Solving Tuberculosis Model with Differential Infectivity using Numerical Methods and PINNs

Zyad Abdelfattah Rowida Mohamed Malak Elhamshary Youssef Kotb Ali Gad Youmna Sabry Youssef Samy

Abdulrahman Emad Ward Elsalkani Yousf Mortada Muhammed Nasser Farah Yehya

***Abstract*—Tuberculosis (TB) continues to be a serious global health issue, and building accurate models to understand and predict its spread is critical for controlling outbreaks. This work utilizes a system of nonlinear ordinary differential equations (ODEs) to simulate TB transmission and compares four numerical methods for solving these equations: Heun's method, the classical fourth-order Runge–Kutta (RK4), the Adaptive Runge–Kutta method, and Physics-Informed Neural Networks (PINNs). Each method is evaluated in terms of accuracy and computational efficiency, using mean squared error (MSE) and execution time as primary metrics. The results indicate that RK4 provides the highest accuracy, while the adaptive method achieves faster performance with acceptable precision. Although PINNs require longer training times, they demonstrate strong generalization capabilities beyond the original data range and perform well under limited data conditions. These findings highlight the trade-offs among speed, accuracy, and flexibility in solving ODEs for disease modeling, and suggest that PINNs may offer valuable advantages in real-world, data-driven scenarios where conventional methods are less effective.**

I. Introduction

Tuberculosis (TB) remains one of the most common and deadly infectious diseases worldwide. Each year, millions of new cases and deaths are reported, particularly in developing countries. Caused by *Mycobacterium tuberculosis*, TB primarily affects the lungs but can also spread to other organs. Despite ongoing public health efforts and treatment strategies, its transmission continues to pose a significant challenge, especially in the presence of HIV co-infection and antibiotic resistance.

To understand and predict the dynamics of TB spread, mathematical modeling is frequently employed. These models help simulate infection patterns, evaluate the effectiveness of interventions, and guide public health decision-making. A commonly used approach is the SEIR-based compartmental framework, which divides the population into Susceptible (S), Exposed (E), and Infectious (I). In this work, an extended model is considered by incorporating a fourth compartment (L), representing loss of sight. This yields a nonlinear system of four ordinary differential equations (ODEs) as proposed in [2].

Due to the complexity of such models, accurately and efficiently solving the resulting ODEs is essential. Classical numerical methods such as Heun's method and the fourth-order Runge–Kutta (RK4) have long served as reliable tools for this purpose. More advanced techniques, such as the Adaptive Runge–Kutta method (RK45), improve computational efficiency by dynamically adjusting the step size during simulation. In parallel, data-driven approaches like Physics-Informed Neural Networks (PINNs) have emerged as promising tools, especially when data is sparse or noisy.

This paper implements and compares four numerical techniques—Heun's method, RK4, Adaptive RK, and PINNs—for solving the TB transmission model. Each method is evaluated based on accuracy, using Mean Squared Error (MSE), and execution time. The comparison highlights the trade-offs in performance and suggests directions for applying these methods in real-world epidemiological modeling.

II. Literature Review

In the work of Kanwal et al. [3], the authors proposed a TB model comprising five compartments: Susceptible (S), Latent (L), Infected (I), Under Treatment (T), and Recovered (R). The model incorporates nonlinear transitions driven by bilinear infection terms such as SI, LI, and IT, and features feedback mechanisms that couple compartments tightly. While the system is mathematically well-posed, its complex nonlinear structure presents challenges when solved using classical fixed-step methods. Although the exact step sizes used by Kanwal et al. are not stated, their results suggest that numerical instability occurs with relatively large steps, requiring finer resolution to ensure convergence. The authors report that the 4th order Runge Kutta (RK4) method may yield inaccurate or even unstable numerical results unless very small time steps are used. These instabilities are due to the method's inability to adapt to the stiffness and fast transients intrinsic to the model's dynamics.

E. D. Tadesse, M. A. Bhat, and A. Ayele, in their 2024 paper titled "A deterministic compartment model for analyzing tuberculosis dynamics considering vaccination and reinfection [4], introduced an extended ODE-based TB model that accounts for two important real-world factors: vaccination and reinfection. The model divides the population into several compartments, including vaccinated individuals and those who may become reinfected after recovering. This approach allows the simulation of TB behavior in populations where immunity may wane or where treatment is incomplete or inconsistent. While the structure is comprehensive, the authors focus on theoretical equilibrium behavior and basic simulation rather than applying specific numerical solution

methods. This limits the paper's applicability to this project, which emphasizes solving ODEs using multiple numerical techniques (such as Runge–Kutta–Fehlberg, Heun, and adaptive methods). Additionally, while the model is insightful in its handling of reinfection dynamics, it does not include the concept of differential infectivity—central to our selected model—and does not simulate the dynamics of "loss of sight" patients, which is a key feature of the system this paper works with. The simulations use fixed parameter values without exploring solver performance, convergence, or sensitivity, making it less suitable for evaluating numerical accuracy and efficiency—core goals in our study. For these reasons, while the paper provides valuable background and ideas for future model enhancements, it was not used as the primary basis for this paper's solutions of the ODE system. Instead, it is considered a relevant but supplementary reference for understanding extended TB dynamics.

In their work, Syafruddin et al. presented a numerical approach to modeling the transmission of tuberculosis using a classical SIR (Susceptible–Infectious–Recovered) framework [5]. The authors applied the fourth-order Runge–Kutta method (RK4) to solve the system of ordinary differential equations and validate the model using real TB incidence data from Makassar, Indonesia. Despite the model being relatively simple in structure — with no latent or reinfection compartments — its strength lies in demonstrating how a robust numerical method like RK4 can be effectively used to simulate disease dynamics with high accuracy and stability. The results show that RK4 captures the spread of TB over time and aligns closely with reported data, reinforcing the method's reliability. This approach is utilized as one of the main numerical techniques that will be implemented to solve equations [1-4].

