^2 \sigma'(S)^2] = (y_b^j)^2 \mathbb{E}[X_2^2] \mathbb{E}[\sigma'(S)^2] = (y_b^j)^2 \mathbb{E}[\sigma'(S)^2] = (y_b^j)^2 O(((x_b^j)^2 + (y_b^j)^2 + 1)^2)$$

as d_1 goes to infinity.

The third term of (3.9) is

$$\left\langle \frac{du(x_b^j, y_b^j; \theta)}{dW_2}, \frac{du(x_b^j, y_b^j; \theta)}{dW_2} \right\rangle = \frac{1}{d_1} \sum_{k=1}^{d_1} \left(\sigma(W_1^{1k} x_b^j + W_1^{2k} y_b^j + b_1^k) \right)^2.$$

By the law of large numbers, this term converges in probability to $\mathbb{E}[\sigma(S)^2]$ as d_1 goes to infinity. Noticing that

$$\sigma(S)^2 = \begin{cases} S^6, & \text{if } S \geq 0, \\ 0, & \text{if } S < 0, \end{cases}$$

we deduce

$$\mathbb{E}[\sigma(S)^2] = \frac{1}{2} \mathbb{E}[S^6] = O(((x_b^j)^2 + (y_b^j)^2 + 1)^3)$$

by Lemma 1.

The fourth term of (3.9) is

$$\left\langle \frac{du(x_b^j, y_b^j; \theta)}{db_1}, \frac{du(x_b^j, y_b^j; \theta)}{db_1} \right\rangle = \frac{1}{d_1} \sum_{k=1}^{d_1} \left(W_2^k \sigma'(W_1^{1k} x_b^j + W_1^{2k} y_b^j + b_1^k) \right)^2.$$

By the law of large numbers and (3.10), this term converges in probability to

$$\mathbb{E}[\sigma'(S)^2] = O(((x_b^j)^2 + (y_b^j)^2 + 1)^2)$$

as d_1 goes to infinity.

Finally, the last term in (3.9) is

$$\left\langle \frac{du(x_b^j, y_b^j; \theta)}{db_2}, \frac{du(x_b^j, y_b^j; \theta)}{db_2} \right\rangle = 1.$$

Combining them all together, we deduce that

$$\left\langle \frac{du(x_b^j, y_b^j; \theta)}{d\theta}, \frac{du(x_b^j, y_b^j; \theta)}{d\theta} \right\rangle = O\left(((x_b^j)^2 + (y_b^j)^2 + 1)^3 \right).$$

Note that we sample (x_b^j, y_b^j) uniformly from the scaled boundary (3.8), and thus x_b^j and y_b^j are scaled with N_x and N_y, respectively. Thus, we conclude that

$$\frac{\mathrm{Tr}(K_{uu}(0))}{N_b} = \frac{1}{N_b}\sum_{j=1}^{N_b}\left\langle \frac{du(x_b^j, y_b^j;\theta)}{d\theta}, \frac{du(x_b^j, y_b^j;\theta)}{d\theta}\right\rangle = O\left((N_x^2 + N_y^2 + 1)^3\right). \tag{3.11}$$

Next, we compute the limit of $\mathrm{Tr}(K_{rr}(0))/N_r$ as the width d_1 goes to infinity. From the definition,

$$\frac{\mathrm{Tr}(K_{rr}(0))}{N_r} = \frac{1}{N_r}\sum_{i=1}^{N_r}\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{d\theta}, \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{d\theta}\right\rangle,$$

and each summand is decomposed as

$$\begin{aligned}
&\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{d\theta}, \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{d\theta}\right\rangle \\
&= \left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{dW_1^1}, \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{dW_1^1}\right\rangle + \left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{dW_1^2}, \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{dW_1^2}\right\rangle \\
&+ \left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{dW_2}, \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{dW_2}\right\rangle + \left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{db_1}, \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{db_1}\right\rangle \\
&+ \left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{db_2}, \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{db_2}\right\rangle.
\end{aligned} \tag{3.12}$$

Recall that

$$\mathcal{D}[u] = \beta\frac{xy}{N_y}\partial_x u + x(\partial_x u)^+ + (\gamma - \beta\frac{x}{N_x})y\partial_y u.$$

Computing the first derivatives of u, we get

$$\partial_x u(x_r^i, y_r^i;\theta) = \frac{1}{\sqrt{d_1}}\sum_{k=1}^{d_1} W_2^k W_1^{1k}\sigma'(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k)$$

and

$$\partial_y u(x_r^i, y_r^i;\theta) = \frac{1}{\sqrt{d_1}}\sum_{k=1}^{d_1} W_2^k W_1^{2k}\sigma'(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k).$$

Thus, we get

$$\begin{aligned}
\mathcal{D}[u(x_r^i, y_r^i;\theta)] = \beta\frac{x_r^i y_r^i}{N_y\sqrt{d_1}}&\sum_{k=1}^{d_1} W_2^k W_1^{1k}\sigma'(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k) \\
&+ \frac{x_r^i}{\sqrt{d_1}}\left(\sum_{k=1}^{d_1} W_2^k W_1^{1k}\sigma'(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k)\right)^+ \\
&+ (\gamma - \beta\frac{x_r^i}{N_x})\frac{y_r^i}{\sqrt{d_1}}\sum_{k=1}^{d_1} W_2^k W_1^{2k}\sigma'(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k).
\end{aligned}$$

The first term of (3.12) is

$$\begin{aligned}
&\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_1^1}, \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_1^1} \right\rangle \\
&= \frac{1}{d_1} \sum_{k=1}^{d_1} \Big[\frac{\beta x_r^i y_r^i}{N_y} W_2^k \left(W_1^{1k} \sigma''(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) x_r^i + \sigma'(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) \right) \\
&\quad + \delta x_r^i W_2^k \left(W_1^{1k} \sigma''(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) x_r^i + \sigma'(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) \right) \\
&\quad + (\gamma - \beta \frac{x_r^i}{N_x}) y_r^i W_2^k W_1^{1k} \sigma''(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) x_r^i \Big]^2,
\end{aligned}$$

where

$$\delta = \begin{cases} 1 & \text{if } \sum_{k=1}^{d_1} W_2^k W_1^{1k} \sigma'(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) \geq 0, \\ 0 & \text{if } \sum_{k=1}^{d_1} W_2^k W_1^{1k} \sigma'(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) < 0. \end{cases}$$

Temporarily denoting

$$\begin{aligned}
A_k &= \frac{\beta x_r^i y_r^i}{N_y} W_2^k W_1^{1k} \sigma''(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) x_r^i, \\
B_k &= \frac{\beta x_r^i y_r^i}{N_y} W_2^k \sigma'(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k), \\
C_k &= x_r^i W_2^k W_1^{1k} \sigma''(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) x_r^i, \\
D_k &= x_r^i W_2^k \sigma'(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k), \\
E_k &= (\gamma - \beta \frac{x_r^i}{N_x}) y_r^i W_2^k W_1^{1k} \sigma''(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) x_r^i,
\end{aligned}$$

and applying the power-mean inequality, we achieve

$$\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_1^1}, \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_1^1} \right\rangle \leq \frac{1}{d_1} \sum_{k=1}^{d_1} 5(A_k^2 + B_k^2 + C_k^2 + D_k^2 + E_k^2).$$

By the law of large numbers,

$$\begin{aligned}
&\frac{1}{d_1} \sum_{k=1}^{d_1} A_k^2 \xrightarrow{\mathcal{P}} \frac{\beta^2 (x_r^i)^4 (y_r^i)^2}{N_y^2} \mathbb{E}[X_1^2 \sigma''(S)^2], \\
&\frac{1}{d_1} \sum_{k=1}^{d_1} B_k^2 \xrightarrow{\mathcal{P}} \frac{\beta^2 (x_r^i)^2 (y_r^i)^2}{N_y^2} \mathbb{E}[\sigma'(S)^2], \\
&\frac{1}{d_1} \sum_{k=1}^{d_1} C_k^2 \xrightarrow{\mathcal{P}} (x_r^i)^2 \mathbb{E}[X_1^2 \sigma''(S)^2], \\
&\frac{1}{d_1} \sum_{k=1}^{d_1} D_k^2 \xrightarrow{\mathcal{P}} (x_r^i)^2 \mathbb{E}[\sigma'(S)^2], \\
&\frac{1}{d_1} \sum_{k=1}^{d_1} E_k^2 \xrightarrow{\mathcal{P}} (\gamma - \beta \frac{x_r^i}{N_x})^2 (x_r^i)^2 (y_r^i)^2 \mathbb{E}[X_1^2 \sigma''(S)^2].
\end{aligned}$$

where $S = X_1 x_r^i + Y_1 y_r^i + Z$ and $\xrightarrow{\mathcal{P}}$ denotes the convergence in probability.

By (3.10), we have $\mathbb{E}[\sigma'(S)^2] = O\left(((x_r^i)^2 + (y_r^i)^2 + 1)^2\right)$.

Using the fact

$$|\sigma''(X_1 x_r^i + Y_1 y_r^i + Z)|^2 \leq 36|X_1 x_r^i + Y_1 y_r^i + Z|^2,$$

we have

$$\begin{aligned}
&\mathbb{E}\left[X_1^2 \sigma''(S)^2\right] \\
&\leq \frac{36}{(2\pi)^{3/2}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^2(x^2(x_r^i)^2 + y^2(y_r^i)^2 + z^2 + 2xyx_r^i y_r^i + 2yzy_r^i + 2xzx_r^i) \exp\left(-\frac{x^2+y^2+z^2}{2}\right) \mathrm{d}z\mathrm{d}y\mathrm{d}x \\
&= 36(|x_r^i|^2\mathbb{E}[X_1^4] + (y_r^i)^2\mathbb{E}[X_1^2 Y_1^2] + \mathbb{E}[X_1^2 Z^2]) \\
&= 36\left(3(x_r^i)^2 + (y_r^i)^2 + 1\right).
\end{aligned} \tag{3.13}$$

Since we sample (x_r^i, y_r^i) from the scaled domain (3.7), x_r^i and y_r^i are scaled with N_x and N_y, respectively, as we adjust N_x, N_y. Thus, we have

$$\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_1^1}, \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_1^1} \right\rangle = O\left((N_x^4 + N_x^2 + N_x^2 N_y^2)(3N_x^2 + N_y^2 + 1) + N_x^2(N_x^2 + N_y^2 + 1)^2\right), \tag{3.14}$$

where we omitted other variables such as β and γ, as our primary focus is on the dependency on N_x and N_y.

