International Journal of Multidisciplinary Studies and Innovative Technologies

e-ISSN: 2602-4888
dergipark.org.tr/en/pub/ijmsit
Research Article

2025, 9 (1), 59-63
DOI: 10.36287/ijmsit.9.1.9
Received: May 28, 2025; Accepted: June 16, 2025

Numerical Solutions of Hyperbolic Telegraph Equations using Difference Schemes and Neural Network

Aleyna Akaydın [1], Ahmed Amara [2] and Mehmet Emir Köksal [3]

[1] *Department of Mathematics, Ondokuz Mayıs University, Samsun, Turkey, aaleynaakaydin@gmail.com, ORCID: 0009-0001-0937-0775*
[2] *Department of Mathematics, Ondokuz Mayıs University, Samsun, Turkey, ahmedamara111990@gmail.com, ORCID: 0009-0006-9748-0902*
[3] *Department of Mathematics, Ondokuz Mayıs University, Samsun, Turkey, mekoksal@omu.edu.tr, ORCID: 0000-0001-7049-3398*

Abstract **–** In this study, the hyperbolic telegraph differential equation describing the behavior of certain wave-like phenomena, such as the transmission of signals along a telegraph line, is considered. Numerical solutions of a telegraph equation are computed by using various operator-difference schemes and physics-informed neural networks. The error analysis is performed, and the results are compared.

Keywords – *Hyperbolic Equations, Numerical Solution, Error Analysis, Difference Scheme, Neural Network*

I. Introduction

PDEs are equations that involve partial derivatives of a multivariable function. They play a crucial role in various fields such as physics, engineering, finance, and biology, as they can describe a wide range of phenomena, including heat conduction, fluid flow, and wave propagation [1].

Hyperbolic equations are a type of partial differential equation characterized by the propagation of waves and signals. They describe systems where information travels at finite speeds, making them essential in various fields like physics, engineering, and finance. The most common example is the wave equation, which describes the propagation of waves, such as sound waves, light waves, water waves and how waveforms evolve over time.

The hyperbolic telegraph equation is a specific type of partial differential equation that describes the propagation of signals [2] in a medium with both wave-like and dissipative characteristics. It combines elements of both wave propagation and damping, making it particularly relevant for modeling phenomena in electrical engineering and acoustics.

The telegraph equations (also known as transmission line equations) describe the voltage and current on an electrical transmission line with distributed parameters. These equations also apply to RLC circuits when they are modeled as distributed parameter systems, such as in long transmission lines [3].

Numerical methods are used to approximate the solution of the telegraph equation when an analytical solution is not feasible. There are many different numerical solution methods for telegraph equations. Finite difference methods, finite element methods, finite volume methods are some them. Stability is a critical concept in the numerical solution of partial differential equations using finite difference methods. It refers to whether small errors (e.g., from round-off or truncation) grow uncontrollably as the computation progresses.

Various initial boundary value problems can be reduced to initial value problems [4-5]. In this paper, the following initial value problem is considered:

$$\begin{cases} \dfrac{d^2u(t)}{dt^2} + \alpha \dfrac{du(t)}{dt} + Au(t) = f(t), \ \ 0 \le t \le T, \\ u(0) = \varphi, \ \ u'(0) = \psi. \end{cases} \tag{1}$$

Here A is self-adjoint unbounded operator in Hilbert space. In this paper, for the approximate solutions of the above problems, first, second, third order of accuracy difference schemes are presented in the next section. A test problem is considered and numerical solutions of the test problem are found by using difference schemes. The results are compared with each other. Moreover, approximate solutions of the test problem are also found by using neural network in the fourth section.

II. Difference Schemes

For approximate solutions of the problem (1), the following first order of accuracy difference scheme

$$\begin{cases} \dfrac{u_{k+1} - 2u_k + u_{k-1}}{\tau^2} + \alpha \dfrac{u_{k+1} - u_k}{\tau} + Au_{k+1} = f_k, \\ f_k = f(t_{k+1}), \ \ 1 \le k \le N-1, \ \ N\tau = T, \\ u_0 = \varphi, \ \ (1+\varphi\tau)\dfrac{u_1 - u_0}{\tau} + A\tau u_1 = \psi \end{cases}$$