Recently, a technique to solve systems of differential equations using neural networks, calling Physics Informed Neural Networks (PINNs) has emerged. In their paper, M. Raissi, P. Perdikaris, and G. Karinakaris [6] proposed implementing a neural network that encodes the physical nature of any system of differential equations through its loss function and, given sufficient data points and a proper architecture, can estimate accurately the solution of that system. Their approach will be among the ones implemented in this paper to solve the differential equation system mentioned in the introduction. The notable advantage of this technique is its ability to solve notoriously difficult differential equations without having to rely on a small step size or losing any accuracy. As shown in [6], the paper employed an implicit Rungue-Kutta scheme for Burger's equation with 500 stages and the PINN was able to approximate that scheme in a single step with an error of $8.2 * 10^{-4}$, the lowest error ever reported for this problem and 2 orders in magnitude smaller than the previous lowest error value. Due to how effective this technique is, several papers have employed this technique to solve other differential equations. A notable paper is the master's thesis of A. Johannessen [7], which outlined the architectures that succeeded for solving various types of differential equations using PINNs, like linear PDEs, non-linear PDEs, and non-linear ODEs – our interest in this paper. The architectures proposed by both [6] and [7] were the basis for the architecture and parameters selected for the proposed PINN model in this paper. In their paper, Pal et al. developed a Physics-Informed Neural Network (PINN) model to simulate and predict tuberculosis dynamics, particularly in diabetic patients a population with elevated TB risk due to immunosuppression [8]. The model formulates the problem using ODEs and embeds them within the neural network's training loss, ensuring that solutions conform to biological laws despite noisy or scarce data. Implemented using the DeepXDE library, the network simultaneously estimated latent variables such as the susceptible S , infected I , and treated T populations over time. By combining data loss and physics loss, An Adam optimizer with a learning rate of 0.001 is employed to minimize total loss , and the network parameters are initialized using a method such as the Glorot normal initializer. By minimizing the combined loss the PINN framework produced reliable parameter estimates and temporal forecasts. This approach highlighted the advantage of incorporating domain knowledge into neural architectures, especially for epidemiological modeling with limited observations.

In their paper, Lu et al. introduced DeepXDE, a Python library designed to solve differential equations using deep learning [10]. Their framework supports forward and inverse problems in ODEs, PDEs, and fractional systems. In the context of disease modeling, DeepXDE facilitates the implementation of PINNs by providing automatic differentiation, compact code structure, and flexible architecture design. Though not focused on tuberculosis, the library's capabilities made it central to building the model used by Pal et al., reinforcing its role as a powerful tool for data-driven differential equation modeling.

III. ODE System Description

To simulate the spread of tuberculosis (TB), we adopt a compartmental SEIL model that divides the population into four groups: Susceptible (S), Exposed (E), Infectious (I), and Latent (L). The model is governed by a system of nonlinear ordinary differential equations (ODEs), each describing the rate of change for one compartment over time.

Let $Y(t) = [S(t), E(t), I(t), L(t)]$. The dynamics of the model are given by:

$$\frac{dS}{dt} = \Lambda - \beta S(I + \delta L) - \mu S \tag{1}$$

$$\frac{dE}{dt} = (1-p)\beta S(I + \delta L) + r_2 I - (\mu + k(1-r_1))E \tag{2}$$

$$\frac{dI}{dt} = p\beta S(I + \delta L) + k(1-r_1)E + \gamma L - (\mu + d_1 + \phi(1-r_2) + r_2)I \tag{3}$$

$$\frac{dL}{dt} = \phi(1-r_2)I - (\mu + d_2 + \gamma)L \tag{4}$$

Each term corresponds to a biological or epidemiological process:

$\Lambda = 2$ (year^{-1}): recruitment rate into the susceptible population
$\beta = 0.025$: transmission coefficient
$\delta = 1$: adjusts the infectivity of latent individuals
$p = 0.3$: fraction of individuals who progress directly to the infectious stage
$\mu = 0.0101$: natural death rate
$k = 0.005$: progression rate from exposed to infectious
$r_1 = 0$, $r_2 = 0.8182$: early treatment and successful recovery rates
$\phi = 0.02$: rate of transition from infectious to latent
$\gamma = 0.01$: reactivation rate from latent to infectious
$d_1 = 0.0227$, $d_2 = 0.20$: death rates for infectious and latent individuals

The model is simulated over a 20-year period using the following initial conditions:

$$S(0) = \frac{\Lambda}{\mu} \approx 198.02, E(0) = 1.0, I(0) = 0.0, L(0) = 0.0$$

These values represent an endemic scenario where one exposed individual is introduced into an otherwise steady-state population. The SEIL framework allows us to capture both direct transmission and complex behaviors like relapse, treatment, and reactivation, making it well-suited for evaluating numerical methods applied to real-world epidemic models.

IV. Methodology

This paper explores several methods to solve the system represented by equations 1-4. The first method is the Heun method, a method known for its simplicity and precision over Euler's method. The second method is the 4th order classical Runge-Kutta method. This method is widely known for its precision in solving ODE systems as well as its relative speed. The third method is adaptive runge-kutta method, which is a robust method combining both speed and precision on demand. Finally, the paper implements the first-ever use of Physics Informed Neural Networks (PINNs) to solve the aforementioned coupled ODE system. The implementation for each method was done in Python. All methods were estimated from $t = 0$ to $t = 20$ except for PINNs, which was computed on that interval as well as the interval from $t = 0$ to $t = 100$ to test its extrapolation accuracy. Each method's solutions were plotted on the same graph as the solutions of [1] using matplotlib. The error of each method was computed as the mean square error between the solutions of [1] for each compartment and the method's solutions. Each method was timed using Python's timeit library for 1000 repeats with one execution per repeat. The average of those times was calculated and labeled as the average execution time of the said method.

