Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Developing a Physics-Informed Neural Network to Estimate the Parameters of a Delay Differential Equation

Amogha S Shastry
Indian Institute of Technology Roorkee
Mayank Goswami
mayankfph@iitr.ac.in
Indian Institute of Technology Roorkee

Research Article

Keywords: DDE, Delayed Logistic Growth, Nicholson's blowfly data, PINN, Population dynamics

Posted Date: November 10th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11652924/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Developing a Physics-Informed Neural Network to Estimate the Parameters of a Delay Differential Equation

Amogha S Shastry[1] and Mayank Goswami,[1,2]

[1]Department of Physics, IIT Roorkee
[2]MF School of AI and DS, IIT Roorkee
mayank.goswami@ph.iitr.ac.in

***Abstract*—Delay differential equations (DDEs) arise naturally in physics, engineering and the life sciences, for example in lasers with delayed optical feedback, machining chatter and population dynamics. Unlike ordinary differential equations, they require a history function and are harder to treat numerically. Physics-informed neural networks (PINNs) embed a governing equation in the training loss of a neural network, which makes them well suited to problems that combine measured data with unknown parameters. We use the delayed logistic equation, fitted to Nicholson's blowfly data, as a test case for a PINN applied to a DDE, implemented in MATLAB® [1] with the Deep Learning Toolbox™ [2]. The network learns the delay τ, growth rate r and carrying capacity K together with the population curve, and we compare the result obtained for different physics weights with a solution from the traditional solver `dde23`. The learned parameters ($\tau \approx 8.8$ days, $r\tau \approx 1.8, K \approx 2400$) are close to published values, and the two solutions agree in period and amplitude. We also note that the PINN is reliable only within the time range in which the equation was enforced.**

***Index Terms*—DDE, Delayed Logistic Growth, Nicholson's blowfly data, PINN, Population dynamics**

I. Introduction

ORDINARY differential equations (ODEs) arise in many branches of science and engineering and have been extensively studied. Delay differential equations (DDEs) incorporate a delay term within their form which often yields interesting dynamics vastly different from what they would be without it. See [3] for a more comprehensive introduction to the behavior of delay differential equations.

DDEs arise across physics in fields such as fluid dynamics and wave mechanics. The authors of [4] have proposed a non-linear model which incorporates oceanic wave transit effects through a delayed feedback. In [5], the equations of motion governing the dynamics of light-like tachyon condensation are found to be delay-differential equations. [6] examine the dynamics of a prototypical bistable system under influence of noise by employing a delayed feedback. The authors of [7] and [8] employ DDEs in order to model the frequency- and amplitude-dependent material properties of filler-reinforced rubber. [9] provide a DDE approach for simulating passive mode-locking in quantum dot lasers. Thus, the use case of DDEs is multidisciplinary and its intersection with physics is evident.

Corresponding author: mayank.goswami@ph.iitr.ac.in

In this report we apply a Physics-Informed Neural Network (PINN) [10] to the delayed logistic equation,

$$x'(t) = rx(t)(1 - x(t-\tau)/K) \tag{1}$$

subject to initial history function

$$x(\theta) = \phi_0(\theta) > 0, \theta \in [-\tau, 0]$$

using Nicholson's blowfly data (see 1) [11], [12]. PINNs incorporate two kinds of losses in the loss function: a data loss and a physics loss. The data loss is the mean squared error (MSE) between the network output and training data. For our purposes, the physics loss refers to the mean square residual of 1 computed at random collocation points by using the value of $x'(t)$ as obtained by automatic differentiation. The delayed term is evaluated using either the network estimation $\hat{x}(t)$ or the history function if $0 < t < \tau$.

This network is trained on the data and on the differential equation residual, and learns the growth rate r, the carrying capacity K and the time delay τ together with the population curve. We compare the result with a numerical solution from the traditional solver, MATLAB®'s `dde23`, and with the results in [3, Chapter 5 Sec 5.9], and we discuss the limits of the method outside the trained time range. The same approach can in principle be adapted to other DDEs.