To proceed, we now observe that the second term in (3.12) is written as

$$\begin{aligned}
&\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_1^2}, \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_1^2} \right\rangle \\
&= \frac{1}{d_1}\sum_{k=1}^{d_1}\Big[\frac{\beta x_r^i y_r^i}{N_y} W_2^k W_1^{1k} \sigma''(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) y_r^i \\
&\quad + \delta x_r^i W_2^k W_1^{1k} \sigma''(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) y_r^i \\
&\quad + (\gamma - \beta\frac{x_r^i}{N_x}) y_r^i W_2^k \left(W_1^{1k}\sigma''(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) y_r^i + \sigma'(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k)\right)\Big]^2.
\end{aligned}$$

Similarly, we temporarily denote

$$\begin{aligned}
A_k &= \frac{\beta x_r^i y_r^i}{N_y} W_2^k W_1^{1k} \sigma''(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) y_r^i, \\
B_k &= x_r^i W_2^k W_1^{1k} \sigma''(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) y_r^i, \\
C_k &= (\gamma - \beta\frac{x_r^i}{N_x}) y_r^i W_2^k W_1^{2k} \sigma''(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) y_r^i, \\
D_k &= (\gamma - \beta\frac{x_r^i}{N_x}) y_r^i W_2^k \sigma'(W_1^{1k} x_r^i + W_1^{2k} y_r^i + b_1^k) y_r^i,
\end{aligned}$$

and apply the power mean inequality to deduce

$$\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_1^2}, \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_1^2} \right\rangle \leq \frac{1}{d_1} \sum_{k=1}^{d_1} 4(A_k^2 + B_k^2 + C_k^2 + D_k^2).$$

By the law of large numbers, (3.10), (3.13), and the symmetry between $\mathbb{E}[X_1^2\sigma''(S)^2]$ and $\mathbb{E}[Y_1^2\sigma''(S)^2]$, we have

$$\begin{aligned}
\frac{1}{d_1}\sum_{k=1}^{d_1} A_k^2 &\xrightarrow{\mathcal{P}} \frac{\beta^2 (x_r^i)^2 (y_r^i)^4}{N_y^2}\mathbb{E}[X_1^2\sigma''(S)^2] && = O\left(N_x^2 N_y^2 (3N_x^2 + N_y^2 + 1)\right),\\
\frac{1}{d_1}\sum_{k=1}^{d_1} B_k^2 &\xrightarrow{\mathcal{P}} (x_r^i)^2 (y_r^i)^2 \mathbb{E}[X_1^2\sigma''(S)^2] && = O\left(N_x^2 N_y^2 (3N_x^2 + N_y^2 + 1)\right),\\
\frac{1}{d_1}\sum_{k=1}^{d_1} C_k^2 &\xrightarrow{\mathcal{P}} (\gamma - \beta\frac{x_r^i}{N_x})^2 (y_r^i)^4 \mathbb{E}[Y_1^2\sigma''(S)^2] && = O\left(N_y^4 (N_x^2 + 3N_y^2 + 1)\right),\\
\frac{1}{d_1}\sum_{k=1}^{d_1} D_k^2 &\xrightarrow{\mathcal{P}} (\gamma - \beta\frac{x_r^i}{N_x})^2 (y_r^i)^2 \mathbb{E}[\sigma'(S)^2] && = O\left(N_y^2 (N_x^2 + N_y^2 + 1)^2\right).
\end{aligned}$$

Thus, we get

$$\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_1^2}, \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_1^2} \right\rangle = O\left(N_x^2 N_y^2 (3N_x^2 + N_y^2 + 1) + N_y^4 (N_x^2 + 3N_y^2 + 1) + N_y^2 (N_x^2 + N_y^2 + 1)^2\right). \tag{3.15}$$

The third term of (3.12) is

$$\begin{aligned}
\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_2}, \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_2} \right\rangle = \frac{1}{d_1}\sum_{k=1}^{d_1} \Big[&\frac{\beta x_r^i y_r^i}{N_y} W_1^{1k}\sigma'(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k)\\
&+ \delta x_r^i W_1^{1k}\sigma'(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k)\\
&+ (\gamma - \beta\frac{x_r^i}{N_x}) y_r^i W_1^{2k}\sigma'(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k)\Big]^2.
\end{aligned}$$

Denoting

$$\begin{aligned}
A_k &= \frac{\beta x_r^i y_r^i}{N_y} W_1^{1k}\sigma'(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k),\\
B_k &= x_r^i W_1^{1k}\sigma'(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k),\\
C_k &= (\gamma - \beta\frac{x_r^i}{N_x}) y_r^i W_1^{2k}\sigma'(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k),
\end{aligned}$$

we have

$$\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_2}, \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{dW_2} \right\rangle \leq \frac{1}{d_1} \sum_{k=1}^{d_1} 3(A_k^2 + B_k^2 + C_k^2).$$

By the law of large numbers,

$$\frac{1}{d_1}\sum_{k=1}^{d_1} A_k^2 \xrightarrow{\mathcal{P}} \frac{\beta^2 (x_r^i)^2 (y_r^i)^2}{N_y^2}\mathbb{E}[X_1^2\sigma'(S)^2],$$
$$\frac{1}{d_1}\sum_{k=1}^{d_1} B_k^2 \xrightarrow{\mathcal{P}} (x_r^i)^2\mathbb{E}[X_1^2\sigma'(S)^2],$$
$$\frac{1}{d_1}\sum_{k=1}^{d_1} C_k^2 \xrightarrow{\mathcal{P}} (\gamma - \beta\frac{x_r^i}{N_x})^2 (y_r^i)^2\mathbb{E}[Y_1^2\sigma'(S)^2].$$

By Lemma 1 and the fact that $\sigma'(x)^2 \le 9x^4$, we have

$$\begin{aligned}
&\mathbb{E}[X_1^2\sigma'(S)^2]\\
&\le \frac{9}{(2\pi)^{3/2}}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^2(xx_r^i + yy_r^i + z)^4 \mathrm{d}z\mathrm{d}y\mathrm{d}x\\
&= \frac{9}{(2\pi)^{3/2}}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^2(x^4(x_r^i)^4 + y^4(y_r^i)^4 + z^4 + 6x^2y^2(x_r^iy_r^i)^2 + 6y^2z^2(y_r^i)^2 + 6x^2z^2(x_r^i)^2)\mathrm{d}z\mathrm{d}y\mathrm{d}x\\
&= 9(15(x_r^i)^4 + 3(y_r^i)^4 + 3 + 18(x_r^iy_r^i)^2 + 6(y_r^i)^2 + 18(x_r^i)^2) = O(5(x_r^i)^4 + (y_r^i)^4 + 6(x_r^iy_r^i)^2).
\end{aligned}$$

Similarly,

$$\mathbb{E}[Y_1^2\sigma'(S)^2] \le 9(15(y_r^i)^4 + 3(x_r^i)^4 + 3 + 18(x_r^iy_r^i)^2 + 6(x_r^i)^2 + 18(y_r^i)^2) = O((x_r^i)^4 + 5(y_r^i)^4 + 6(x_r^iy_r^i)^2).$$

Thus, we have

$$\begin{aligned}
\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{dW_2}, \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{dW_2} \right\rangle &= O\left(N_x^2(5N_x^4 + N_y^4 + 6N_x^2N_y^2) + N_y^2(N_x^4 + 5N_y^4 + 6N_x^2N_y^2)\right)\\
&= O(5N_x^6 + 7N_x^4N_y^2 + 7N_x^2N_y^4 + 5N_y^6).
\end{aligned} \tag{3.16}$$

The fourth term of (3.12) is

$$\begin{aligned}
\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{db_1}, \frac{d\mathcal{D}[u(x_r^i, y_r^i;\theta)]}{db_1} \right\rangle = \frac{1}{d_1}\sum_{k=1}^{d_1}\Big[&\frac{\beta x_r^iy_r^i}{N_y}W_2^kW_1^{1k}\sigma''(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k)\\
&+ \delta x_r^iW_2^kW_1^{1k}\sigma''(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k)\\
&+ (\gamma - \beta\frac{x_r^i}{N_x})y_r^iW_2^kW_1^{2k}\sigma''(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k)\Big]^2.
\end{aligned}$$

Denoting

$$\begin{aligned}
A_k &= \frac{\beta x_r^iy_r^i}{N_y}W_2^kW_1^{1k}\sigma''(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k),\\
B_k &= x_r^iW_2^kW_1^{1k}\sigma''(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k),\\
C_k &= (\gamma - \beta\frac{x_r^i}{N_x})y_r^iW_2^kW_1^{2k}\sigma''(W_1^{1k}x_r^i + W_1^{2k}y_r^i + b_1^k),
\end{aligned}$$

we have

$$\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{db_1}, \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{db_1} \right\rangle \leq \frac{1}{d_1} \sum_{k=1}^{d_1} 3(A_k^2 + B_k^2 + C_k^2).$$

By the law of large numbers,

$$\frac{1}{d_1} \sum_{k=1}^{d_1} A_k^2 \xrightarrow{\mathcal{P}} \frac{\beta^2 (x_r^i)^2 (y_r^i)^2}{N_y^2} \mathbb{E}[X_1^2 \sigma''(S)^2],$$

$$\frac{1}{d_1} \sum_{k=1}^{d_1} B_k^2 \xrightarrow{\mathcal{P}} (x_r^i)^2 \mathbb{E}[X_1^2 \sigma''(S)^2],$$

$$\frac{1}{d_1} \sum_{k=1}^{d_1} C_k^2 \xrightarrow{\mathcal{P}} (\gamma - \beta \frac{x_r^i}{N_x})^2 (y_r^i)^2 \mathbb{E}[Y_1^2 \sigma''(S)^2].$$

By (3.13), we have

$$\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{db_1}, \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{db_1} \right\rangle = O\left(N_x^2(3N_x^2 + N_y^2 + 1) + N_y^2(N_x^2 + 3N_y^2 + 1)\right). \tag{3.17}$$

Lastly, the fifth term of (3.12) is simply

$$\left\langle \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{db_2}, \frac{d\mathcal{D}[u(x_r^i, y_r^i; \theta)]}{db_2} \right\rangle = 0. \tag{3.18}$$

Combining (3.14)–(3.18), we obtain

$$\frac{\mathrm{Tr}(K_{rr}(0))}{N_r} = O\left(P(N_x, N_y)\right), \tag{3.19}$$

where $P(x, y)$ is a degree 3 homogeneous polynomial in x^2 and y^2 with positive coefficients.