A. Heun method

Heun's method relies on improving euler's method. First, the initial prediction from euler's method is obtained (taking the S compartment as an example:

$$\tilde{S}_{n+1} = S_n + h \cdot f_S(E_n, I_n, L_n) \tag{5}$$

This prediction is used to estimate the slope at the point S_{n+1}. Then the average slope between the slope at the next point and the slope at the current point is calculated. This new slope is then used to estimate the true next value:

$$S_{n+1} = S_n + \frac{h}{2}\left[f_S(E_n, I_n, L_n) + f_S(E_{n+1}, I_{n+1}, L_{n+1})\right] \tag{6}$$

This method was implemented in python with the inputs being the step size h and a list of the initial values. The solution was calculated at $h = 0.5$ and the solutions, mean square error, and the elapsed time were recorded.

B. Classical 4th order Runge Kutta Method

The Runge-Kutta methods are widely used numerical techniques for approximating solutions to ordinary differential equations (ODEs), especially when analytical solutions are difficult or impossible to obtain. These methods are essential in scientific disciplines such as physics, biology, and engineering, where systems often evolve over time based on complex interactions.

A first-order ODE can generally be written as:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0$$

where the rate of change of the quantity y is given as a function of time t and the value of y itself. The goal is to compute the approximate value of y at a series of time points.

The 4th Order Runge-Kutta (RK4) method is one of the most commonly used algorithms for numerical integration of ODEs. It achieves a high level of accuracy by estimating the solution at each time step using a weighted average of four slope calculations.

To estimate y_{n+1} from y_n, the RK4 method computes [2]:

$$\begin{aligned}
k_1 &= f(t_n, y_n) \quad \text{(slope at the beginning)} \\
k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \\
k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\
k_4 &= f(t_n + h, y_n + hk_3)
\end{aligned}$$

Then the next value is given by [2]:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

This approach significantly improves the accuracy of the solution without a large computational burden. The solution was calculated at $h = 0.5$. In many real-world models, such as infectious disease dynamics, the system consists of multiple coupled differential equations. For instance, in the tuberculosis (TB) model, we consider four state variables in our equation:

$$\mathbf{y}(t) = [S(t), E(t), I(t), L(t)]$$

representing susceptible, exposed, infectious, and latent populations, respectively.

Each variable has its own rate of change:

$$\left[\frac{dS}{dt}, \frac{dE}{dt}, \frac{dI}{dt}, \frac{dL}{dt}\right] = f(\mathbf{y}, \text{parameters})$$

The RK4 method can be directly extended to handle such systems by treating the state variables as vectors and performing the operations element-wise. All k_1, k_2, k_3, k_4 are computed as vectors, and the update rule applies identically.

C. Adaptive Runge Kutta Method

Adaptive Quadrature(AQ) is a technique for numerical integration that is characterized by dynamic step size. The step size is adjusted so that small intervals are used in regions of rapid variations and larger intervals are used where the function changes gradually. The chosen method is applied at two levels of refinement and the difference between these two levels is used to estimate the truncation error. If the truncation error is less than the set tolerance, no further refinement is required, and the integral estimate is acceptable. If the error is too large, the step size is refined and the process repeated until the error falls within acceptable levels. The total integral is the summation of the integral estimates.

Due to the recursive nature of this technique. It is expected to be time consuming and computationally heavy for low tolerance values. However, It surely provides an accurate approximation for problems that is too hard or nearly impossible to solve analytically. As AQ is used for numerical integration, its principles are fundamental to how ODEs are solved numerically through adaptive step size ODE solvers. The ODE is Transformed into an integral, implicitly. Consider an initial value problem for the S compartment:

$$\frac{dS}{dt} = f(E, I, L), S(t_0) = S_0 \qquad (5)$$

To find S_{t1} for $t_1 > t_0$, both sides of (1) can be integrated and simplified into: $\text{Sy}(\text{t}_1) = S(t_0) + \int_{t_0}^{t_1} f(E, I, L)dt$, (6)

This shows that solving an ODE numerically involves approximating an integral at each step. The challenge is that the integrand $f(E, I, L)$ depends on the solution itself. So, at any point t between t_0 and t_1, to evaluate the integrand, the value of E, I, L is needed. ODE solvers, like Runge-Kutta(RK) methods, are used to estimate E, I, L within the interval to approximate the integral.

Similar to adaptive Simpson's, where two estimates are made, RK methods, of order 4 and 5, are calculated using mostly the same function evaluations which is computationally efficient. To calculate the estimates S_{n+1}, with RK methods, the integral in (6) becomes a weighted sum of s intermediate values, called stages or slopes, denoted k_i, where each k_i is an evaluation of the function $f(S_n)$: $k_i = f(S_n + \sum_{j=1}^{s-1} a_{sj} k_j$ $S(t_1) = S(t_0) + h \sum_{i+1}^{s} b_i k_i$

Where:

- c_i (nodes) represent the fraction of the step h at which the function is evaluated for each stage.
- a_{sj} (weights for intermediate stages) determine how the previously calculated k_j values contribute to the estimate of S at which f is evaluated for the current stage k_i. This means each k_i only depends on $k_1, ..., k_{i-1}$.
- b_i (weights for the estimate) specify how each stage k_i contributes to the final approximation of S_{n+1}.

These values can be combined in what is called Butcher Tableau for convenience [10]. each of these values are depends on the order of the RK method used. Here is the Butcher Tableau for RK of order 4, b_i, and of order 5, $\hat{b}_i$.