PINNs have mostly been developed for ODEs and PDEs, and their application to DDEs is still comparatively limited [13]. Parameters of the delayed logistic equation have previously been estimated from Nicholson's blowfly data using other methods [14]. Here, we examine how a PINN performs on this problem, estimating r, K and τ directly from the measurements.

II. Problem Definition

In this section, we introduce our main problem statement and explain the origin of the various terms involved

in the equation. We also elaborate on the objectives of this paper.

The main goal of this paper is to learn a function $\hat{x}(t)$ which closely fits not only our training data but also our form of the differential equation 1. The history function we have chosen is $\phi_0(\theta) = x_0 e^{\theta/\tau}$, where x_0 is the first data point in our training set. This incorporates a continuous transition.

We also aim to estimate the parameters r, K and τ of the delayed logistic growth model. By examining the data and residual errors obtained by our model, we can compare the PINN solution to the numerical solution obtained by `dde23` which can inform us about the effectiveness of our PINN in estimating a solution for this DDE. We can also compare the results of the estimated parameters with those available in literature.

III. Relevant Theory

In this section, we provide some relevant theory on the DDE we've used as well as some background on PINNs. We also expand further on Nicholson's data that we've for our training.

A. The Logistic Growth Model

The logistic growth model [15] was first published in order to model the self-limiting effects of the population growth of biological species. Due to finite resources, growth is sustained only up until one point (where the carrying capacity K is reached). Thus, we have the $(1-x/K)$ term. The delay term [16] was introduced later as a modification to the original logistic growth in order to simulate the time lag associated with biological processes such as gestation, digestion, etc. It facilitated the incorporation of periodic motions, bifurcations, etc. in scalar population dynamics.

The equation under consideration (namely, 1) reads

$$x'(t) = rx(t)(1 - x(t-\tau)/K)$$

whereas without delay it would simply be

$$x'(t) = rx(t)(1 - x(t)/K).$$

Here r refers to the intrinsic growth rate of the population, K the carrying capacity and τ the time delay to account for biological processes. The equation has two equilibrium points, $x_1^* = 0$ and $x_2^* = K$. In order to examine the stability of x_2^*, the more biologically relevant equilibrium, we linearize the equation about $x_2^* = K$, and x_1^* is unstable anyways [3]. First, we make the following substitutions to make our equation a bit more simple; $s = rt, \rho = r\tau$ and

$$y(t) = x(t)/K.$$

We note that the equilibrium point $x_2^* = K$ now becomes $y^* = 1$. After further simplification and following the steps in [3], we obtain:

$$\frac{dy(s)}{ds} = y(s)(1 - y(s-\rho)) \tag{2}$$

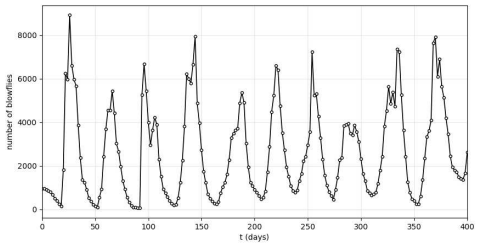


Fig. 1: The plot is of samples collected bi-daily for 722 days and plotted for 400 days. The periodic behavior of the population is evident.

When we further linearize (2) about y^* by making a substitution, $u(s) = y(s) - 1$ we obtain the following linear delay-differential equation:

$$\frac{du(s)}{ds} = -u(s-\rho) \tag{3}$$

Further, if we assume the solution of (3) to be of the form $e^{\lambda s}$, we obtain the following characteristic equation:

$$\lambda = -e^{-\lambda\rho} \tag{4}$$

We then enforce the condition for stability, i.e, $\Re(\lambda) < 0$, proceed as in [3] and obtain the following result [17], [18]:

$$0 \leq \rho < \pi/2. \tag{5}$$

This means that, for $\rho \in [0, \pi/2)$, the equilibrium is stable, and perturbations decay (monotonically for $\rho < e^{-1}$, with oscillations above that). Thus, in order to observe sustained oscillations like in Nicholson's data (see Section C), we would need to enforce the following condition:

$$\rho > \pi/2$$

which we implement in the Methods section.