By (3.11) and (3.19), we finally conclude that the average convergence rate of NTK increases as the variable scaling factors N_x and N_y increase. In the special case where we set $N_x = N_y = N$, the right-hand sides of both (3.11) and (3.19) become $O(N^6)$. □

In practice, we use gradient descent in training, which is not identical to the ideal gradient flow. Consequently, arbitrarily increasing N_x and N_y may lead to excessively large parameter updates during a single step of gradient descent, potentially destabilizing the training process. Therefore, it is crucial to empirically determine an appropriate value for N_x and N_y to ensure stable learning.

4. Methodology

Our goal is to approximate u and u^{r_0} using the PINN framework without discretizing the spatial domain and to use these results to estimate the optimal switching control via (2.6) with high accuracy. To achieve this, we train u and u^{r_0} separately by solving (3.3) within a fixed domain $D := [0, \ell_x] \times [\mu, \mu + \ell_y]$, where ℓ_x, ℓ_y and $\mu > 0$ are given. While the original domain is unbounded, we restrict it to a bounded subset for training, since sampling can only be performed over a finite

region. To reduce potential boundary effects near the truncated region along the positive y-axis, we impose suitable boundary conditions. This approximation is physically justified, as the total population is finite and the state variables x and y are inherently bounded in real-world epidemic settings.

We then identify the optimal control r via (2.6), which minimizes the eradication time, by modeling τ as a neural network function. For the ease of training the switching time τ, we introduce another neural network to approximate the uncontrolled dynamics $(S(t), I(t))$ satisfying (2.7).

Figure 2 outlines the full workflow, where we first train u and u^{r_0} using PINNs, then learn the uncontrolled system dynamics, and finally optimize τ using the DPP framework.

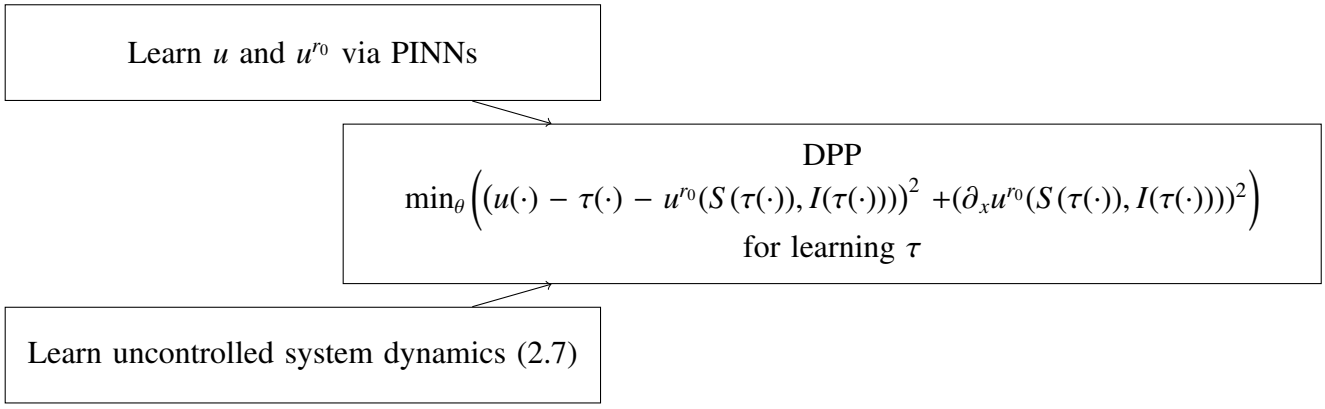


Figure 2. Procedure for solving τ using the DPP framework, with trained u and u^{r_0}, and the learned uncontrolled dynamics $(S(t), I(t))$.

4.1. Solving HJB equations

Let us begin by solving

$$\beta xy\partial_x u + x(\partial_x u)^+ + (\gamma - \beta x)y\partial_y u = 1 \quad \text{in} \quad [0, \ell_x] \times [\mu, \mu + \ell_y],$$

and

$$\beta xy\partial_x u^{r_0} + x\partial_x u^{r_0} + (\gamma - \beta x)y\partial_y \quad \text{in} \quad [0, \ell_x] \times [\mu, \mu + \ell_y],$$

via PINNs, or equivalently,

$$\mathcal{D}[u] = 1 \quad \text{and} \quad \mathcal{D}^0[u^{r_0}] = 1 \quad [0, \ell_x] \times [\mu, \mu + \ell_y],$$

where

$$\mathcal{D} = \beta xy\partial_x + x(\partial_x)^+ + (\gamma - \beta x)y\partial_y \quad \text{and} \quad \mathcal{D}^0 = \beta xy\partial_x + x\partial_x + (\gamma - \beta x)y\partial_y.$$

To leverage the idea of the VS-PINN, we set

$$u(x, y; \theta_1) = \text{NN}\Big(N_x x + b_x, N_y y + b_y; \theta_1\Big) \quad \text{and} \quad u^{r_0}(x, y; \theta_2) = \text{NN}\Big(N_x x + b_x, N_y y + b_y; \theta_2\Big),$$

where $\mathrm{NN}(\cdot;\theta_1)$ and $\mathrm{NN}(\cdot;\theta_2)$ represent fully connected neural networks parametrized by θ_1 and θ_2, respectively. Note that N_x, N_y, b_x and b_y are not trained during training. We then explicitly define the loss function.

Residual loss: with the prespecified training data $\{(x_r^i, y_r^i)\}_{i=1}^{N_r} \in [0, \ell_x] \times [\mu, \ell_y + \mu]$ for $N_r \in \mathbb{N}$, the residual loss associated with u is given as

$$\mathcal{L}_r[u(\cdot;\theta_1)] = \frac{1}{N_r}\sum_{i=1}^{N_r}(\mathcal{D}[u(x_r^i, y_r^i;\theta_1)] - 1)^2.$$

Similarly, the residual loss corresponding to u^{r_0} is defined as

$$\mathcal{L}_r^0[u^{r_0}(\cdot;\theta_2)] = \frac{1}{N_r}\sum_{i=1}^{N_r}(\mathcal{D}^0[u^{r_0}(x_r^i, y_r^i;\theta_2)] - 1)^2.$$

Boundary loss: we define the boundary loss function on

$$\Gamma := \underbrace{[0, \ell_x] \times \{\mu\}}_{=:D_1} \cup \underbrace{\{0\} \times [\mu, \mu + \ell_y]}_{=:D_2} \cup \underbrace{[0, \ell_x] \times \{\mu + \ell_y\}}_{=:D_3} \subset \partial D, \tag{4.1}$$

where ∂D denotes the boundary of the domain D. Given $N_b^k \in \mathbb{N}$, we randomly sample N_b^k points from D_k and denote them by $\{(x_{b,k}^j, y_{b,k}^j)\}_{j=1}^{N_b^k}$ for $k = 1, 2, 3$. With $N_b = N_b^1 + N_b^2 + N_b^3$, let

$$\mathcal{L}_b[u(\cdot;\theta_1)] = \frac{1}{N_b}\sum_{k=1}^{3}\sum_{j=1}^{N_b^k}\left(u(x_{b,k}^j, y_{b,k}^j;\theta_1) - u(x_{b,k}^j, y_{b,k}^j)\right)^2. \tag{4.2}$$

We note that the boundary loss for learning u^{r_0} is identical to that used for learning u.

One key challenge is the approximation of $u(x_{b,k}^j, y_{b,k}^j)$ at the boundary points for each j and k. To address this, we revisit an intrinsic property of optimal control for the minimum eradication time, demonstrating that the optimal vaccination strategy takes the form of (2.5). Accordingly, we propose Algorithm 1 for numerical implementation. Leveraging the structure of the optimal control, we compute the first time when I falls below μ by using the controls $r_{sd\tau}$ for $s \in \mathbb{N} \cup \{0\}$ and $d\tau > 0$. The solution to the controlled SIR model (2.1) is approximated using the fourth-order Runge-Kutta method with a step size $dt > 0$.

4.2. Optimal switching control

The recovery of the optimal control is significantly simplified by the theoretical result from [4] that the optimal policy is of bang-bang form with at most one switch, as we pointed out in (2.5). This allows us to avoid general policy parameterization and instead directly optimize over a scalar switching time, as formalized via the dynamic programming principle.

The optimal control can be recovered efficiently by exploiting its known structure: as shown in [4] and (2.5), the policy takes a bang-bang form with at most one switch. This allows us to reduce the control synthesis to a scalar optimization over the switching time, bypassing general policy parameterization.

Algorithm 1 Find the minimum eradication time

Input: Time resolution of the optimal control $d\tau$, the maximum switching time L, the discretization size dt, the maximum iteration step M, the eradication threshold μ, and the initial susceptible and infectious population $x, y \in D$.
Output: $\tau > 0$ such that r_τ is an optimal control.
Set $(S_0, I_0) = (x, y)$ and $\tau = 0$.
while True **do**
    for $m = 0, 1, 2, \ldots, M$ **do**
        Compute S_{m+1}, I_{m+1} using the Runge-Kutta method corresponding to

$$\begin{cases} \dot{S} &= -\beta S I - r_\tau S, \\ \dot{I} &= \beta S I - \gamma I. \end{cases}$$

    end for
    $m_\tau := \min\{m' : I_{m'} \le \mu\}$.
    $\tau = \tau + d\tau$.
    Break if $\tau > L$.
end while
Return $u(x, y) = \min_\tau\{m_\tau\}$.