and two different types of second order of accuracy difference schemes

$$\begin{cases} \dfrac{u_{k+1}-2u_k+u_{k-1}}{\tau^2}+\alpha\dfrac{u_{k+1}-u_{k-1}}{2\tau} \\ +\dfrac{A}{2}(u_{k+1}+u_{k-1})=f_k \\ f_k=f(t_k),\ \ 1\le k\le N-1, \\ u_0=\varphi,\ \ \dfrac{u_1-u_0}{\tau}+\dfrac{\tau}{2}Au_1=\psi+\dfrac{\tau}{2}(-\alpha\psi+f_0), \\ f_0=f(0), \end{cases}$$

and

$$\begin{cases} \dfrac{u_{k+1}-2u_k+u_{k-1}}{\tau^2}+\alpha\dfrac{u_{k+1}-u_{k-1}}{2\tau}+\dfrac{A}{2}u_k \\ +\dfrac{A}{4}(u_{k+1}+u_{k-1})\ =f_k \\ f_k=f(t_k),\ \ 1\le k\le N-1, \\ u_0=\varphi,\ \ \dfrac{u_1-u_0}{\tau}+\dfrac{\tau}{2}Au_1=\psi+\dfrac{\tau}{2}(-\alpha\psi+f_0), \\ f_0=f(0). \end{cases}$$

were developed in [6]. Stability estimates for the solutions of difference schemes were constructed. For the approximate solutions of problem (1), the following third order of accuracy difference scheme

$$\begin{cases} \dfrac{u_{k+1}-2u_k+u_{k-1}}{\tau^2}+\alpha\dfrac{u_{k+1}-u_{k-1}}{2\tau}+\dfrac{2}{3}Au_k \\ +\dfrac{1}{6}A(u_{k+1}+u_{k-1}) \\ +\dfrac{\tau^2}{12}\left[(-\alpha^3+2\alpha A)\dfrac{u_{k+1}-u_k}{\tau}-(\alpha^2-A)Au_{k+1}\right]=f_k, \\ f_k=\dfrac{2}{3}f(t_k)+\dfrac{1}{6}[f(t_{k+1})+ft_{k-1}] \\ -\dfrac{\tau^2}{12}[(\alpha^2-A)f(t_{k+1})-\alpha f'(t_{k+1})+f''(t_{k+1})], \\ 1\le k\le N-1, \\ u_0=\varphi, \\ \left(I+\dfrac{\tau^2}{2}A\right)\dfrac{u_1-u_0}{\tau}=\left(I+\dfrac{\tau^2}{2}A\right)\psi+\dfrac{\tau}{2}(-\alpha\psi-A\varphi) \\ +\dfrac{\tau^2}{6}(\alpha^2\psi+\alpha A\varphi-A\psi)+f_0, \\ f_0=\dfrac{\tau}{2}f(0)-\dfrac{\tau^2}{6}[f(0)-f'(0)]. \end{cases}$$

was developed in [7] in 2016. Stability inequalities for the solution of this third order of accuracy difference scheme were also done. In the following section, we consider one dimensional telegraph equation as a test problem to support theoretical results.

III. Numerical Solutions

For MATLAB implementations, the following initial-boundary value problem is considered:

$$\begin{cases} \dfrac{\partial^2u(t,x)}{\partial t^2}+2\dfrac{\partial u(t,x)}{\partial t}+u(t,x) \\ =\dfrac{\partial^2u(t,x)}{\partial x^2}+2(1+t+t^2)\sin x, \\ 0<t<1,\ \ 0<x<\pi, \\ u(0,x)=\sin x\,, u_t(o,x)=-\sin x\,,\ \ 0\le x\le\pi, \\ u(t,0)=\ u(t,\pi)=0,\ \ 0\le t\le 1. \end{cases} \quad (2)$$

The exact solution of the problem (2) is $u(t,x)=(1-t+t^2)\sin x$. When the step number for time and space variable are taken as 5, the difference between exact solution and approximate solution does not appear as shown from figures 3.1 and 3.2-3.5.