B. On PINNs

The network first estimates a function $\hat{x}(t)$ and numerically computes the time derivative $\hat{x}'(t)$ via automatic differentiation. We train the network on a loss function given by the form:

$$L = \lambda_{data} L_{data} + \lambda_{phys} L_{phys}, \tag{6}$$

where

$$L_{data} = \frac{1}{N}\sum [x(t_i) - \hat{x}(t_i)]^2$$
$$L_{phys} = \frac{1}{M}\sum [r\hat{x}(t_j)(1 - \hat{x}(t_j - \tau)/K) - \hat{x}'(t)]^2$$

In order to compute the delayed term required in the physics loss, we use the network estimate whenever $t_j > \tau$ and $r\hat{x}(t_j)(1 - \phi_0(t_j - \tau)/K)$ when $t_j \in [0, \tau]$. Thus, r, K and τ become trainable parameters.

C. Nicholson's Data

We include a plot of Nicholson's data for easy viewing for the reader. Please see Fig. 1.

IV. Methodology

We utilized the Deep Learning Toolbox™ available within MATLAB® R2025a in order to design and train our PINN.

A. Network Properties

Our network properties are summarized in Table I.

TABLE I: Network Properties

Name	Type	Learnable Sizes	Number of Learnables
in	Feature Input	-	0
fc1	Fully Connected	$W : 48 \times 81$ $B : 48 \times 1$	3936
tanh1	Tanh	-	0
fc2	Fully Connected	$W : 48 \times 48$ $B : 48 \times 1$	2352
tanh2	Tanh	-	0
fc3	Fully Connected	$W : 48 \times 48$ $B : 48 \times 1$	2352
tanh3	Tanh	-	0
fc4	Fully Connected	$W : 48 \times 48$ $B : 48 \times 1$	2352
tanh4	Tanh	-	0
out	Fully Connected	$W : 1 \times 48$ $B : 1 \times 1$	49

We used a fully connected Fourier feature network [19] consisting of 81 feature inputs on the network input, which is a normalized time $s = t/400$ mapped to the features $\{s, \sin(2\pi ks), \cos(2\pi ks)\}, k \in \{1, \ldots, 40\}$. It behaves as a fully connected multilayer perceptron (MLP) with 4 hidden layers consisting of 48 tanh activated units, as visualized in Fig. 2. Overall, we have ∼11,000 learnable parameters.

B. Training

We trained the network for $N_{\text{iter}} = 10000$ iterations with an initial guess of $\tau = 9$, $r\tau = 2.1$ and K to be half of the maximum of the training data. The values of τ and r were frozen for the first $N_{\text{freeze}} = 6000$ iterations. K and the network weights were learned throughout, which allowed the network to fit the training data before learning r and τ. We restricted τ to between 3 and 30 days, and r to at most 1 (i.e, $r \leq 1$ day^{-1}). This allows the estimates to be physically plausible whilst preventing issues that occur with runaway parameter values. We also imposed the following condition on the training:

$$(r\tau)_{\min} = \pi/2 + 0.05 \approx 1.62,$$

by projecting r and τ into the allowed region after each update.

This allows our model to support sustained oscillations (Sec. III-A). A threshold of 0.05 is employed so that we remain in the oscillatory regime. If the learned value of $r\tau$ ever settled on the boundary, it would mean that the unconstrained optimized value would lie in the damped region characterized by decaying oscillations. This could either point to noise in the training data, a mismatch between model and training data or a poor local minimum that the model cannot escape.

The physics weight λ_{phys} ramps linearly from 0 to the target value for the first $N_{\text{warm}} = 3000$ iterations, prioritizing the training data over the physics residual in the initial phase of training. The time and population were both normalized as $s = t/400$ and $u = x/x_s$, with x being the population and x_s, the maximum of the training data, following which the equations were rewritten and the PINN was trained.

We employed the Adam optimizer with a learning rate of 5×10^{-4} for r, K and τ and 10^{-3} for weights while implementing full-batch data loss and element-wise gradient clipping (threshold ±1). The data loss uses all the training points and the physics loss uses 600 random collocation points that are resampled from 0 to 400 days at every iteration. The PINN was built by incorporating the `dlgradient`, `dlarray` and `adamupdate` in a custom loop.