With u and u^{r_0} computed in the previous section, we now solve for the optimal switching time τ satisfying (2.6),

$$u(x, y) = \min_{\tau \ge 0}\{\tau + u^{r_0}(S(\tau), I(\tau))\},$$

which involves a complex and nonlinear optimization process. To avoid solving this optimization problem directly, we introduce a neural network $\tau(x, y; \theta)$ and propose optimizing the following problem:

$$\min_\theta \left(u(\cdot) - \tau(\cdot; \theta) - u^{r_0}(S(\tau(\cdot; \theta)), I(\tau(\cdot; \theta)))\right)^2,$$

where S and I satisfy the uncontrolled system of ODEs (2.7), that is,

$$\begin{cases} \dot{S} &= -\beta S I, \\ \dot{I} &= \beta S I - \gamma I, \end{cases}$$

with $(S(0), I(0)) = (x, y)$. Since the closed-form solutions of S and I are unavailable, we reduce the above optimization problem to

$$\min_\theta \left((u(\cdot) - \tau(\cdot; \theta) - u^{r_0}(S(\tau(\cdot; \theta); \omega), I(\tau(\cdot; \theta); \omega)))^2\right),$$

where we approximate $(S(t), I(t))$ using a neural network $w(x, y, t; \omega)$, that is,

$$(S(t), I(t)) \approx w(x, y, t; \omega).$$

This approximation corresponds to training a neural network $w(x, y, t; \omega)$ with $w(x, y, 0; \omega) = (x, y)$ that best mimics the uncontrolled dynamics $(S(t), I(t))$ satisfying (2.7) with $(S(0), I(0)) = (x, y)$.

4.2.1. Learning the uncontrolled SIR model

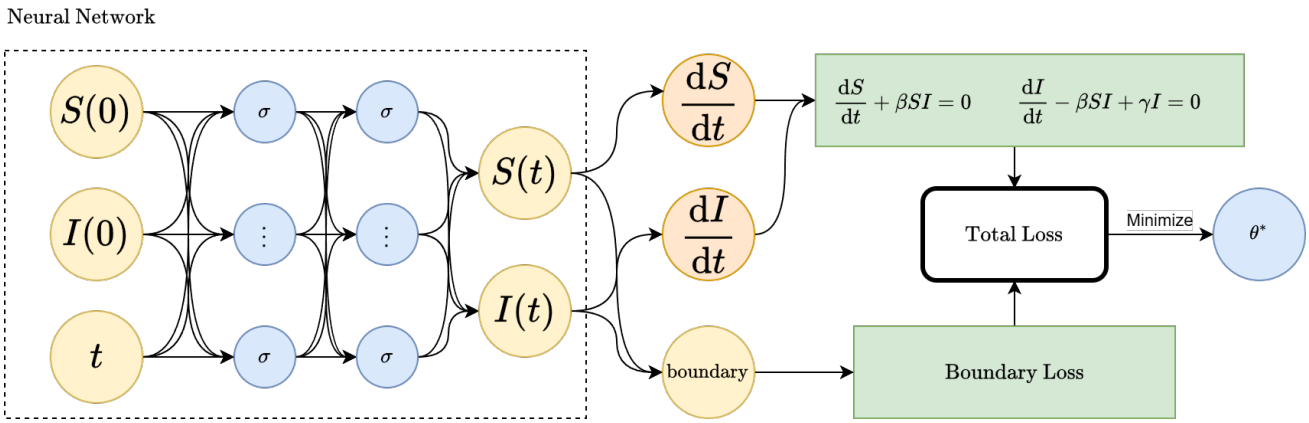


Figure 3. Training of the uncontrolled dynamics via the PINN framework.

Algorithm 2 Training of the dynamics

Input: The number of initial samples N_p, the number of interior collocation points N_{int}, the number of boundary points N_{bdry}, the number of temporal discretization steps N_T, the total time horizon T, and the timestep size $dt = T/N_T$.

Output: Trained neural network $w(x, y, t; \omega) \approx (S(t), I(t))$, approximating the solution of (2.7) with initial condition $(S(0), I(0)) = (x, y)$.

Sample the boundary data $\{(x^k_{\text{bdry}}, y^k_{\text{bdry}}, t_k)\}_{k=1}^{N_{\text{bdry}}} \subset \partial D \times \mathcal{T}$.

for $k = 1$ to N_{bdry} **do**

  Initialize $(S_0, I_0) \leftarrow (x^k_{\text{bdry}}, y^k_{\text{bdry}})$.

  for $m = 0$ to t_k **do**

    Compute (S_{t_k}, I_{t_k}) using the Runge-Kutta method applied to:

$$\begin{cases} \dot{S} = -\beta S I, \\ \dot{I} = \beta S I - \gamma I. \end{cases}$$

  end for

  Assign $w(x^k_{\text{bdry}}, y^k_{\text{bdry}}, t_k) \leftarrow (S_{t_k}, I_{t_k})$.

end for

while training has not converged **do**

  Sample the interior collocation points $\{(x^j_{\text{int}}, y^j_{\text{int}}, t_j)\}_{j=1}^{N_{\text{int}}} \subset D \times [0, T]$.

  Sample the initial points $\{(x^i_p, y^i_p)\}_{i=1}^{N_p} \subset D$.

  Compute the loss function $\mathcal{L}[w(\cdot; \omega)]$ as defined in (4.3).

  Update the parameters: $\omega \leftarrow \text{Adam}(\mathcal{L})$.

end while

We propose to train a neural network $w(x, y, t; \omega) : (x, y, t) \mapsto \mathbb{R}^2$ that approximates the flow of uncontrolled dynamics $(S(t), I(t))$ with $(S(0), I(0)) = (x, y)$ for all points $(x, y) \in D$. To train such a w, we first randomly choose interior points from the domain $D = [0, \ell_x] \times [\mu, \mu + \ell_y]$. For the stability of training, we further sample points from the boundary of D given by

$$\partial D = [0, \ell_x] \times \{\mu\} \cup \{0\} \times [\mu, \mu + \ell_y] \cup [0, \ell_x] \times \{\mu + \ell_y\} \cup \{\ell_x\} \times [\mu, \mu + \ell_y].$$

Let T, N_T, N_p, N_{int}, and N_{bdry} be given and set $dt = T/N_T$ and $\mathcal{T} := \{mdt : m = 0, 1, 2, ..., N_T\}$. With random samples $\{(x_p^i, y_p^i)\}_{i=1}^{N_p} \subset D$, $\{(x_{\text{int}}^j, y_{\text{int}}^j, t_j)\}_{j=1}^{N_{\text{int}}} \subset D \times [0, T]$, $\{(x_{\text{bdry}}^k, y_{\text{bdry}}^k, t_k)\}_{k=1}^{N_{\text{bdry}}} \subset \partial D \times \mathcal{T}$, and $N_{\text{tot}} = N_p + N_{\text{int}} + N_{\text{bdry}}$, let us define the loss function as

$$\mathcal{L}[w(\cdot;\omega)] := \frac{1}{N_{\text{tot}}}\Bigg(\sum_{i=1}^{N_p} \underbrace{\mathcal{L}_0[w(x_p^i, y_p^i, 0; \omega)]}_{\text{initial loss}} + \sum_{j=1}^{N_{\text{int}}} \underbrace{\mathcal{L}_p[w(x_{\text{int}}^j, y_{\text{int}}^j, t_j; \omega)]}_{\text{ODE system loss}} + \sum_{k=1}^{N_{\text{bdry}}} \underbrace{\mathcal{L}_{\text{bdry}}[w(x_{\text{bdry}}^k, y_{\text{bdry}}^k, t_k; \omega)]}_{\text{boundary loss}} \Bigg) \tag{4.3}$$

for

$$\mathcal{L}_0[w(x, y, 0; \omega)] := (w(x, y, 0; \omega)[1] - x)^2 + (w(x, y, 0; \omega)[2] - y)^2,$$

and

$$\mathcal{L}_p[w(\cdot;\omega)] := \left(\frac{\partial w(\cdot;\omega)[1]}{\partial t} + \beta w(\cdot;\omega)[2] w(\cdot;\omega)[1]\right)^2 + \left(\frac{\partial w(\cdot;\omega)[2]}{\partial t} - \beta w(\cdot;\omega)[1] w(\cdot)[2] + \gamma w(\cdot;\omega)[2]\right)^2,$$

where $w(\cdot)[1]$ and $w(\cdot)[2]$ denote the first and second components of w.

Lastly, $\mathcal{L}_{\text{bdry}}$ is defined via

$$\mathcal{L}_{\text{bdry}}[w(x, y, kdt; \omega)] := \|w(x, y, kdt; \omega) - g(x, y, kdt)\|_2^2,$$

where the reference vector $g(x, y, kdt)$ is computed as follows:

$$g(x, y, kdt) = (S_k, I_k),$$

where the reference vector $g(x, y, kdt)$ is obtained by numerically solving the uncontrolled dynamics (2.7) using the fourth-order Runge-Kutta method, with the initial conditions $(S_0, I_0) = (x, y)$ and a step size $dt > 0$. The training details are provided in Algorithm 2, and an overview of the framework is illustrated in Figure 3.

4.2.2. Learning the optimal switching time τ

Given u, u^{r_0}, and a neural network $w(x, y, t; \omega)$ that approximates the uncontrolled SIR dynamics satisfying (2.7) with $(S(0), I(0)) = (x, y)$, we minimize the following loss function:

$$\mathcal{L}[\tau(\cdot;\theta)] := \frac{1}{N} \sum_{i=1}^{N} \left\{ \left(u(x_i, y_i) - \tau(x_i, y_i; \theta) - u^{r_0}(S(\tau(x_i, y_i; \theta)), I(\tau(x_i, y_i; \theta))) \right)^2 + \text{pen}(x_i, y_i, \tau(\cdot;\theta)) \right\}, \tag{4.4}$$

over θ for the randomly sampled points $\{(x^i, y^i)\}_{i=1}^N \in D$ where

$$\text{pen}(x, y, \tau) = \begin{cases} \left(\partial_x u^{r_0}(w(x, y, \tau)[1], w(x, y, \tau)[2]) \right)^2 & \text{if} \quad (x, y) \in \mathcal{S}^C, \\ 0 & \text{otherwise.} \end{cases}$$

Here, $w(x, y, \tau)$ approximates the solution $(S(\tau), I(\tau))$ to the uncontrolled SIR with $(S(0), I(0)) = (x, y)$ and is learned from the previous section. The penalty term ensures that τ satisfies (2.10).