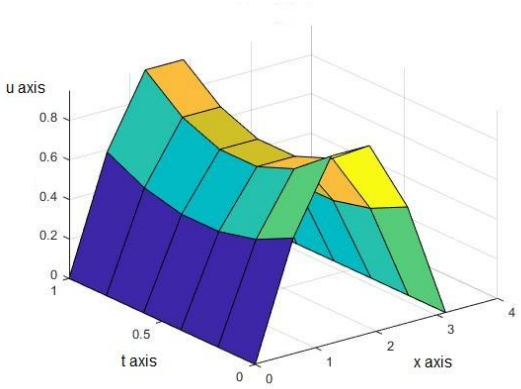

Fig. 3.1. Exact Solution

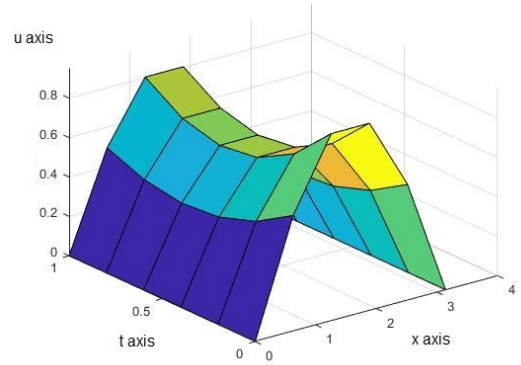

Fig. 3.2. First-order

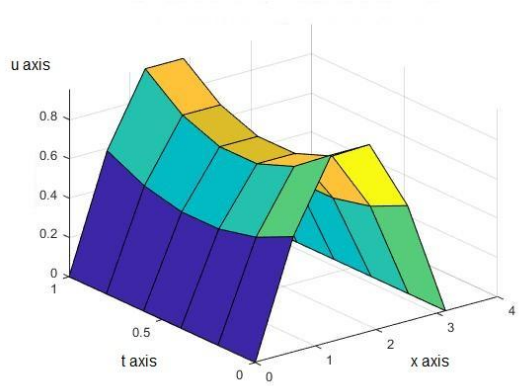

Fig. 3.3. Second-order I

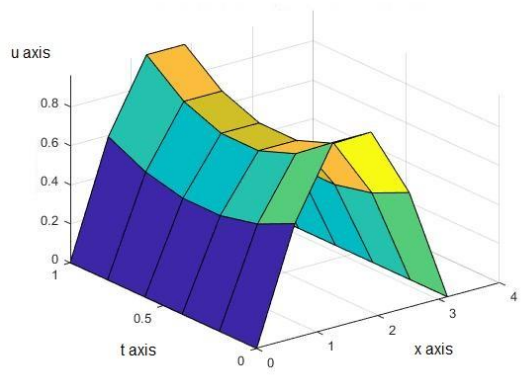

Fig. 3.4. Second-order II

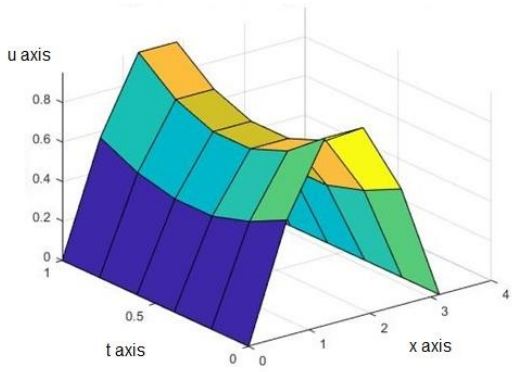

Fig. 3.5. Third-order

To compare the results between exact solution and approximate solution, we need to give the results by table. Error formula is define as follows

$$E_0 = \max_{1\le k\le N-1}\left(\sum_{n=1}^{M-1}|u(t_k,x_n)-u_n^k|^2\ h\right)^{\frac{1}{2}},$$

where $u(t_k,x_n)$ is exact solution and u_n^k is approximate solution at the point (t_k,x_n). h is step size for space variable. N and M are step numbers for time and space variables respectively. As shown in Table 3.1, the accuracy of second order difference schemes are better than the first order difference scheme. The accuracy of the third order difference scheme is better than the accuracy of second order difference schemes. There is no much difference between second order difference scheme 1 and 2. When the number of step for time variable is increased 2 times, the error gets smaller 2 time for first order difference scheme. When the number of step for time variable is increased 2 times, the errors get smaller 4 and 8 times for the second and third order different schemes respectively.