The training data consisted of population numbers recorded over a span of 400 days, in which 300 days was allocated for training and the rest for validation.

V. Results

In this section, we report our results obtained after multiple training instances and for varying values of the physics weights λ_{phys} and the tables report the results of a typical representative run. We discuss the limitations of our methods, the regions of applicability and highlight the distinguishing characteristics. We obtain visual plots which we compare with the plot of Nicholson's data as well as a plot of the numerical solution obtained by `dde23` using the parameter values given in [3]. We trained the PINN with 6 different physics weights, 0, 0.01, 0.03, 0.05, 0.1 and 0.5. The plots obtained for the network estimation $\hat{x}(t)$ for these weights are provided below (See Figure 4), as are the respective MSE values for each trial. The plot of $x(t)$ obtained by solving the DDE numerically using `dde23` is given in Figure 5 for comparison. See Figure 3 for a plot of our losses against the physics weight used.

A. Baseline: physics weight = 0

The network simply memorized the data by interpolating every training point, and not the dynamics. The curve oscillates rapidly in the validation region and occasionally dips below zero in the training and validation regions, which is unphysical. The values of the parameters remain unchanged from initialization and are not learned. The physics error is extremely large which indicates that this network is unable to reproduce the governing dynamics when trained on data alone.

B. Physics-Informed runs

The physics-informed runs can be categorized into two groups: those where the network collapsed to 0 (for the weights 0.1 and 0.5) and those that did not collapse to 0.

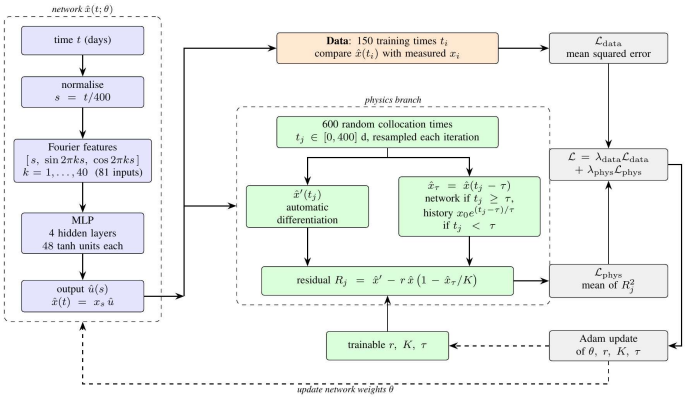


Fig. 2: A visualization of our Fourier Feature Network

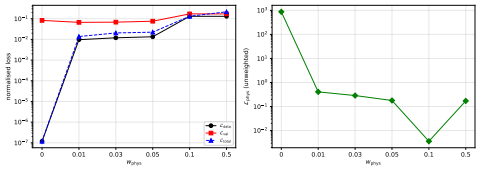


Fig. 3: A plot of losses against physics weights

Validation loss at $\lambda_{\text{phys}} = 0.01$ is about 20% lower than the baseline (6.66×10^{-2} against 8.28×10^{-2}). Raising the weight to 0.05 lowers the residual (0.41 to 0.18) but raises data (0.0097 to 0.0134) and validation loss (0.0666 to 0.0756). See Table III for more.

Figure 3 shows that physics loss cannot purely be indicative of PINN performance as the network collapses for $\lambda_{\text{phys}} = 0.1$, and nor can data loss alone determine it, since the physics error is also large for the base trial. As such, determining the optimum weights for the PINN to work proves to be a challenge. Qualitatively, the curves for $\lambda_{\text{phys}} = 0.01$, 0.03 and 0.05 are smooth, oscillate with a period of about 37 days, and resemble the `dde23` solution of Fig. 5 much more closely than the curve for $\lambda_{\text{phys}} = 0$. They do not reproduce its amplitude, however: the PINN peaks at roughly 5,200–6,300, whereas the `dde23` solution with the parameters of [3] peaks near 8,000, with a period of about 41 days.