Table 2. Coefficients for RK5(4)7M

c_i	a_{ij}						$\hat{b}_i$	b_i
0							$\frac{35}{384}$	$\frac{5179}{57600}$
$\frac{1}{5}$	$\frac{1}{5}$						0	0
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					$\frac{500}{1113}$	$\frac{7571}{16695}$
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				$\frac{125}{192}$	$\frac{393}{640}$
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			$-\frac{2187}{6784}$	$-\frac{92097}{339200}$
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		$\frac{11}{84}$	$\frac{187}{2100}$
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0	$\frac{1}{40}$

Fig. 1. The Butcher Tableau describing the coefficients of the rk4 and rk5 methods used in adaptive runge kutta [10]

The Local Truncation Error(LTE) is estimated by the formula: $LTE \approx y_{n+1}^{(5)} - y_{n+1}^{(4)}$ If the LTE is less than or equal to the set tolerance, the higher order solution is accepted as the result of the current step and the integration proceeds. If the LTE is too large, the step is rejected, and the solver re-estimates with a smaller step size.

Step size prediction is what defines the solver to be adaptive. It is done using the estimated error and the orders of the two methods that is used for estimation: $h_{new} = h_{old}.F.(\frac{T}{LTE})^{\frac{1}{(p+1)}}$, where T is the set tolerance, $p+1$ is the order of the higher-order method, which is 5 in DP method, and F is a safety factor, 0.8 or 0.9, that prevents oscillations and ensures robustness. Limits are also often placed on how much the step size can change in a single step, not increasing by more than a factor of 5 or decreasing by less than a factor of 0.2.

D. PINN

1) Neural Network Architecture : A Physics Informed Neural Network (PINN) model was developed to approximate the tuberculosis compartment values as continuous functions of time. The PINN is a fully connected forward neural network that takes a single input t and outputs the four compartment variables S(t), E(t), I(t) and L(t) simultaneously (one output neuron for each compartment). The network consists of an input layer, several hidden layers of neurons, and an output layer of 4 dimensions. This paper's implementation utilized two hidden layers with 128 neurons each and Tanh activation functions (a smoothly saturating nonlinearity) for all hidden units. This network size was chosen to provide sufficient

capacity to capture the dynamics of the system. The output layer applies the softplus activation function to ensure all predicted compartment values remain non-negative

2) Physics-Informed Neural Network Formulation: The PINN is trained to satisfy the system of ordinary differential equations (ODEs) that govern the tuberculosis model, in lieu of using a traditional numerical solver. Let f_S, f_E, f_I and f_L denote the right-hand sides of the ODEs in equations 1-4. Let $S_\theta(t)$ be the neural network's prediction for S at time t, where theta represents all trainable weights and biases. The same notation is assumed for $E_\theta(t)$,$I_\theta(t)$ and $L_\theta(t)$. 4 types of losses were used to guide the network into the correct solution. The first loss is the physics residual, describing how well the model fits the structure of the equation. For a particular collection point t_c, automatic differentiation (provided by PyTorch) was used to compute the time derivative of the network's output at that point, e.g. $\frac{dS}{dt}|_{(t=t_c)}$. The S physics residual $R_S(t)$ is then formed as $R_S(t) = \frac{dS}{dt}|_{(t=t_c)} - f_S(t_c)$ Where $f_S(t_c)$ is the R.H.S of equation 1 estimated at $E(t_c)$, $I(t_c)$, and $L(t_c)$. Similar procedures were followed to estimate R_E,R_I and R_L. Since this residual is the difference between L.H.S and R.H.S at any of equations [1-4], the ideal value for it must be 0. To ensure the continuous accuracy of the model at any collection point, a set of 200 collection points from $t = 0$ to $t = 25$ was used to estimate the physics loss in the loss function across the entire time region of interest. The tensors of estimate from each compartment are then judged using the selected criterion against a tensor of zeros. This results in 4 scalars $L_{S-physics}$,$L_{E-physics}$, $L_{I-physics}$ and $L_{L-physics}$, each represents the loss of a particular ODE from equations 1-4. To estimate one scalar representing the entire physics loss, a weighted sum was employed.

$$L_{\text{physics}} = \lambda_S \cdot L_{S\text{-physics}} + \lambda_E \cdot L_{E\text{-physics}} + \lambda_I \cdot L_{I\text{-physics}} + \lambda_L \cdot L_{L\text{-physics}} \quad (7)$$

The weights of each loss were determined empirically through the corresponding loss value during experimentation. The higher the loss, the lower the weight and vice versa. This ensures equal contributions from each ODE equation without making the model biased towards one equation only.

The second loss is the initial condition loss. The provided initial conditions are at $t_0 = 0$. Thus, a tensor was formed from $S_\theta(t_0)$, $E_\theta(t_0)$,$I_\theta(t_0)$ and $L_\theta(t_0)$. This tensor was judged against a tensor of the real initial conditions using the selected criterion. This also results in another scalar, L_{IC} representing how well the model follows the initial condition.

The third loss is the initial derivative loss. Using the initial conditions, the derivative terms $\frac{dS}{dt}$, $\frac{dE}{dt}$, $\frac{dI}{dt}$ and $\frac{dL}{dt}$ can be estimated at $t = 0$ from equations 1-4. Those real values are made into a tensor and judged against the network's time derivatives at $t = 0$ via autograd using the selected criterion. This results in a third scalar, $L_{\text{IC-Derivative}}$ representing how accurately the model is following both the initial conditions and the physics of the ODE system.

The fourth loss is the data loss. A small number of solution points for which high confidence values of the compartments are available from a conventional ODE solver were used as a final loss for the model. For each of the known data points, the solution at that data point was compared to the model's prediction. In the implementation, a 2D tensor representing the true solutions from $t = 0$ to $t = 20$ was judged against the tensor produced by the neural network for the same time points using the chosen criterion. This results in a single scalar L_{data} representing how well does the model follow the data points. The reasoning behind using this loss will be elaborated upon in the results section. The total loss for the network is hence estimated as:

$$L = \lambda_{\text{physics}} \cdot L_{\text{physics}} + \lambda_{\text{IC}} \cdot L_{\text{IC}} + \lambda_{\text{IC-derivative}} \cdot L_{\text{IC-derivative}} + \lambda_{\text{data}} \cdot L_{\text{data}} \quad (8)$$

Where each of the coefficients is determined empirically.