5. Experimental results

5.1. Parameter selection and implementation details

In all experiments presented in this paper, the parameters of the controlled SIR model are set as follows: $\beta = 2$, $\gamma = 10$, and $\mu = 0.01$. We optimize using the Adam optimizer [37] with a fixed learning rate of 0.0001. For hardware, we used a single local GPU (RTX 3090) for all experiments.

Various model parameters are explored to investigate the optimal eradication time. Since the focus of this study is to characterize the optimal control, we focus on identifying model parameters that yield non-trivial control, excluding scenarios in which the optimal control is identically one, i.e., $r(t) \equiv 1$.

To train models, we employ uniform sampling over the bounded computational domain. This choice is supported by the semiconcavity of the viscosity solution to the HJB equation [6] and the known structure of the optimal control, which is bang-bang with at most one switching time. Variations in hyperparameters, including batch sizes and learning rates, did not produce significant differences in performance. Notably, the incorporation of skip connections led to improvements in the prediction of u and uncontrolled dynamics, but had a negative effect on τ. Detailed discussions are provided in Appendix A.

5.1.1. Learning u and u^{r_0}

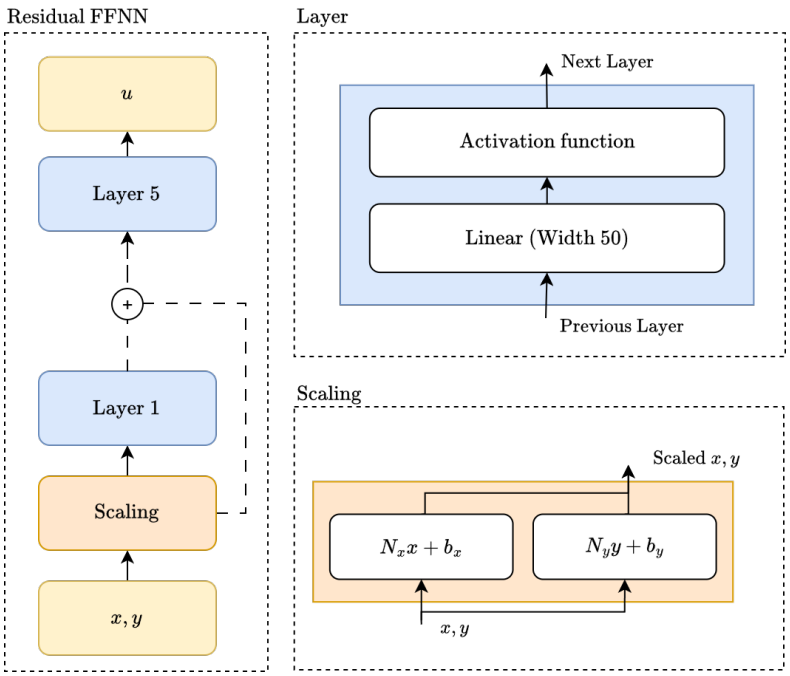


Figure 4. Feed-forward network with residual connections and scaling scheme.

We employ feed-forward neural networks with skip connections where the width is fixed as 50 and the depth consists of five hidden layers unless otherwise stated, as illustrated in Figure 4. Regarding the activation functions, we use the hyperbolic tangent function. Our experimental results indicate that

the variable scaling method remains effective for the hyperbolic tangent activation function.

To test the efficiency of our method for approximating u and u^{r_0}, we consider the domain $[0, 20] \times [0.01, 1]$ by setting $\ell_x = 20$ and $\ell_y + \mu = 1$, while varying the scaling factors N_x and N_y in Section 4.1. During training, we sample 1000 interior collocation points at each iteration.

To define the boundary loss (4.2), we randomly select 100 data points from each boundary D_i defined in (4.1) for $i = 1, 2, 3$, resulting in a total of 300 points, as illustrated in Figure 5. For the reference values obtained by Algorithm 1, we set $dt = d\tau = 0.001$.

Additional experiments validating the robustness of the variable scaling method are presented in Appendix B.

5.1.2. Learning the uncontrolled dynamics

To learn the uncontrolled dynamics, we use the same neural network architecture and activation functions as described above but consider a slightly larger spatio-temporal sampling domain given by $[0, 20] \times [0.01, 2] \times [0, 2.5]$, setting $\ell_x = 20$, $\ell_y + \mu = 2$, and $T = 2.5$ in Section 4.2.1. The value of T is determined heuristically on the basis of the observation that the susceptible and infectious populations approach zero after a sufficiently long duration, which we set to $T = 2.5$.

To define the boundary loss, we set $N_T = 250$, and hence, $dt = 0.01$ in Algorithm 2 and sample 4000 points from $\partial D \times \mathcal{T}$ by setting $N_{\text{bdry}} = 4000$ in (4.3).

For the initial loss, we sample 1000 points per batch from D by setting $N_p = 1000$, shown as green dots in the figure. Additionally, to define the ODE system loss in (4.3), 1000 collocation points $(S(0), I(0), t)$ are sampled per batch uniformly by setting $N_{\text{int}} = 1000$.

5.1.3. Learning the optimal switching time τ

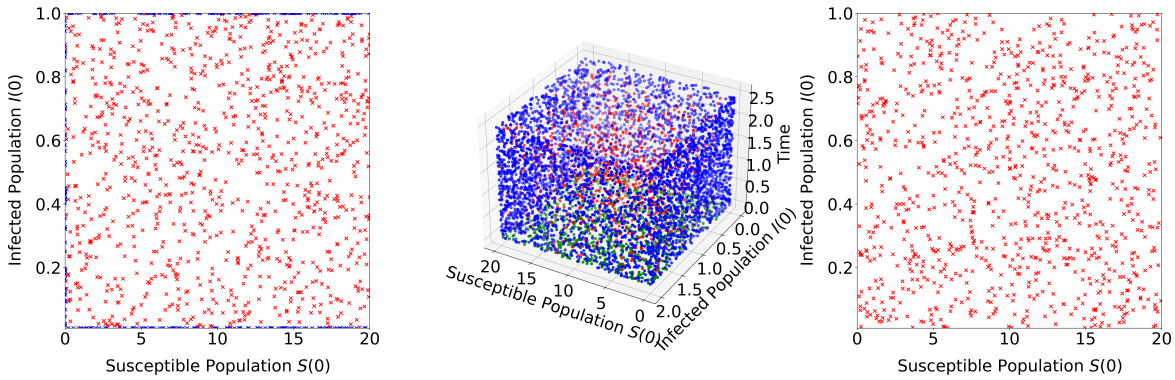


Figure 5. Collocation points and data points used in training u, u^{r_0}, w, and τ. For training u and u^{r_0}, collocation points (red dots) and data points (blue dots) are used (left). For training w, collocation points (red dots), data points (blue dots), and initial condition (green area) are used as shown in the middle. For training τ, only collocation points are used (right).

We use a neural network without skip connections, with a width of 200 and five hidden layers, where leaky ReLU functions are applied in all hidden layers, and the Softplus function is employed in the final layer.

The domain of interest is given by $[0, 20] \times [0.01, 1]$. Recalling the loss function (4.4), we note that the values of u are required in the same domain, while a larger domain must be considered for u^{r_0} and w since $I(\tau(x, y; \theta))$ may exceed 1. Therefore, we approximate u^{r_0} in the domain $[0, 20] \times [0.01, 10]$, using variable scaling factors of $N_x = 1$ and $N_y = 4$.

Similar to the previous experiments, we randomly sample 1,000 collocation points per batch uniformly, as shown in Figure 5 (right), and train using the DPP loss function (4.4).

5.2. Result of learning u and u^{r_0} via PINNs

Table 1. Evaluation of the mean square error (MSE) of variable scaling

N_x	N_y	MSE of u	MSE of u^{r_0}
1	1	3.743×10^{-5}	7.067×10^{-6}
1	20	5.893×10^{-6}	3.883×10^{-6}
1	40	7.992×10^{-6}	5.490×10^{-5}
1	80	4.608×10^{-5}	2.217×10^{-5}
2	1	1.488×10^{-5}	7.130×10^{-6}
2	20	9.979×10^{-6}	7.257×10^{-6}
2	40	8.175×10^{-6}	3.403×10^{-5}
4	80	5.372×10^{-5}	1.917×10^{-5}
5	1	7.730×10^{-6}	1.046×10^{-5}

Table 2. Evaluation of the mean square error (MSE) of different depths in the domain $[0, 20] \times [\mu, 1]$ with $\mu = 0.01$, $N_x = 1$, and $N_y = 20$.

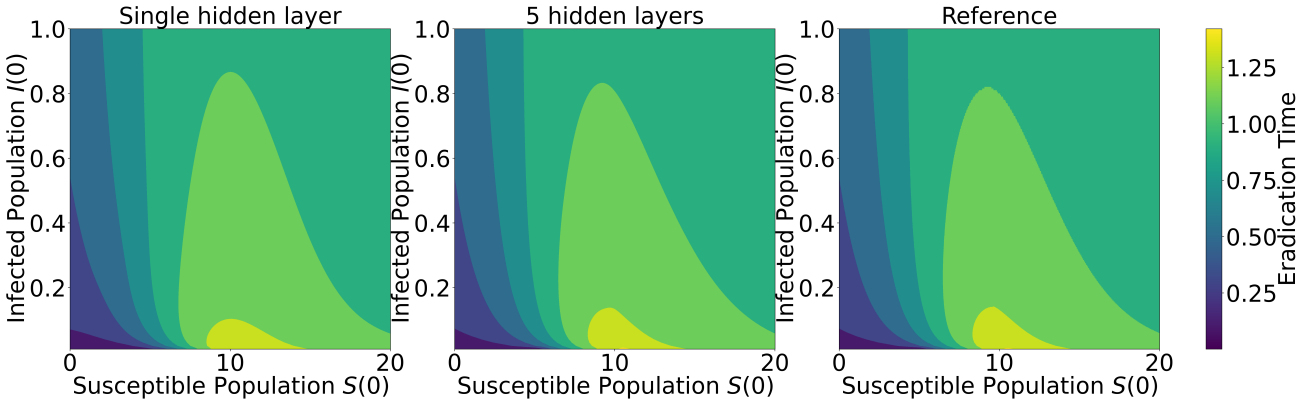
Number of hidden layers	MSE of u	MSE of u^{r_0}
5	5.893×10^{-6}	3.883×10^{-6}
1	3.111×10^{-4}	4.006×10^{-5}

We use the minimum eradication time u obtained from Algorithm 1 as a reference and compare it with the approximate minimum eradication time computed by our PINN algorithm. Similarly, we compute the eradication time under the control u^{r_0} using the Runge-Kutta method with a timestep of $dt = 0.001$. The mean square error (MSE) is then evaluated over the entire set of reference points obtained from Algorithm 1 across the domain.