For $\lambda_{\text{phys}} = 0.1$ and 0.5 the network collapses to a solution close to zero. This is consistent with the DDE having the trivial solution $x \equiv 0$, which satisfies the equation exactly, so a large physics weight offers no incentive to leave it, while the data term is outweighed. The residual is small for $\lambda_{\text{phys}} = 0.1$ (3.6×10^{-3}) but not for 0.5 (1.7×10^{-1}), which suggests that the latter curve is close to, but not exactly, zero.

The learned parameters (Table II) are consistent with the values in [3] for the runs that did not collapse: τ lies within 2–4% of 9 d, r within 4–11% of 0.233 d^{-1}, K within 3–15% of 2500, and $r\tau$ within 9–13% of 2.1. No single weight is closest on all four quantities. The collapsed runs ($\lambda_{\text{phys}} \geq 0.1$) are excluded, as their parameters sit on the imposed bounds. The PINN is only reliable on the time range in which it was trained and validated. Beyond day 400 the curve is a periodic extrapolation produced by the Fourier input features, not a prediction, and should not be trusted.

While the PINN was able to reproduce the results of [3] in period and amplitude, it was unable to fully reproduce the dynamics of Nicholson's data. The output is yet a deterministic cycle that cannot replicate the irregular cycles observed in Figure 1. Thus, while we cannot treat the PINN output as a true estimate for the solution of the DDE 1, we can verify that the network output was able to replicate the numerical solution of `dde23` in period, and with amplitudes matching Nicholson's data. Further, the parameters τ, K and r were able to be estimated to an accurate degree.

TABLE II: Learned Parameters for Different Physics Weights

λ_{phys}	r (d^{-1})	K	τ (d)	$r\tau$
0	—	—	—	—
0.01	0.207	2417	8.80	1.82
0.03	0.217	2227	8.70	1.88
0.05	0.223	2131	8.62	1.92
0.1†	0.540	4331	3.00	1.62
0.5†	0.540	4325	3.00	1.62
Ref. [3]	0.233	2500	9	2.1

$\lambda_{\text{phys}} = 0$: parameters receive no gradient and are not learned.
†Collapsed runs; τ and $r\tau$ sit at their bounds (3 d and $\pi/2 + 0.05$).

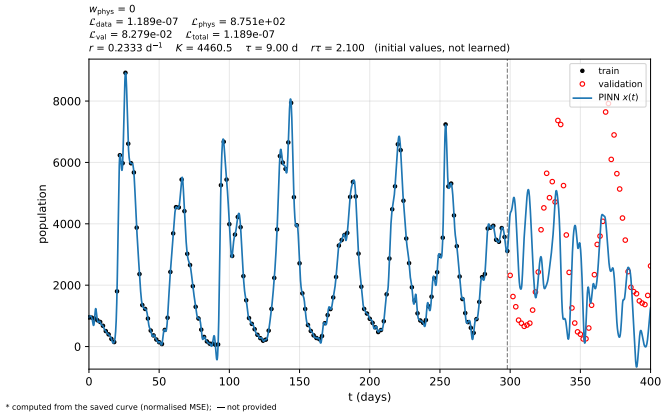
TABLE III: Final Losses for Different Physics Weights

λ_{phys}	$\mathcal{L}_{\text{data}}$	$\mathcal{L}_{\text{phys}}$	$\mathcal{L}_{\text{val}}$	$\mathcal{L}_{\text{total}}$
0	1.19×10^{-7}	8.75×10^{2}	8.28×10^{-2}	1.19×10^{-7}
0.01	9.68×10^{-3}	4.07×10^{-1}	6.66×10^{-2}	1.37×10^{-2}
0.03	1.19×10^{-2}	2.86×10^{-1}	6.81×10^{-2}	2.05×10^{-2}
0.05	1.34×10^{-2}	1.78×10^{-1}	7.56×10^{-2}	2.23×10^{-2}
0.1†	1.31×10^{-1}	3.58×10^{-3}	1.71×10^{-1}	1.31×10^{-1}
0.5†	1.31×10^{-1}	1.71×10^{-1}	1.71×10^{-1}	2.16×10^{-1}