3) Training Procedure and Model Parameters: The PINN training was implemented in Python using PyTorch. NumPy was used for general numerical routines. Matplotlib was used for visualization of results. The model training was performed on a standard workstation and took advantage of GPU acceleration for faster computation. The chosen criterion was mean square error. The network was trained using a two-phase optimization strategy, which was proven in literature to be effective in PINN applications. In the first phase Adam Optimizer was used with an initial learning rate of 0.001 and an exponential decay of 1% every 1000 epochs (a stochastic gradient descent method with adaptive learning rates) gradually decrease the learning rate as training progressed. The Adam phase lasted 30000 epochs. During this phase, adaptive loss weighting and curricilum learning were implemented as shown in Table I below:

Epoch	$\lambda_{physics}$	λ_{data}
$0 <$ epoch < 1000	1	1000
$1000 \leq$ epoch < 3000	10	1000
$3000 \leq$ epoch < 7000	100	1000
$7000 \leq$ epoch < 12000	100	100
$12000 \leq$ epoch < 20000	100	10
epoch ≥ 20000	100	1

TABLE I
TRAINING SCHEDULE FOR PHYSICS AND DATA LOSS WEIGHTS

Starting with a low physics coefficient and gradually increasing it is standard practice in literature. It aims to define the boundaries of the model by initial conditions and a rough estimation with physics. As the model converges into the correct parameters, the coefficient for physics is increased since their gradients decrease as the model becomes more accurate. The reasoning behind the data points loss coefficient will be elaborated upon in the results section. In the second phase of training, L-BFGS optimizer, a quasi-Newton full-batch optimization method, was utilized. L-BFGS was used to fine-tune the network parameters and achieve very low training error once the solution was already close to the optimum. The optimizer ran until convergence (with a maximum of a few hundred iterations, which was sufficient for the loss to plateau).

The combination of Adam (for robust, fast initial convergence) and L-BFGS (for precise final optimization) leverages the strengths of both optimizers and is commonly recommended in PINN implementations. At the end of training, the PINN provided a continuous approximation of the compartment dynamics that satisfied the tuberculosis ODE system to high accuracy.

V. Results

A. Heun method results

The method demonstrates acceptable accuracy in approximating the tuberculosis compartment values over the time range of 0 to 20 years. The Mean Squared Errors (MSE) compared to the reference solution are as shown in Table II The closeness of the method to the solution of [1] is

Compartment	MSE
S	3.687817
E	3.394948
I	0.022697
L	0.000002

TABLE II
MEAN SQUARED ERROR FOR EACH COMPARTMENT IN HEUN METHOD

demonstrated in figure 2 The median execution time for

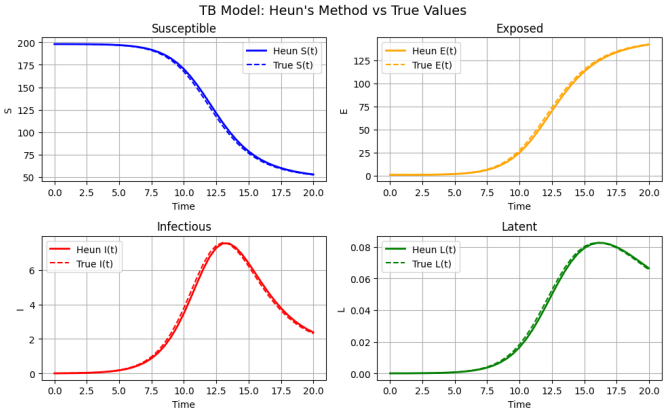

Fig. 2. The solution of Heun's method compared to the solution of [1]

this method is about 0.67961 ms. Thus, although the Mean Squared Errors (MSEs) obtained using this method are higher compared to more advanced methods like Runge-Kutta, the computational efficiency compensates for the slight loss in accuracy. Therefore, this method is particularly useful in applications where execution speed is more critical than high accuracy, such as in real-time or embedded systems.

B. Classical 4th order Runge Kutta Results

The method demonstrated the most accurate results out of all methods implemented in this paper in approximating the tuberculosis compartment values over the time range of 0 to 20 years. The Mean Squared Errors (MSE) compared to the reference solution are as follows: The closeness of the method to the solution of [1] is also demonstrated in figure 3

Compartment	MSE
S	0.000129
E	0.000120
I	0.000001
L	0.000000

TABLE III
MEAN SQUARED ERROR FOR EACH COMPARTMENT IN CLASSICAL RUNGE KUTTA

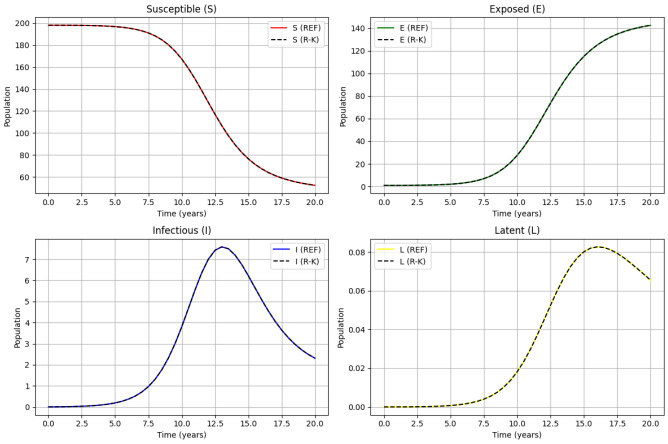

Fig. 3. The solution of classical rungue kutta compared to the solution of [1]

Those results indicate a very accuarate computation of the solution for the ODE system. This accuracy was achieved with only a step size of 0.5. This allowed the method to demonstrate a reasonable time of execution, with a median of 1.2 ms.