Table 3.1. Error for E_0

	N=M²	64, 8	128, 11	256, 16
FO	Error	0.01069	0.00523	0.00265
	N=M	32, 32	64, 64	128, 128
SO1	Error	0.0002349	0.0000575	0.0000142
SO2	Error	0.0003417	0.0000848	0.0000211
	N³=M²	10, 32	20, 89	40, 253
TO	Error	0.0003249	0.0000381	0.0000046
	N=M	32	64	128
FO	Error	0.0271183	0.0137000	0.0068864
SO1	Error	0.0002349	0.0000575	0.0000142
SO2	Error	0.0003417	0.0000848	0.0000211
TO	Error	0.0001813	0.0000482	0.0000124

IV. NEURAL NETWORKS

Artificial neural networks are computational models that mimic the synaptic structure of neurons in the human brain [8]. These networks consist of processing units (nodes) interconnected across multiple layers: the input layer, one or more hidden layers, and the output layer. The input layer receives raw data, and the hidden layers process this data and extract representations of increasing levels of abstraction, leading to the output layer that produces the final result. These neural networks have demonstrated tremendous potential in approximating any arbitrary function (Universal Approximation Theorem) and learning complex patterns in images, audio, and text, leading to significant scientific advances in fields such as computer vision and natural language processing [9].

With the development of deep learning, the need has emerged to incorporate physical knowledge into the training process to overcome total reliance on observed data, especially in problems where sufficient data is difficult to obtain. Hence, the idea of Physics-Informed Neural Networks (PINNs) ([10]) emerged, adding a term to the loss function that represents the residuals of the partial differential equations governing the system. Leveraging automatic differentiation, the network computes the necessary derivatives of the resulting solution to ensure the network satisfies initial and boundary conditions as well as the physical conservation laws during learning. This enables it to solve forward (finding a solution) and inverse (discovering unknown coefficients or parameters) problems efficiently without the need to generate an explicit mesh or mesh nodes [11].

4.1. Methodology

To build PINNs, we have the following steps.

4.1.1. Problem Formulation

Define the partial differential equation

$$f[u(x,t)] = 0$$

with boundary conditions

$$u(x_b,t_b) = g(x_b,t_b)$$

and with initial conditions

$$u(x_i,0) = h(x_i)$$

4.1.2. Neural Network Architecture

A fully connected feed-forward network $u_\theta(x,t)$ is chosen, which receives the spatial variables x and temporal variables t as inputs and produces the approximate solution u.

4.1.3. Loss Function

Weighted loss function is defined as the following

$$\mathcal{L}(\theta) = w_{PDE}\mathcal{L}_{PDE} + w_{BC}\mathcal{L}_{BC} + w_{IC}\mathcal{L}_{IC}$$

$$\mathcal{L}(\theta) = w_{PDE} *$$

$$\frac{1}{N_r}\sum_{j=1}^{N_r}\left(f[\theta](x_j,t_j)\right)^2 + w_{BC}\frac{1}{N_b}\sum_{k=1}^{N_b}(u_\theta(x_{b,k},t_{b,k}) -$$

$$g(x_{b,k},t_{b,k}))^2 + w_{IC}\frac{1}{N_l}\sum_{i=1}^{N_l}(u_\theta(x_{i,l},t_{i,l}) - h(x_{i,l}))^2\ ,$$

where $\mathcal{L}_{PDE}$ is mean squared residual of the partial differential equations over N_r interior collocation points, $\mathcal{L}_{BC}$ is mean squared error on N_b boundary points against boundary data g, $\mathcal{L}_{IC}$ is mean squared error on N_l initial points against initial data h.

4.1.4. Collocation Point Generation

$\{(x_j, t_j)\} \subset \Omega$ is interior points for the partial differential equations residual. Boundary points $\{(x_{b,k}, t_{b,k})\} \subset \partial\Omega$ for the boundary conditions. Initial points $\{(x_{i,l}, t_{i,l} = t_0)\} \subset \Omega$ at the initial time t_0, then impose

$$u(x_{i,l}, t_0) = h(x_{i,l}) \text{ and } u_t(x_{i,l}, t_0) = h_t(x_{i,l}).$$

4.1.5. Training Loop

Forward pass is passing input through the network layers to get the prediction compute loss. Backward pass is auto-differentiation is used to calculate the gradients of the loss function with respect to the weights, and the optimizer then updates the weights.