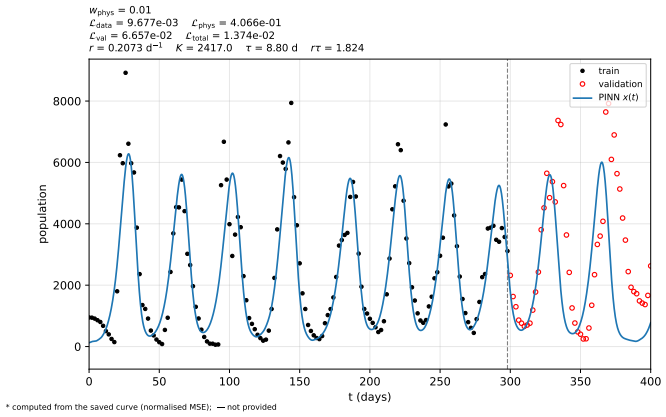
†Network collapsed to $\hat{x} \approx 0$. Losses are normalised.

VI. Video

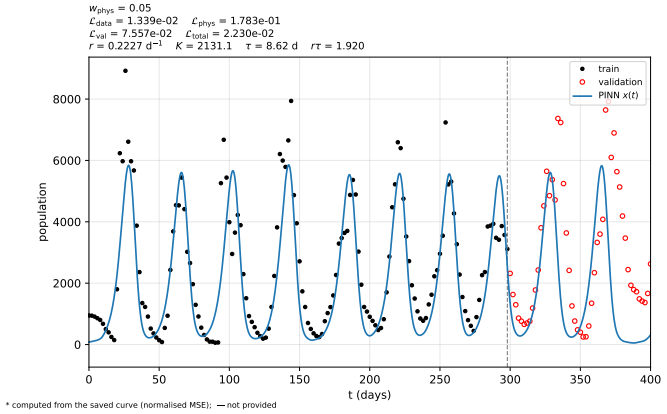
https://youtu.be/YexskfW0nAg



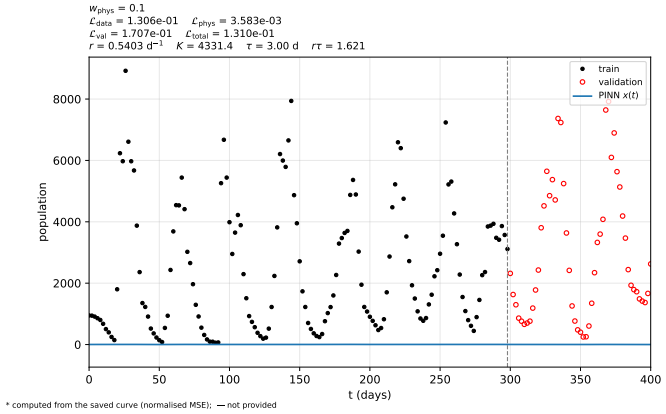
(a) $\lambda_{\text{phys}} = 0$



(b) $\lambda_{\text{phys}} = 0.01$



(c) $\lambda_{\text{phys}} = 0.05$



(d) $\lambda_{\text{phys}} = 0.1$

Fig. 4: PINN fits for different physics weights. Black dots: training data (days 0–298); open red circles: validation data (days 300–400).

