Predicting traction return current in electric railway systems through physics-informed neural networks

Kapoor, Taniya; Wang, Hongrui ; Nunez, Alfredo; Dollevoet, Rolf

DOI
10.1109/SSCI51031.2022.10022290

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the 2022 IEEE Symposium Series on Computational Intelligence (SSCI)

Predicting traction return current in electric railway systems through physics-informed neural networks

Taniya Kapoor*, Hongrui Wang†, Alfredo Núñez‡ and Rolf Dollevoet§
Section of Railway Engineering, Department of Engineering Structures, Delft University of Technology
Delft, The Netherlands
Email: *t.kapoor@tudelft.nl, †h.wang-8@tudelft.nl, ‡a.a.nunezvicencio@tudelft.nl, §r.p.b.j.dollevoet@tudelft.nl

2022 IEEE Symposium Series on Computational Intelligence (SSCI) | 978-1-6654-8768-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/SSCI51031.2022.10022290

***Abstract*—This paper addresses the problem of determining the distribution of the return current in electric railway traction systems. The dynamics of traction return current are simulated in all three space dimensions by informing the neural networks with the Partial Differential Equations (PDEs) known as telegraph equations. In addition, this work proposes a method of choosing optimal activation functions for training the physics-informed neural network to solve higher-dimensional PDEs. We propose a Monte Carlo based framework to choose the activation function in lower dimensions, mitigating the need for ensemble training in higher dimensions. To further strengthen the applicability of the Monte Carlo based framework, experiments are presented under two loss functions governed by L^2 and L^∞ norms. The presented method efficiently simulates the traction return current for electric railway systems, even for three-dimensional problems.**

***Index Terms*—Traction return current, electric railway systems, physics-informed neural networks, Monte Carlo, activation functions.**

I. Introduction

Locomotives operating on an electric traction system are the backbone of modern railway transportation infrastructure [1]. The electric railway traction system consists of the path between the power station/substation and the locomotive, as shown in Fig. 1. These locomotives receive the power in the form of alternating current carried by overhead cables known as catenaries connected to the substations [2]. The alternating current is introduced in the locomotive through a pantograph. This alternating current is then converted to direct current, which is ultimately used by the locomotive. Additionally, as any electrical system should have a closed path for the current to flow, the grounded railway tracks provide a path for the traction return current. However, the current received back by the substation is less than the current sent to the locomotive indicating losses in the traction return current [3], and [4].

The incurred electric losses extensively affect the amount of electricity that needs to be produced, subsequently affecting the overall cost of operating railways. Electric losses need to be estimated precisely to have control over the costs. These losses necessitate predicting the dynamics of traction return current. In practice, one method to estimate these losses is through carrying out experiments on the prototypes [5]. These prototypes are generally smaller-scale experimental setups built to compute the traction return current. However, prototypes are costly, and it is difficult to capture all the possible dynamics and local conditions that hundreds of kilometres of

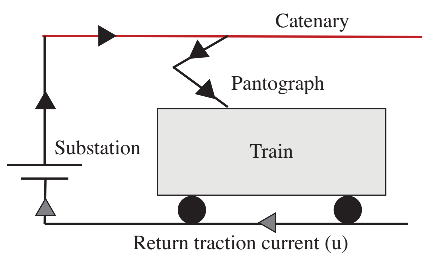

Fig. 1. Electric traction system for railways

traction current systems have in real-life [6]. Mathematically, the traction return current dynamics could be modelled as a Partial Differential Equation (PDE) known as a telegraph equation [4], which could be solved to predict the return current at any instant of time.

In general, from designing an aircraft (Euler equations) to studying supernovas in astrophysics (Magnetohydrodynamic equations), PDEs find their place in many scientific models. The wide variety of applications that PDEs possess makes solving and finding their solutions necessary. Analytical solutions are desirable but seldom found for PDEs modelling real-world phenomena. Therefore, numerical approximations of the solution of PDEs is a very well-researched field. Despite the tremendous progress in numerical methods such as finite difference, finite element, finite volume, and spectral methods, one still faces several challenges such as quantifying uncertainties and solving multiscale and multiphysics problems [7], [8], and [9]. Most numerical methods require a mesh for computation. Mesh generation is tedious and worsens the problem for complicated geometries and higher-dimensional problems [7].