C. Adaptive Runge Kutta results

The adaptive method implemented demonstrates high accuracy in approximating the tuberculosis compartment values over the time range of 0 to 20 years. The Mean Squared Errors (MSE) compared to the reference solution are as follows: The closeness of the method to the solution of [1] is also

Compartment	MSE
S	0.016306
E	0.015111
I	0.000112
L	0.000000

TABLE IV
MEAN SQUARED ERROR FOR EACH COMPARTMENT IN ADAPTIVE RUNGE KUTTA

demonstrated in figure 4

These results indicate that the adaptive integration method closely follows the reference solution, with particularly high accuracy for the L (Latent) compartment, where the MSE is effectively zero down to 6 significant figures. This is largely due to the low magnitude of numbers generated in the L compartment, resulting in a very small absolute error. The errors for S and E are slightly higher but still within an acceptable range, while I shows very low error. This suggests that the method captures the dynamics of the system well.
The time taken by adaptive rk method is the fastest out of all

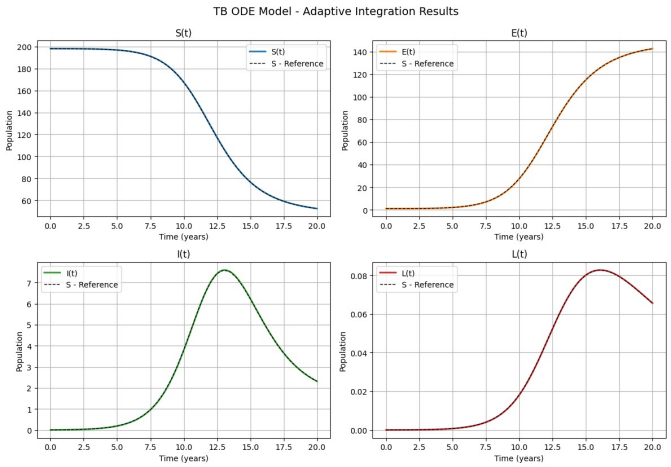

Fig. 4. The solution of adaptive rk compared to the solution of [1]

methods in this ppaer. The median time taken is only 0.0595 ms. This shows that the adaptive runge kutta method provides a very reliable accuracy in a fast time of computation.

D. PINN Results

1) Early experiments and the necessity of datapoints: At first, the PINN was estimated without relying on data loss. This resulted in graphs similar to Figure 5. As shown from the figure, the model converged at a trivial solution where none of the inputs change. Although this solution invalidates the physics of the ODE, the model couldn't escape the local minimum this solution provided. Several approaches were tested to mitigate this issue. One approach was to expand the dimensions and the depth of the neural network. Several architectures were tested, all of which yielded the same trivial solution result. Those architectures are summarized below in Table V.

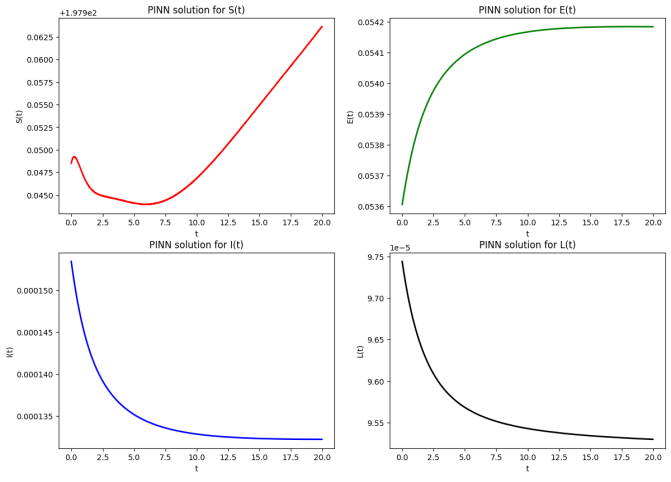

Fig. 5. The solution of the PINN model without the assistance of data points

To force the model into escaping the trivial solution, an additional loss term, defined as the ratio between the final

Number of hidden layers	Neurons per layer
5	128
3	256
9	60
9	128
5	512

TABLE V
THE ARCHITECTURES TESTED TO TRAIN A MODEL WITHOUT DATA POINTS, ALL OF WHICH DID NOT YIELD CORRECT RESULTS

S prediction and the initial S prediction multiplied by 105, was introduced. Since the true solution has the S decreasing with time, this loss condition aimed to add a large loss corresponding to the trivial solution. This did not work too, with the model being stuck at a local minimum with a loss of 105. A subsequent approach was removing this term and enforcing the physics loss with a multiplier of 100 million and an adam learning rate of 2. The goal was to give a push to the model through a large learning rate and large gradients to escape the inaccurate local minima into the true solution. This approach did not work too, with the model being stuck at the same local minima with a very large loss. The same test was repeated but with a learning rate of 10^{-3} for adam, the same results were observed. Thus, it was concluded that without a hint of errors directing the model to the true solution, the model will converge to a wrong trivial solution. Research should be conducted to determine if there is a PINN network with a set of parameters that absolutely converges for this problem without data points.