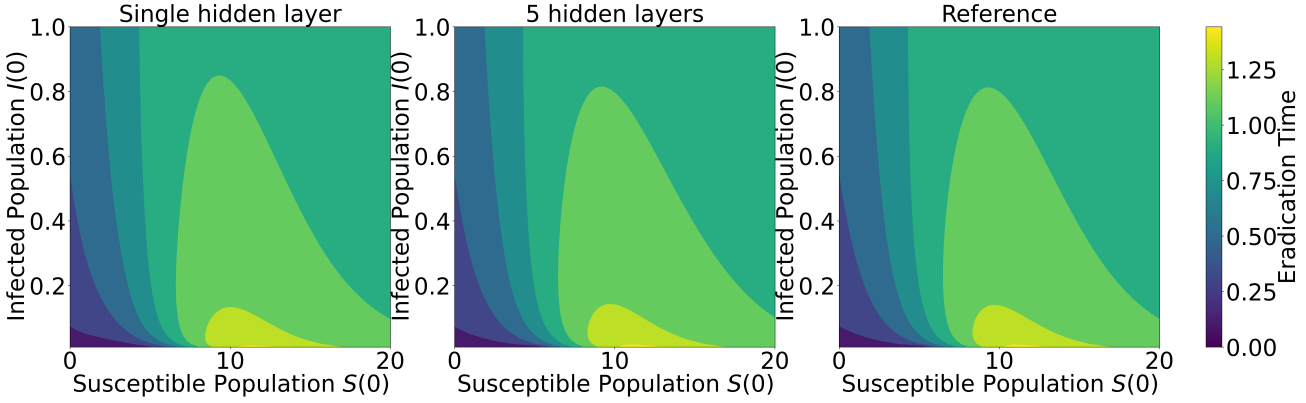
Table 1 summarizes the MSE values corresponding to different choices of the scaling factors N_x and N_y. As observed, applying variable scaling with various scaling parameters N_x and N_y consistently improves the MSE of the base case ($N_x = N_y = 1$), which aligns with the error analysis presented in Section 3.3. Among the many choices of scaling parameters, we empirically found that the result with $N_x = 1, N_y = 20$ results in the lowest MSE, which reduced the MSE of the baseline setting by approximately $\frac{3.743\times10^{-5}-5.893\times10^{-6}}{3.743\times10^{-5}} \times 100\% \simeq 84.3\%$.

We also investigate the effect of network depth in the domain $[0, 20] \times [0.01, 10]$. Table 2 presents the results for different numbers of hidden layers. Training with a model containing five hidden layers yields a lower MSE compared with training with a single hidden layer, demonstrating that deeper networks improve the approximation accuracy. The level set for values of u and u^{r_0} is presented in Figure 6.

Furthermore, we provide a comparison with the finite difference method (FDM) for solving u^{r_0} in Appendix C, where FDM provides inaccurate estimates of the solution, supporting the idea that our method is more efficient.



(a) Values of u learned via the PINN with a different number of hidden layers and $N_x = 1, N_y = 20$: A single hidden layer (left), five hidden layers (middle), and the reference values of u computed by Algorithm 1 (right).



(b) Values of trained u^{r_0} learned via the PINN with a different number of hidden layers and $N_x = 1, N_y = 20$: A single hidden layer (left), five hidden layers (middle), and reference values of u^{r_0} computed by Algorithm 1 (right).

Figure 6. u and u^{r_0} trained with different hidden layer sizes.

5.3. Result of learning uncontrolled SIR dynamics

To evaluate the trained uncontrolled SIR dynamics w, we use the Runge-Kutta method. For reference data, we first sample $\{(x_i, y_i)\}_{i=1}^{1000} \subset [0, 20] \times [0.01, 1]$ and approximate $(S(t_k), I(t_k))$, satisfying the uncontrolled dynamics (2.7) with the initial conditions $(S(0), I(0)) = (x_i, y_i)$ for all $i = 1, \ldots, 1000$ and $t_k \in \{0.025k \mid 0 \le k \le 100\}$, using the Runge-Kutta method.

The MSE of w in estimating the population dynamics is 7.557×10^{-3}, indicating that the neural network effectively estimates the uncontrolled dynamics. Figure 7 illustrates a comparison between the trajectories approximated by the trained neural network and those obtained via the Runge-Kutta method.

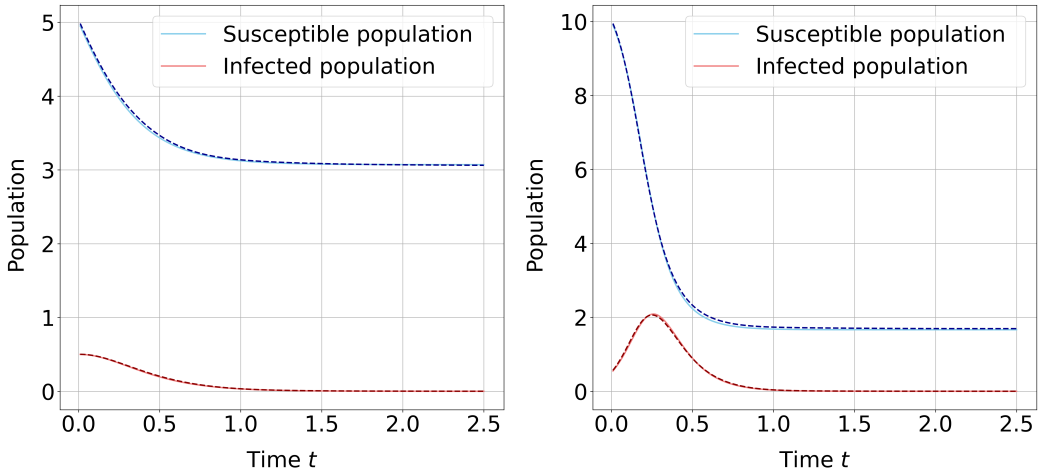


Figure 7. Comparison between $w(S(0), I(0), t; \omega)$ and the ground truth computed via the Runge-Kutta method for selected initial conditions $(S(0), I(0))$ that are not part of the training set or the boundary conditions. The left shows the trajectories for $S(0) = 5.0, I(0) = 0.5$, while the right panel corresponds to $S(0) = 10.0, I(0) = 0.5$. The dashed lines represent the results obtained using the Runge-Kutta method, whereas the solid lines indicate the estimations from the trained neural network.

5.4. Result of learning the optimal switching time

Recalling (2.11) and (2.10), the region where the optimal switching time is nonzero is characterized by the inequality $\partial_x u(x, y) \leq 0$, as demonstrated in Figure 8.

To evaluate the accuracy of the estimated switching time, we compute the MSE of the estimated switching time τ using the samples $\{(x_i, y_i)\}_{i=1}^{1000} \subset \mathcal{S}^C$.

The optimal switching time for the initial conditions $(S(0), I(0)) : \{(0.01i, 0.01j) \mid 0 \leq i \leq 2000, 0 \leq j \leq 100\}$ is approximated using the reference values obtained from Algorithm 1, with a time discretization of $dt = 0.001$ and $d\tau = 0.001$.

Furthermore, we assess the performance of our model using different numbers of hidden layers. When using a single hidden layer, the model achieves an MSE of 6.662×10^{-4}. However, increasing the depth to five hidden layers results in a slightly higher MSE of 6.667×10^{-4}, indicating that additional depth does not necessarily improve accuracy in this setting.

Figure 9 confirms that our trained model achieves accurate estimation of τ with minimal error.

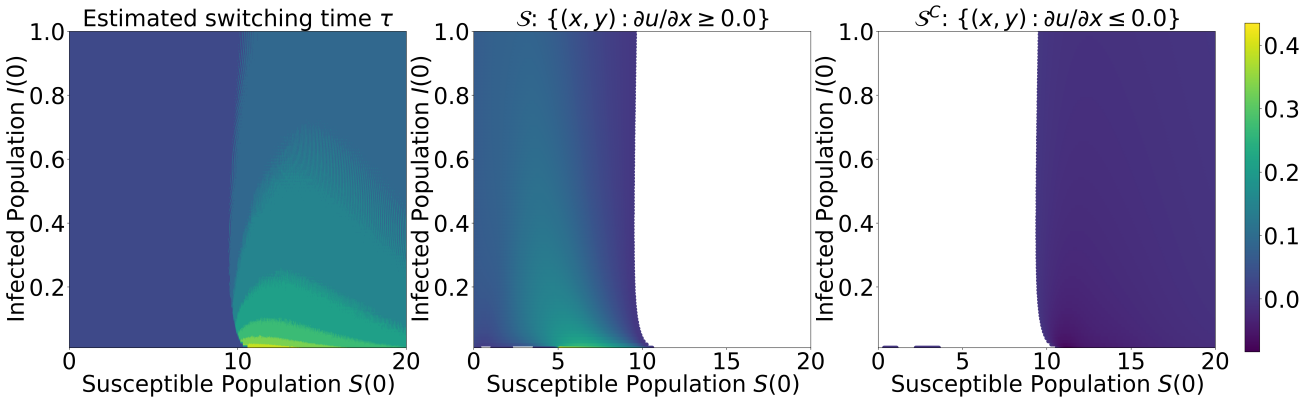


Figure 8. Heatmap of the region $\mathcal{S}$ and its complement. The figure illustrates the ground truth of the optimal switching time τ (left), the region $\mathcal{S} = \{(x, y) : \partial_x u(x, y) \geq 0\}$ (middle), and the complementary region $\mathcal{S}^C = \{(x, y) : \partial_x u(x, y) \leq 0\}$ (right).