4.2. Practical Implementation

We seek to solve the problem (2). We approximated $u(x,t)$ by a dense feed-forward network $u_\theta(x,t)$ taking (x,t) input, passing through hidden layers of different widths with tanh activations, and outputting a single scalar. Its loss is a weighted sum $\mathcal{L}(\theta) = \mathcal{L}_{PDE} + 5.\mathcal{L}_{BC} + \mathcal{L}_{IC}$, where $w_{PDE} = 1, w_{BC} = 5, w_{IC} = 1$ and $\mathcal{L}_{PDE}$ is the mean squared residual of the partial differential equation over 100*100 (and over 200*100) interior collocation points, $\mathcal{L}_{IC}$ enforces $u(x,0) = sinx \; and \; u_t(x,0) = -sinx$ over 200 initial points, and $\mathcal{L}_{BC}$ enforce $u(0,t) = u(\pi,t) = 0$ over 200 boundary points.

We trained the neural network for 10000 epochs using the Adam optimizer (learning rate is 0.01), computing all derivatives via automatic differentiation and checkpointing the model state that achieves the lowest MSE on the training set. Finally, we evaluated the saved model on independent grids of size 100×100 $(200 * 100)$, reporting mean square error and Max Δt-weighted L_2 over x error. Max Δt-weighted L_2over x is

$$\max_{i=1,\dots,N_x} \sqrt{\Delta t \sum_{j=1}^{N_t} (u_{pred}(x_i, t_j) - u_{true}(x_i, t_j))^2} \text{ where } \Delta t = \frac{1-0}{N_t - 1} \text{ is the uniform time-step on [0,1].}$$

Table 4.2.1. Training and Evaluation Performance Metrics of Different PINN Architectures after 10,000 Epochs on a 100×100 Grid

10000 epochs & $N_x * N_t = 100 * 100$				
Performance Metrics / Architecture	**Best Train MSE**	**Best iteration**	**Evaluation Grid $100 * 100$**	
			MSE	**Max Δt-weighted L_2 over x**
[2-50-50-50-1]	3.093e-07	9775	2.894e-07	1.073e-03
[2-50-50-50-50-1]	9.979e-07	4416	8.734e-07	2.471e-03
[2-50-50-50-50-50-1]	2.153e-06	2695	2.089e-06	3.028e-03
[2-100-100-100-1]	2.559e-07	4488	2.299e-07	9.902e-04
[2-32-64-128-64-32-1]	1.014e-05	1180	9.915e-06	5.039e-03

Table 4.2.2. Training and Evaluation Performance Metrics of Different PINN Architectures after 10,000 Epochs on a 200×100 Grid

10000 epochs & $N_x * N_t = 200 * 100$				
Performance Metrics / Architecture	**Best Train MSE**	**Best iteration**	**Evaluation Grid $200 * 100$**	
			MSE	**Max Δt-weighted L_2 over x**
[2-50-50-50-1]	3.574e-07	7647	3.379e-07	1.489e-03
[2-50-50-50-50-1]	6.883e-07	9006	6.680e-07	1.598e-03
[2-50-50-50-50-50-1]	1.036e-06	3999	1.005e-06	1.979e-03
[2-100-100-100-1]	2.767e-07	3065	2.560e-07	1.354e-03
[2-32-64-128-64-32-1]	5.245e-06	1863	5.103e-06	4.180e-03