2) Results of using data points: To guide the model into the true solution, a set of data points were estimated using Gauss-Legendre quadrature. The points were selected from t=0 to t=20 with a step size of 0.5. This resulted in 40 data points that were converted to a tensor and used to train the model. After experimenting with data points, it was determined that the values of L were 2 orders of magnitude lower than I and E, which were in turn 2 orders of magnitude lower than S. This was reflected in the losses of each. The losses of L were 2 orders of magnitude lower than I and E, whose losses was in turn 2 orders of magnitude lower than the loss of S. To ensure that each compartment contributed equally to losses, the values of the physics loss coefficients were selected as following:

Coefficient	Value
λ_S	0.3
λ_E	30
λ_I	30
λ_L	30000

TABLE VI
PHYSICS LOSS COEFFICIENTS

Which reduces the physics loss equation to:

$$L = 0.3L_{\text{physics}} + 30L_{\text{IC}} + 30L_{\text{IC-derivative}} + 30000L_{\text{data}}$$

The chosen values for λ_{IC} and $\lambda_{\text{IC-derivative}}$ were chosen to be 100 and 1000, respectively. The selected curriculum learning

for λ_{data}, outlined in the methodology, was chosen such that at the beginning of the training, the data points have a very large weight that forces the model into converting to a correct minimum. As the training continues, the model's accuracy becomes higher and higher, which reduces the need to rely largely on the data points. Since they on their own have an inherent error from the numerical integration, relying less on them as training progresses allows the model to fine tune its trained parameters to rely more on the ODE system's physics and initial conditions, which results in a more accurate training. This reasoning can be shown in Figure 6. At the beginning, the model starts with a very high loss. As the model progresses towards epoch 3000, the loss is reduced by an order of magnitude, indicating the model arriving at the correct region for the true solution. Around epoch 7000, the model has already achieved a significantly lower loss. Thus, the data loss coefficient is reduced to allow the model to rely more in physics and initial conditions. The same reasoning can be followed as the model approaches the final epochs to illustrate the reasoning behind the chosen curriculum learning.

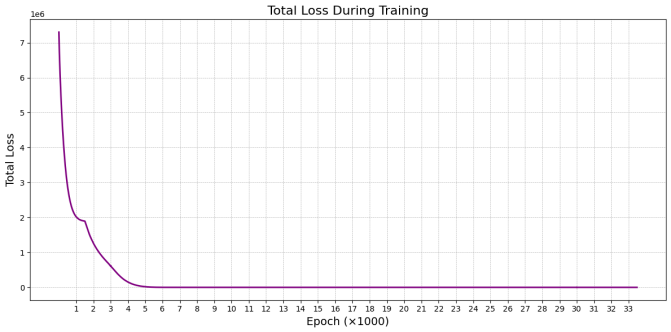

Fig. 6. The loss of the model at different epochs. It converges into the true minimum around epoch 5000

Figure 7 shows the model's solution compared to the numerical solution from [1]. As inferred from the graph, the model perfectly follows the general trend and the correct values for the entire range of the solution provided by [1]. The values of mean square error across the time range was computed; the results are shown in Table VII Several other experiments with

Compartment	MSE
S	1.37
E	1.27
I	0.009
L	0.000001

TABLE VII
MEAN SQUARED ERROR FOR EACH COMPARTMENT IN THE PINN MODEL

different architecture were conducted. In all of them, the model converged absolutely to the correct solution given sufficiently good architecture and sufficiently accurate points. The model performed exceptionally well during extrapolation too. Figure 8 shows the model's extrapolation over the range [0:100], five times the original training range: As can be shown from the figure, the model's accuracy is exceptionally high for all

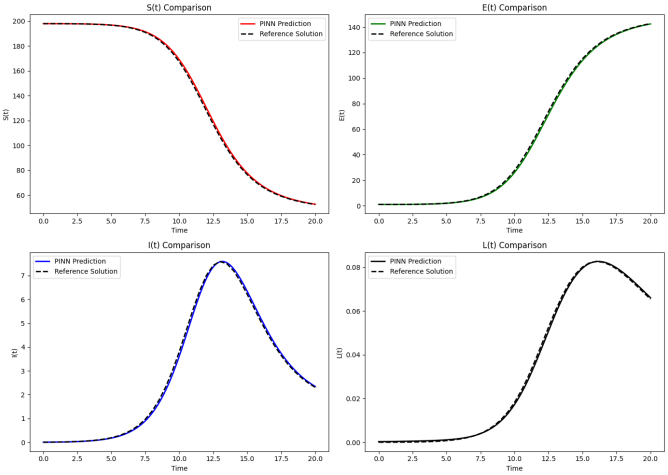

Fig. 7. The solution of the PINN model compared to solution of [1]

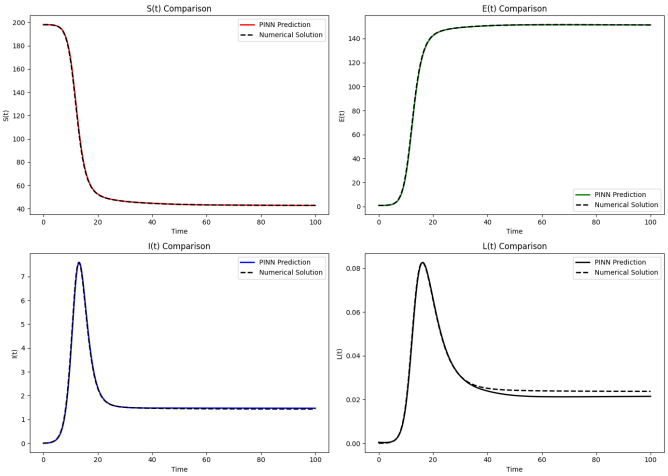

Fig. 8. The model's extrapolation up to t=100, demonstrating solid accuracy

compartments. The true solution was computed using Gauss-Legendre quadrature from t=0 to t=100 with 10000 points. Despite using only 40 data points that were computed from numerical integration on the range of 0 to 20, the PINN was able to extrapolate a range five times the original range without losing much accuracy. This is illustrated below in the table showing the MSE across the entire range: Despite taking a

Compartment	MSE
S	0.281159
E	0.264011
I	0.002425
L	0.000004

TABLE VIII
MEAN SQUARED ERROR FOR EACH COMPARTMENT OVER THE REGION FROM T=0 TO T=100

relatively long time to train, the PINN demonstrates excellent accuracy and reliability. Furthermore, unlike numerical methods, the PINNs are step size independent. The network provides a continuous function that can be used to calculate

the output at any time point, which is beneficial for continuous analysis and visualizing stiff systems.