In recent years, there has been rapid progress in scientific machine learning, which fuses scientific computing and machine learning approaches to approximate the solutions of PDEs. State-of-the-art developments in scientific machine learning have been covered in detail in the review paper [10] and references therein. One method for numerically approximating PDEs using Deep Neural Networks (DNNs) is to use representation formulas like the Feynman-Kac formula for parabolic PDEs [11]–[13], whose compositional structure is used in DNNs to approximate the solution. These methods have been applied in various situations, including approximating very high dimensional problems in mathematical finance [11]. Another class of methods tries to improve existing numerical approaches by incorporating deep learning modules, such as learning parameters of numerical schemes from data as presented in [14], and [15] among others.

Provided sufficient data, DNNs can approximate any continuous function, even measurable functions, according to the universal approximation theorem [16]. Machine learning has emerged as a viable alternative to the aforementioned numerical methods. Still, it requires a large amount of data, possibly expensive and noisy in the case of some biological and engineering systems. One possible way to avoid this problem is to collocate the PDE residual at training points, similar to leveraging the physical equation in the training process. References [17], and [18] proposed this method for the first time. However, starting with [19], it has been revived and expanded in much greater detail in the pioneering work of Karniadakis and collaborators. The underlying neural networks are Physics Informed Neural Networks (PINNs).

PINNs are universal function approximators that embed the physical equations described by PDEs along with the given data-set in the learning process. Prior knowledge of physical principles acts as a regularization agent in the training of DNNs, limiting the space of admissible solutions and raising the accuracy of the function approximation. Embedding this prior knowledge into neural networks improves the information content of the given data, making it easier for the learning algorithm to capture the correct solution and generalize well even with a small number of training samples. As a result, PINN is used to find a high-fidelity optimal solution using some knowledge of the physical properties of the problem and some form of training data (even sparse and partial). In a brief period, PINN has already proven to be a very effective paradigm for approximating solutions of partial differential equations, a very partial list of references include [7], [19]–[21] and references therein.

Developing methods for solving high-dimensional partial differential equations has been challenging for a long time [11]. PINNs is a deep learning-based technique that possibly removes the curse of dimensionality and approximates the solution for high-dimensional PDEs [22], provided the selection of the optimal activation functions in higher dimensions. In practice, ensemble training is carried out [22], considering a range of values of the hyperparameters, for instance, the number of hidden layers, neurons, residual parameters, and activation functions, among others. For each configuration in the ensemble, the resulting model is retrained several times with different random starting values of the trainable weights in the optimization algorithm. The one yielding the smallest testing error for the choice of metric is selected. Although, very successful in practice, this method is computationally expensive, particularly in higher dimensions.

This work aims to approximate the traction return current for electrified railway systems in several space dimensions through physics-informed neural networks. Training the neural network for lower dimensions is computationally cheaper than training it in higher dimensions. We propose a Monte Carlo based method to choose activation functions for high-dimensional PDEs through training the neural networks for lower-dimensional PDEs only. Additionally, the proposed technique is examined on two loss functions evaluating its dependency on different norms. To the authors' best knowledge, the contributions of the current paper are as follows,

- We propose estimating the traction return current for electric railways systems through physics-informed neural networks.
- We propose a Monte-Carlo based framework to optimally choose an activation function for training the neural networks for higher-dimensional PDEs.
- Experiments are presented for all three dimensions on two different loss functions governed by L^2 and L^∞ norms.

The rest of the manuscript is organized as follows. In section II, the model of the traction return current is described. Section III presents the Monte Carlo based framework for choosing the activation functions. Section IV contains the numerical experiments. The paper concludes with sections V and VI presenting the discussions and conclusions.

II. Hyperbolic Telegraph Equation

The telegraph equation is a time-dependent scalar equation with many real-world applications, including the propagation of traction return current in the transmission lines. In electrical engineering, the telegraph equation describes the spatio-temporal features of the travelling wave on a long transmission line [23]. Furthermore, depending on the parameters of the telegraph equation, the model can describe the parallel flow of viscous Maxwell's fluids [24] or propagation of acoustic waves in Darcy-type porous media [25].