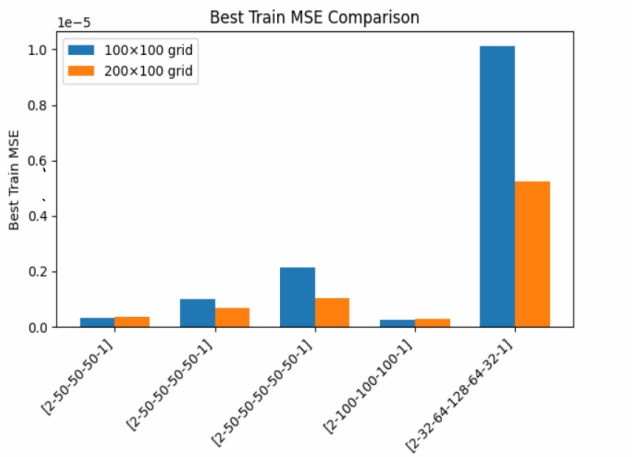

Fig. 4.2.1. Best Train MSE Comparison

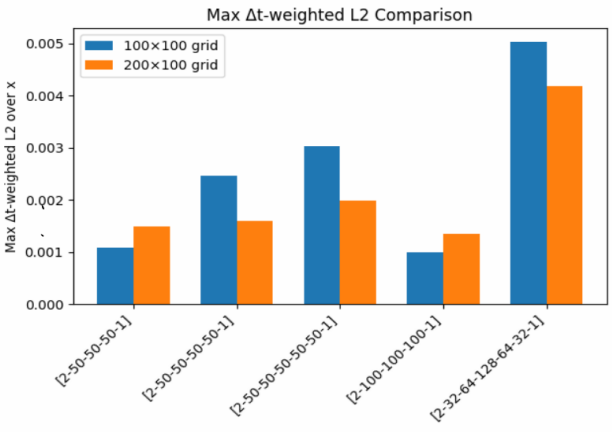

Fig. 4,2.2. Max Δt-weighted L_2 Comparison

4.3. Discussion of PINNs Results

The architecture with five hidden layers [2-50-50-50-50-1] reached the lowest mean square error at iterations 2695 (for a 100×100 grid) and 3999 (for a 200×100 grid), meaning it finished improving the loss earlier than the other architectures. However, the solution quality (Best Train mean square error and Max Δt-weighted L_2 over x) was the worst among all architectures, indicating that it reached an early local minimum without reaching a truly low error. The widest-

deepest hierarchical architecture [2-32-64-128-64-32-1] converges the fastest. This architecture reached its best point at iterations 1180 (100×100) and 1863 (200×100), significantly faster than the other networks. However, it also achieved the highest errors, demonstrating that convergence speed alone is not a quality measure.

The medium-depth and wide-width [2-100-100-100-1] networks show a good balance. They reached their best point at around 4500 iterations (4488 and 3065), significantly faster than the simple [2-50-50-50-1] networks (9775 and 7647) but better performing than the very deep networks. They achieved the lowest errors, reflecting a balance between convergence speed and solution quality. Relatively narrow [2-50-50-50-1] networks converge slowly. They require almost the full number of iterations (9,775 and 7,647 out of 10,000) to achieve the optimal loss. However, they achieve a low error, rivaling [2-100-100-100-1] and outperforming some other architectures. The results also highlight the inherent challenge of finding neural network architectures where determining the optimal architecture requires numerous experimental trials.

V. Conclusion

In this study, one dimensional telegraph equation is considered. Numerical solutions of this equation found by using first, second and third order difference schemes. It was shown that the accuracy of third order scheme is better than second order scheme which is better than first order scheme. There was no much difference between two second order difference schemes. Numerical solutions of the equation were also found by using neural network. The results are satisfactory.

References

[1] A. Ashyralyev, M. E. Koksal, A Numerical Solution of Wave Equation Arising in Non-Homogeneous Cylindrical Shells, *Turkish Journal of Mathematics,* 32 (4) 407-419, 2008

[2] M. E. Koksal, An Operator-Difference Method for Telegraph Equations Arising in Transmission Lines, *Discrete Dynamics in Nature and Society,* 1-17, 2011.

[3] M. E. Koksal, Time and frequency responses of non-integer order RLC circuits, *AIMS Mathematics,* 4 (1) 61-75, 2019.

[4] G. S. Krein, Linear Differential Equations in a Banach Space, *Birkhauser*, 1966.

[5] O. H. Fattorini, Second Order Linear Differential Equations in Banach Space, *Mathematics Studies*, 107, 1985.

[6] A. Ashyralyev, M. Modanli, An operator method for telegraph partial differential and difference equations, *Boundary Value Problems,* Artical ID;41, 1-17, 2015.

[7] A. Ashyralyev, M. E. Koksal, K. T. Turkcan. Numerical solutions of telegraph equations with the dirichlet boundary condition, *International Conference on Analysis and Applied Mathematics,* 1759;1, 1-6, 2016.

[8] J. Schmidhuber, ‘‘Deep learning in neural networks: An overview’’ *Neural Networks, 61*, 85–117, 2015.

[9] Y. LeCun, Y. Bengio & G. Hinton, Deep learning, *nature*, *521*(7553), 436-444, 2015.

[10] V. J. da Cunha Farias, M. B. Siqueira, Physics-Informed Machine Learning for Numerical Solution of Hyperbolic Partial Differential Equations: An Application to the Second Order One Dimensional Linear and Nonlinear Telegraph Equation, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4778353, 1-15, 2024.

[11] 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 *Journal of Computational Physics, 378*, 686–707, 2019.