E. Summary of all results

Figure 9 summarizes all the results of each method implemented and tested in this paper compared to the result of [1]. As can be inferred from the figure, all methods did exceptionally well at predicting the correct solution. Table IX

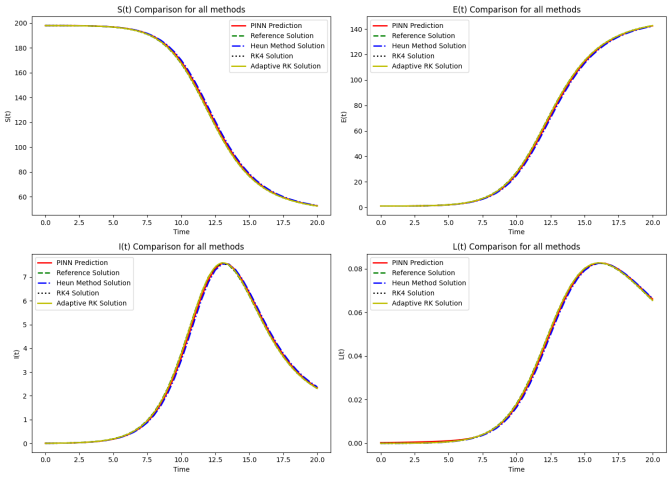

Fig. 9. The results from all methods plotted against the solution of [1]

Shows the different mean square errors of each compartment of each method. As described in the results section, the classical 4th order runge kutta is the most accurate one of them for a chosen step size.

Var	Heun's Method	Classical RK4	Adaptive RK	PINN Model
S	3.687817	0.000129	0.016306	1.37
E	3.394948	0.000120	0.015111	1.27
I	0.022697	0.000001	0.000112	0.009
L	0.000002	0.000000	0.000000	0.000001

TABLE IX
MEAN SQUARED ERROR COMPARISON ACROSS DIFFERENT METHODS FOR EACH COMPARTMENT

Table X shows time comparisons of the executions of each method. As per the table, the adaptive rk method is the fastest by a very large margin. It's followed by the PINN model. However, it should be noted that this time doesn't take into account the training time, which lasts for about 4 minutes.

Method	Median Execution Time (ms)
Heun's Method	0.67961
Classical 4th Order Runge-Kutta	1.2
Adaptive Runge-Kutta	0.0595
PINN Model	0.38
Solve IVP Python	6

TABLE X
EXECUTION TIME COMPARISON ACROSS THE DIFFERENT METHODS

VI. CONCLUSION

In this study, four distinct methods were examined for solving a nonlinear system of ordinary differential equations modeling tuberculosis transmission: Heun's method, the classical fourth-order Runge–Kutta (RK4), the Adaptive Runge–Kutta method, and Physics-Informed Neural Networks (PINNs). Each method was evaluated in terms of accuracy and computational efficiency to highlight their respective strengths and limitations.

The results confirm that RK4 provides the highest accuracy, achieving minimal mean squared error across all compartments, making it a dependable choice for precise modeling. The adaptive method, while slightly less accurate, was the fastest and most efficient, offering a compelling trade-off for real-time or resource-constrained scenarios. Heun's method, though relatively simple and less accurate, showed satisfactory performance with low computational cost, making it useful in applications where rapid estimation is preferred.

On the other hand, the PINN approach demonstrated unique capabilities. Although training required significant time and careful tuning, PINNs excelled in generalization, offering a smooth, continuous approximation that extended well beyond the original training domain. This property is particularly valuable in data-limited environments or when modeling scenarios requiring extrapolation.

Overall, the findings underscore the importance of selecting numerical techniques based on specific problem requirements. While traditional solvers remain powerful tools for structured systems, data-driven approaches like PINNs open new avenues for integrating physics with machine learning, potentially transforming how epidemiological modeling is approached in complex and uncertain conditions.

REFERENCES

[1] Schiesser, W. E. (2014). Differential equation analysis in biomedical science and engineering. URL

[2] Applied Numerical Methods with Python for Engineers and Scientists (1st ed.). (2021). McGraw-Hill Higher Education. URL

[3] S. Kanwal, M. K. Siddiqui, E. Bonyah, K. Sarwar, T. S. Shaikh, and N. Ahmed, "Analysis of the epidemic biological model of tuberculosis (TB) via numerical schemes," Complexity, vol. 2022, Art. ID 5147951, 13 pp., Mar. 2022, doi: 10.1155/2022/5147951.

[4] E. D. Tadesse, M. A. Bhat, and A. Ayele, "A deterministic compartment model for analyzing tuberculosis dynamics considering vaccination and reinfection," Heliyon, vol. 9, no. 10, p. e19674, 2023, doi: 10.1016/j.heliyon.2023.e19674

[5] S. Side, A. M. Utami, S. Sukarna, and M. I. Pratama, "Numerical solution of SIR model for transmission of tuberculosis by Runge–Kutta method, Journal of Physics: Conference Series, vol. 1040, p. 012021, 2018

[6] Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). Physics Informed Deep Learning (Part I): Data-driven solutions of nonlinear partial differential equations. arXiv (Cornell University). URL

[7] Rasheed, A. (2024). Modeling Dynamical Systems with Physics Informed Neural Networks with Applications to PDE-Constrained Optimal Control Problems. https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3130805?show=full

[8] Pal, B., Rahaman, R., et.al. (2025). A deep learning approach to model and predict tuberculosis using PINNs. SSRN: URL

[9] Lu, L., Meng, X., Mao, Z., & Karniadakis, G. E. (2019). DeepXDE: A deep learning library for solving differential equations. arXiv. URL

[10] Walters, S. J., Turner, R. J., & Forbes, L. K. (2022). A COMPARISON OF EXPLICIT RUNGE–KUTTA METHODS. The ANZIAM Journal, 64(3), 227–249. URL