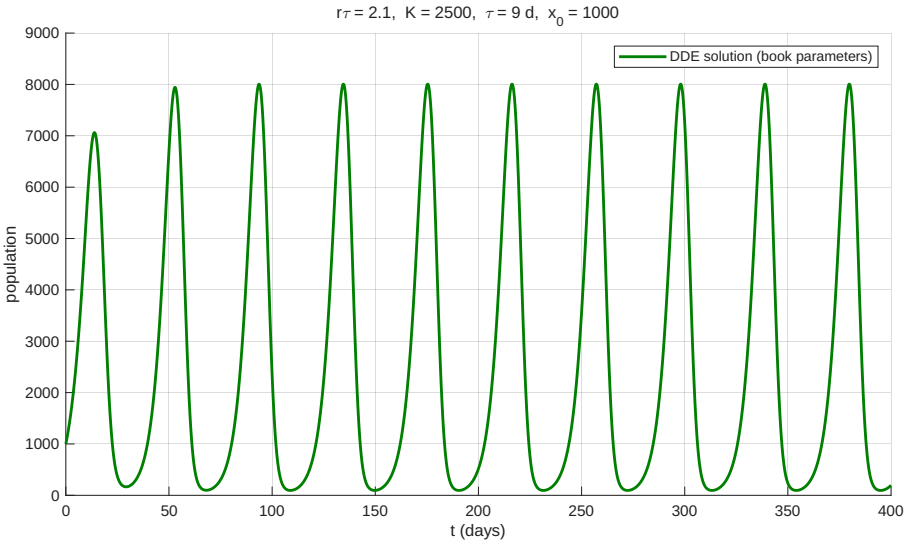


Fig. 5: Numerical solution obtained by dde23

References

[1] T. M. Inc., "Matlab version: 25.1 (r2025a)," 2025.

[2] T. M. Inc., "Deep learning toolbox version: 25.1 (r2025a)," 2025.

[3] L. J. Allen, "An introduction to mathematical biology," *(No Title)*, 2007.

[4] M. J. Suarez and P. S. Schopf, "A delayed action oscillator for enso," *Journal of Atmospheric Sciences*, vol. 45, no. 21, pp. 3283–3287, 1988.

[5] F. Beaujean and N. Moeller, "Delays in open string field theory," 2010.

[6] L. S. Tsimring and A. Pikovsky, "Noise-induced dynamics in bistable systems with delay," *Physical Review Letters*, vol. 87, no. 25, p. 250602, 2001.

[7] M. Rendek and A. Lion, "Amplitude dependence of filler-reinforced rubber: Experiments, constitutive modelling and fem – implementation," *International Journal of Solids and Structures*, vol. 47, no. 21, pp. 2918–2936, 2010.

[8] P. Höfer and A. Lion, "Modelling of frequency- and amplitude-dependent material properties of filler-reinforced rubber," *Journal of the Mechanics and Physics of Solids*, vol. 57, no. 3, pp. 500–520, 2009.

[9] M. Rossetti, P. Bardella, and I. Montrosset, "Modeling passive mode-locking in quantum dot lasers: A comparison between a finite-difference traveling-wave model and a delayed differential equation approach," *IEEE Journal of Quantum Electronics*, vol. 47, no. 5, pp. 569–576, 2011.

[10] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[11] A. J. Nicholson, "The self-adjustment of populations to change," in *Cold spring harbor symposia on quantitative biology*, vol. 22, pp. 153–173, Cold Spring Harbor Laboratory Press, 1957.

[12] D. R. Brillinger, J. Guckenheimer, P. Guttorp, and G. Oster, "Empirical modelling of population time series data: the case of age and density dependent vital rates," in *Selected Works of David Brillinger*, pp. 335–360, Springer, 2011.

[13] S. Yao, V. Kumar, and R. Guglielmi, "Physics informed neural networks for nonlinear delay differential equations," *arXiv preprint arXiv:2607.00380*, 2026.

[14] L. Wang and J. Cao, "Estimating parameters in delay differential equation models," *Journal of agricultural, biological, and environmental statistics*, vol. 17, no. 1, pp. 68–83, 2012.

[15] P.-F. Verhulst, "Notice sur la loi que la population suit dans son accroissement," *Correspondence mathematique et physique*, vol. 10, pp. 113–129, 1838.

[16] G. E. Hutchinson *et al.*, "Circular causal systems in ecology," *Ann. NY Acad. Sci*, vol. 50, no. 4, pp. 221–246, 1948.

[17] J. M. Cushing, *Integrodifferential Equations and Delay Models in Population Dynamics*, vol. 20 of *Lecture Notes in Biomathematics*. Berlin, Heidelberg: Springer-Verlag, 1977.

[18] K. Gopalsamy, *Stability and Oscillations in Delay Differential Equations of Population Dynamics*, vol. 74 of *Mathematics and Its Applications*. Dordrecht: Kluwer Academic Publishers, 1992.

[19] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in neural information processing systems*, vol. 33, pp. 7537–7547, 2020.