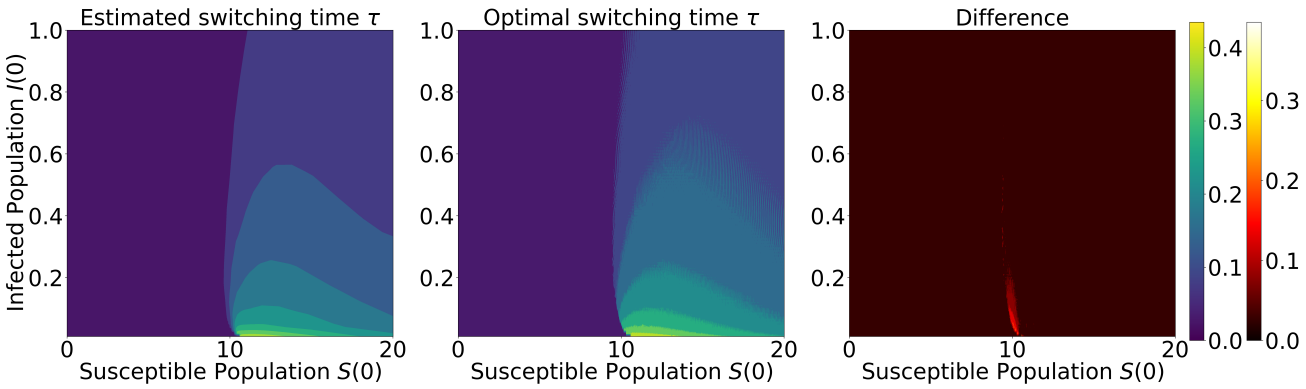


Figure 9. Results of the trained switching time τ. The figure presents the estimated switching time $\tau(\cdot;\theta)$ (left), the ground truth τ (middle), and the difference between the ground truth and the estimated values, $\tau - \tau(\cdot;\theta)$ (right).

6. Conclusions

We proposed a novel approach for approximating the minimum eradication time and synthesizing optimal vaccination strategies for a controlled SIR model via variable scaling PINNs and the dynamic programming principle. A salient feature of our method is that it does not require domain discretization and offers a computationally efficient solution to HJB equations related to the minimum eradication time. The experimental results confirmed the accuracy and robustness of our method in approximating the eradication time and deriving optimal vaccination strategies. Furthermore, the theoretical justification for our approach is established on the basis of NTK theory.

Despite these contributions, our study has several limitations. The current framework assumes a

time-homogeneous controlled SIR model with constant infection and recovery rates, which might limit its applicability to real-world scenarios with time-inhomogeneous dynamics or heterogeneous populations. Additionally, while our method improves computational efficiency, the training of physics-informed neural networks remains sensitive to the hyperparameter choices and boundary conditions, which require further investigation.

Future work will focus on extending this framework to time-inhomogeneous SIR models, for which theoretical support has been proposed in [9]. In addition, exploring systems with state-dependent infection rates $\beta = \beta(S, I)$, as considered in nonlinear incidence models [5], is also a promising direction. Another important extension is to adapt the framework to impose the minimal, non-augmented boundary conditions that ensure the well-posedness of the HJB equation. These directions would broaden the applicability of the proposed approach and address key theoretical and practical challenges in controlled SIR models.

Use of AI tools declaration

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

Acknowledgments

This research was supported by Seoul National University of Science and Technology.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. W. O. Kermack, A. G. McKendrick, A contribution to the mathematical theory of epidemics, *Math. Phys. Eng. Sci.*, **115** (1927), 700–721. https://doi.org/10.1098/rspa.1927.0118
2. M. Barro, A. Guiro, D. Quedraogo, Optimal control of a SIR epidemic model with general incidence function and a time delays, *CUBO A Math. J.*, **20** (2018), 53–66. https://doi.org/10.4067/S0719-06462018000200053
3. P. A. Bliman, M. Duprez, Y. Privat, N. Vauchelet, Optimal immunity control and final size minimization by social distancing for the SIR epidemic model, *J. Optim. Theory Appl.*, **189** (2021), 408–436. https://doi.org/10.1007/s10957-021-01830-1
4. L. Bolzoni, E. Bonacini, C. Soresina, M. Groppi, Time-optimal control strategies in SIR epidemic models, *Math. Biosci.*, **292** (2017), 86–96. https://doi.org/10.1016/j.mbs.2017.07.011
5. E. V. Grigorieva, E. N. Khailov, A. Korobeinikov, Optimal control for a SIR epidemic model with nonlinear incidence rate, *Math. Model. Nat. Phenom.*, **11** (2016), 89–104. https://doi.org/10.1051/mmnp/201611407
6. R. Hynd, D. Ikpe, T. Pendleton, An eradication time problem for the SIR model, *J. Differ. Equations*, **303** (2021), 214–252. https://doi.org/10.1016/j.jde.2021.09.001

7. R. Hynd, D. Ikpe, T. Pendleton, Two critical times for the SIR model, *J. Math. Anal. Appl.*, **505** (2022), 125507. https://doi.org/10.1016/j.jmaa.2021.125507
8. L. S. Pontryagin, *Mathematical Theory of Optimal Processes*, Routledge, 2018.
9. J. Jang, Y. Kim, On a minimum eradication time for the SIR model with time-dependent coefficients, preprint, arXiv:2311.14657. https://doi.org/10.48550/arXiv.2311.14657
10. H. V. Tran, *Hamilton-Jacobi equations: Theory and applications*, *Am. Math. Soc.*, **213** (2021), 322.
11. S. Ko, S. Park, VS-PINN: A fast and efficient training of physics-informed neural networks using variable-scaling methods for solving PDEs with stiff behavior, *J. Comput. Phys.*, **529** (2025), 113860. https://doi.org/10.1016/j.jcp.2025.113860
12. A. Jacot, F. Gabriel, C. Hongler, Neural tangent kernel: Convergence and generalization in neural networks, in *STOC 2021: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, (2021). https://doi.org/10.1145/3406325.3465355
13. 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. https://doi.org/10.1016/j.jcp.2018.10.045
14. E. Kharazmi, M. Cai, X. Zheng, G. Lin, G. E. Karniadakis, Identifiability and predictability of integer-and fractional-order epidemiological models using physics-informed neural networks, *Nat. Comput. Sci.*, **1** (2021), 744–753. https://doi.org/10.1038/s43588-021-00158-0
15. A. Yazdani, L. Lu, M. Raissi, G. E. Karniadakis, Systems biology informed deep learning for inferring parameters and hidden dynamics, *PLoS Comput. Biol.*, **16** (2020), e1007575. https://doi.org/10.1371/journal.pcbi.1007575
16. S. Cai, Z. Mao, Z. Wang, M. Yin, G. E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: A review, *Acta Mech. Sin.*, **37** (2021), 1727–1738. https://doi.org/10.1007/s10409-021-01148-1
17. M. Raissi, A. Yazdani, G. E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, *Science*, **367** (2020), 1026–1030. https://doi.org/10.1126/science.aaw4741
18. X. Jin, S. Cai, H. Li, G. E. Karniadakis, NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations, *J. Comput. Phys.*, **426** (2021), 109951. https://doi.org/10.1016/j.jcp.2020.109951
19. X. Wang, J. Li, J. Li, A deep learning based numerical PDE method for option pricing, *Comput. Econ.*, **62** (2023), 149–164. https://doi.org/10.1007/s10614-022-10279-x
20. Y. Bai, T. Chaolu, S. Bilige, The application of improved physics-informed neural network (IPINN) method in finance, *Nonlinear Dyn.*, **107** (2022), 3655–3667. https://doi.org/10.1007/s11071-021-07146-z
21. G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4d flow MRI data using physics-informed neural networks, *Comput. Methods Appl. Mech. Eng.*, **358** (2020), 112623. https://doi.org/10.1016/j.cma.2019.112623

22. F. Sahli C., Y. Yang, P. Perdikaris, D. E. Hurtado, E. Kuhl, Physics-informed neural networks for cardiac activation mapping, *Front. Phys.*, **8** (2020), 42. https://doi.org/10.3389/fphy.2020.00042

23. S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, *SIAM J. Sci. Comput.*, **43** (2021), A3055–A3081. https://doi.org/10.1137/20M1318043

24. Y. Liu, L. Cai, Y. Chen, B. Wang, Physics-informed neural networks based on adaptive weighted loss functions for Hamilton–Jacobi equations, *Math. Biosci. Eng.*, **19** (2022), 12866–12896. https://doi.org/10.3934/mbe.2022601

25. S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, *J. Comput. Phys.*, **449** (2022), 110768. https://doi.org/10.1016/j.jcp.2021.110768

26. W. Ji, W. Qiu, Z. Shi, S. Pan, S. Deng, Stiff-pinn: Physics-informed neural network for stiff chemical kinetics, *J. Phys. Chem. A*, **125** (2021), 8098–8106. https://doi.org/10.1021/acs.jpca.1c05102

27. L. D. McClenny, U. M. Braga-Neto, Self-adaptive physics-informed neural networks, *J. Comput. Phys.*, **474** (2023), 111722. https://doi.org/10.1016/j.jcp.2022.111722

28. S. Mowlavi, S. Nabi, Optimal control of PDEs using physics-informed neural networks, *J. Comput. Phys.*, **473** (2023), 111731. https://doi.org/10.1016/j.jcp.2022.111731

29. Y. Meng, R. Zhou, A. Mukherjee, M. Fitzsimmons, C. Song, J. Liu, Physics-informed neural network policy iteration: Algorithms, convergence, and verification, preprint, arXiv:2402.10119. https://doi.org/10.48550/arXiv.2402.10119

30. J. Y. Lee, Y. Kim, Hamilton-Jacobi based policy-iteration via deep operator learning, preprint, arXiv:2406.10920. https://doi.org/10.48550/arXiv.2406.10920

31. W. Tang, H. Tran, Y. Zhang, Policy iteration for the deterministic control problems—A viscosity approach, *SIAM J. Control Optim.*, **63** (2025), 375–401. https://doi.org/10.1137/24M1631602

32. S. Yin, J. Wu, P. Song, Optimal control by deep learning techniques and its applications on epidemic models, *J. Math. Biol.*, **86** (2023), 36. https://doi.org/10.1007/s00285-023-01873-0

33. Y. Ye, A. Pandey, C. Bawden, D. Sumsuzzman, R. Rajput, A. Shoukat, et al., Integrating artificial intelligence with mechanistic epidemiological modeling: A scoping review of opportunities and challenges, *Nat. Commun.*, **16** (2025), 581. https://doi.org/10.1038/s41467-024-55461-x

34. Z. Yang, Z. Zeng, K. Wang, S. Wong, W. Liang, M. Zanin, et al., Modified SEIR and AI prediction of the epidemics trend of COVID-19 in china under public health interventions, *J. Thorac. Dis.*, **12** (2020), 165. https://doi.org/10.21037/jtd.2020.02.64

35. L. Evans, Partial differential equations: 2nd edition, in *Graduate Studies in Mathematics*, *Am. Math. Soc.*, **19** (2010), 749.