The physical model of the traction return current is discussed for our work. The study uses a multiconductor transmission line model of the traction line [3]. The model for this paper is taken from [3], where a system of PDEs is presented for the telegraph equations. However, the system of PDEs could be coupled to form a single PDE as presented in [26] with traction return current as the unknown. We consider our numerical experiments on a generic transmission line with simplified coefficient values.

The resulting telegraph equation is a time-dependent scalar equation. The general form of second order hyperbolic telegraph equation [26], [27] is considered as a test problem in n space dimensions with space time domain $(\mathbf{x}, t) \in D \times T$, where $D = [0,1]^n \subset \mathbb{R}^n$, and $T \subset \mathbb{R}$. The spatial boundary is denoted by Γ and initial temporal boundary is denoted by Γ_0,

$$\mathcal{F} := u_{tt} - \Delta u + 2\alpha u_t + \beta^2 u - f(\mathbf{x}, t) \tag{1}$$

with initial and Dirichlet boundary conditions,

$$\begin{aligned} u(\mathbf{x}_\mathrm{i}, 0) &= h_1(\mathbf{x}_\mathrm{i}) \quad && \forall (\mathbf{x}_\mathrm{i}, t) \in D \times \Gamma_0 \\ u_t(\mathbf{x}_\mathrm{i}, 0) &= h_2(\mathbf{x}_\mathrm{i}) \quad && \forall (\mathbf{x}_\mathrm{i}, t) \in D \times \Gamma_0 \\ u(\mathbf{x}_\mathrm{b}, t) &= g(t) \quad && \forall (\mathbf{x}_\mathrm{b}, t) \in \Gamma \times T \end{aligned} \tag{2}$$

Here, $h_1(\mathbf{x}_\mathrm{i})$ and $h_2(\mathbf{x}_\mathrm{i})$ are the initial conditions, and $g(t)$ is the boundary condition applied on the model. $\mathbf{x}_\mathrm{b}$ and $\mathbf{x}_\mathrm{i}$ are the boundary and initial points respectively. We assume the quantity of interest $u(\mathbf{x}, t)$ is the only unknown in the model which is the traction return current. Numerical experiments in all possible physical dimensions are presented, which is $n \in \{1, 2, 3\}$. The parameters α and β are chosen to be 1. The forcing term $f(\mathbf{x}, t)$ is stated along with the numerical experiments.

III. Method: Physics-Informed Neural Network

The framework of PINNs is based on DNN. In DNN, features map to labels through an iterative composition of hidden layers. The composition consists of weights, biases, and linear or non-linear activation function(s). In a standard supervised learning paradigm, one needs labelled data to minimize the difference between predicted solution u^* and label u to find the optimal parameters (weights and biases) through a suitable optimization algorithm. In the case of forward partial differential equations, the only labelled data available is in the form of initial and boundary conditions. A priori, there is no solution available in the whole domain $D \times T$, making the data at the interior points unlabelled.

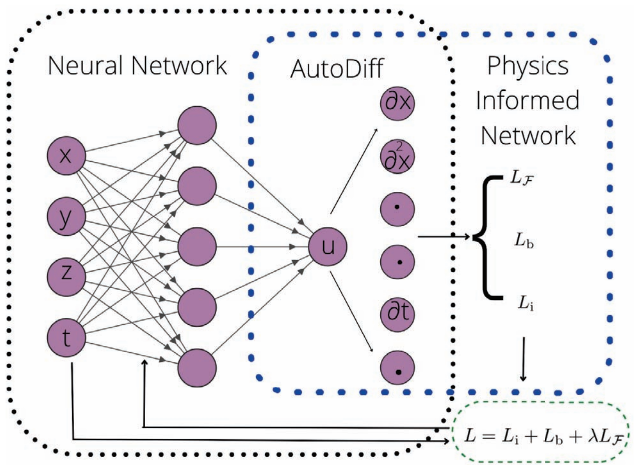

Fig. 2. Physics-informed neural network architecture.