36. R. V. Hogg, J. W. McKean, A. T. Craig, *Introduction to Mathematical Statistics*, 8th edition, 2013.

37. D. Kingma, Adam: A method for stochastic optimization, preprint, arXiv:1412.6980.

Appendix

A. *Choice of timestep size and the number of sampling points*

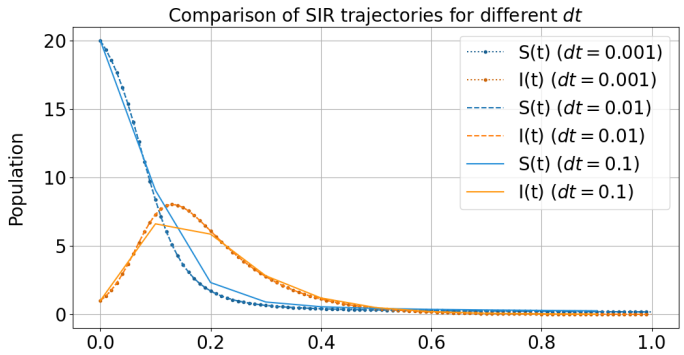


Figure A1. SIR trajectory simulation with various timestep sizes dt.

The distribution of collocation points significantly affects PINNs' performance. We tested both uniform and exponential sampling strategies, and observed that both are effective under our experimental setup.

For the training of the value functions u and u^{r_0}, we use 1000 collocation points per batch, together with 300 fixed boundary points to ensure sufficient coverage near the domain boundaries. To train a neural network $w(x, y, t; \omega)$ that approximates the uncontrolled SIR dynamics, we use 1000 collocation and initial points per batch, along with 4000 fixed boundary points to accurately capture the evolution under the uncontrolled dynamics. For the training of the switching time function $\tau(x, y; \theta)$, we use 1000 collocation points per batch sampled from the $\mathcal{S}^C$.

These choices provide a balance between numerical accuracy and computational efficiency, and are consistent with the stability and convergence behavior reported in the main paper.

We also note that the choice of timestep size in learning the uncontrolled SIR model is critical for obtaining accurate numerical solutions. To determine appropriate values, we first conduct preliminary experiments by plotting the SIR trajectories. We generate (S, I) trajectories with different timestep sizes and confirm that using $dt = 0.01$ and $dt = 0.001$ leads to consistent results, as seen in Figure A1. On the other hand, we choose a smaller timestep $dt = d\tau = 0.001$ when approximating the minimum eradication time and optimal switching time for better accuracy.

B. *Variable scaling under different parameter settings*

To further investigate the robustness of the variable scaling method, we conducted additional experiments under various parameter settings. Specifically, we examined how changes in the infection rate β, the recovery rate γ, and the domain scaling factors affect the error.

As seen in Table A1, the mean squared residual errors for different scaling parameters (N_x, N_y) across several combinations of (β, γ) and domain widths (ℓ_x, ℓ_y). In all cases, the appropriate choice of variable scaling leads to a significant reduction in error, which confirms the robustness and generalizability of the proposed scaling scheme.

Table A1. Evaluation of the mean square error (MSE) of variable scaling on different experiment parameters with the threshold $\mu = 0.01$.

β	γ	x	y	N_x	N_y	MSE of u
2	4	[0, 5]	[μ, 5]	1	1	7.488×10^{-5}
2	4	[0, 5]	[μ, 5]	2	2	4.350×10^{-5}
2	4	[0, 15]	[μ, 8]	1	1	4.049×10^{-5}
2	4	[0, 15]	[μ, 8]	1	2	3.923×10^{-5}
2	2	[0, 5]	[μ, 10]	1	1	2.983×10^{-5}
2	2	[0, 5]	[μ, 10]	1	2	2.694×10^{-5}
2	2	[0, 5]	[μ, 10]	1	8	2.407×10^{-5}

C. Comparison with the finite difference method

To compare our method with the standard finite difference method (FDM), we use the same experimental parameters ($\beta = 2$ and $\gamma = 10$) as in Figure 6(b) and consider the following equation:

$$\beta xy\partial_x u^{r_0} + x\partial_x u^{r_0} + (\gamma - \beta x)y\partial_y u^{r_0} = 1 \quad \text{in} \quad [0, \ell_x] \times [\mu, \mu + \ell_y], \tag{6.1}$$

with boundary conditions

$$\begin{aligned} u^{r_0}(0, y) &= \tfrac{1}{\gamma} \ln\left(\tfrac{y}{\mu}\right) \quad \text{for} \quad \mu \le y \le \mu + \ell_y, \\ u^{r_0}(x, \mu) &= 0 \quad \text{for} \quad 0 \le x \le \frac{\gamma}{\beta}, \\ u^{r_0}(x, \mu) &= b_2(x) \quad \text{for} \quad \frac{\gamma}{\beta} \le x \le \ell_x, \\ u^{r_0}(x, \mu + \ell_y) &= b_1(x) \quad \text{for} \quad 0 \le x \le \ell_x, \end{aligned}$$

where the boundary functions $b_1(x)$ and $b_2(x)$ are obatined via Algorithm 1.

To implement the standard finite difference scheme, we discretize the domain $[0, \ell_x] \times [\mu, \mu + \ell_y]$ using $\Delta x = \Delta y = 0.01$, and obtain the discretized domain

$$\mathcal{G} := \{(x_i = i\Delta x, y_j = \mu + j\Delta y) : 0 \le i \le \ell_x/\Delta x, 0 \le j \le \ell_y/\Delta y\}.$$

For all $(x_i, y_j) \in \mathcal{G}$, letting

$$U_{i,j} \approx u^{r_0}(x_i, y_j),$$

the first-order scheme is given by

$$(\beta\, x_i\, y_j + x_i)\, \frac{U_{j,i} - U_{j,i-1}}{\Delta x} \;+\; (\gamma - \beta\, x_i)\, y_j\, \frac{U_{j,i} - U_{j-1,i}}{\Delta y} \;=\; 1,$$

where

$$\partial_x u^{r_0}(x_i, y_j) \;\approx\; \frac{U_{j,i} - U_{j,i-1}}{\Delta x}, \quad \partial_y u^{r_0}(x_i, y_j) \;\approx\; \frac{U_{j,i} - U_{j-1,i}}{\Delta y}.$$

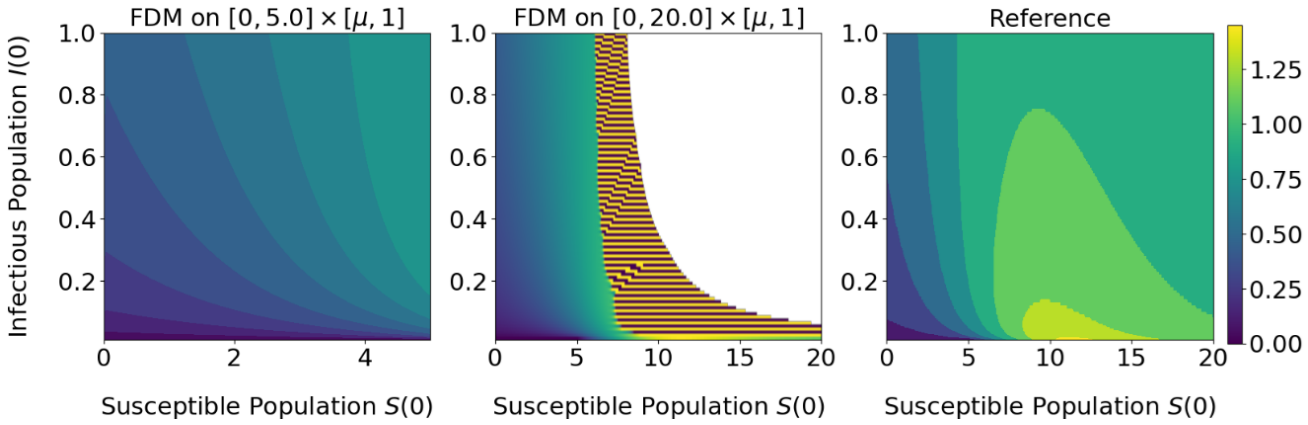


Figure A2. Minimum eradication time computed via the finite difference method over $[0, 5] \times [\mu, 1]$ (left), $[0, 20] \times [\mu, 1]$ (middle), and the reference domain (right), with $\mu = 0.01$.

However, as shown in Figure A2, while the finite difference method provides accurate estimates of the minimum eradication time in the region $S(0) \leq \frac{\gamma}{\beta} = 5$, it fails to yield stable results beyond this threshold. In particular, for $S(0) > 5$, where the value function exhibits steep gradients and rapid transitions (see Figure A2), the FDM scheme produces numerical instabilities and diverging solutions.

In contrast, the PINN approach demonstrates superior robustness and stability, successfully capturing the solution structure even in such challenging regions. This comparison highlights the advantages of the proposed PINN framework, especially in handling domains with sharp transitions or lacking reliable stability theory for traditional schemes.