The novel approach of PINNs does not need output data in the whole domain and takes the leverage of auto-differentiation and back-propagation in neural networks. In the loss function of PINNs, for the interior domain, instead of the labelled output u, one considers the physical equation along with the initial and boundary conditions, making PINNs a kind of semi-supervised technique as shown in Fig. 2.

1) Training Set: To train the neural network, one needs training set (Δ), consisting of spatial boundary points (Δ_b), temporal boundary points (Δ_i) and interior points (Δ_int). Hence, training set can be written as $\Delta = \Delta_\mathrm{i} \cup \Delta_\mathrm{b} \cup \Delta_\mathrm{int}$. In this work, the number of training points in Δ_i, Δ_b, and Δ_int are denoted by N_i, N_b and N_int respectively.

2) Loss function: We aim to obtain the approximate solution of unknown quantity u by minimizing the loss function L defined as,

$$L = L_\mathrm{i} + L_\mathrm{b} + \lambda L_\mathcal{F} \tag{3}$$

where,

$$\begin{aligned} L_\mathcal{F} &= \frac{1}{N_\mathrm{int}} \sum_{n=1}^{N_\mathrm{int}} |\mathcal{F}^n|^p \\ L_\mathrm{b} &= \frac{1}{N_\mathrm{b}} \sum_{n=1}^{N_\mathrm{b}} |g^n(\mathbf{x}_\mathrm{b}, t) - g^*|^p \\ L_\mathrm{i} &= \frac{1}{N_\mathrm{i}} \sum_{n=1}^{N_i} (|u^n(\mathbf{x}_\mathrm{i}, 0) - h_1^*|^p + |u_t^n(\mathbf{x}_\mathrm{i}, 0) - h_2^*|^p) \end{aligned} \tag{4}$$

The term $L_\mathcal{F}$ corresponds to the physical equation, L_b and L_i relate to the boundary and initial data; g^* is the PINN prediction at $(\mathbf{x}_\mathrm{b}, t)$ and h_1^*, h_2^* are the PINN prediction and its derivative at $(\mathbf{x}_\mathrm{i}, 0)$, and λ is the residual parameter in the loss function.

The model is trained with two norm possibilities on the loss function for the numerical experiments. The first norm is the most commonly used loss function in the PINN literature, for $p = 2$, in (4) popularly known as the mean squared error (MSE). Second-order optimizer L-BFGS is used to find the optimal parameters to minimize the loss function.

In addition, one more possibility to train the neural network using L^∞ norm on the loss function is explored. In this case, the terms in the loss function take the following form,

$$\begin{aligned} L_\mathcal{F} &= \max_{1 \le n \le N_\mathrm{int}} (|\mathcal{F}^n|) \\ L_\mathrm{b} &= \max_{1 \le n \le N_\mathrm{b}} (|g^n(\mathbf{x}_\mathrm{b}, t_\mathrm{b}) - g^*|) \\ L_\mathrm{i} &= \max_{1 \le n \le N_\mathrm{i}} (|u^n(\mathbf{x}_\mathrm{i}, 0) - h_1^*| + |u_t^n(\mathbf{x}_\mathrm{i}, 0) - h_2^*|) \end{aligned} \tag{5}$$

To minimize the maximum absolute loss function, first-order optimizer ADAM is used to find the optimal solution u. In addition to predicting the traction return current through the PINN algorithm, an activation function is also sought, which performs optimally in several space dimensions. We propose implementing the PINN algorithm several times in lower dimensions with distinct choices of activation functions and storing the error each time. Since the inputs of the problem, that is, the collocation points in the computational domain, are generated randomly for different retrainings. Hence, we obtain a different result for each run, even if all hyperparameters are kept the same. Similarly, for each choice of activation function, the PINN algorithm is run several times, keeping all other hyperparameters the same. Finally, to choose the optimal activation function for training in higher dimensions, the mean of the obtained errors for each re-training in 1D is computed, which is the Monte Carlo algorithm, and the activation function resulting in the least error is chosen. The algorithm for a single PINN run could be described in brief as follows,

Algorithm

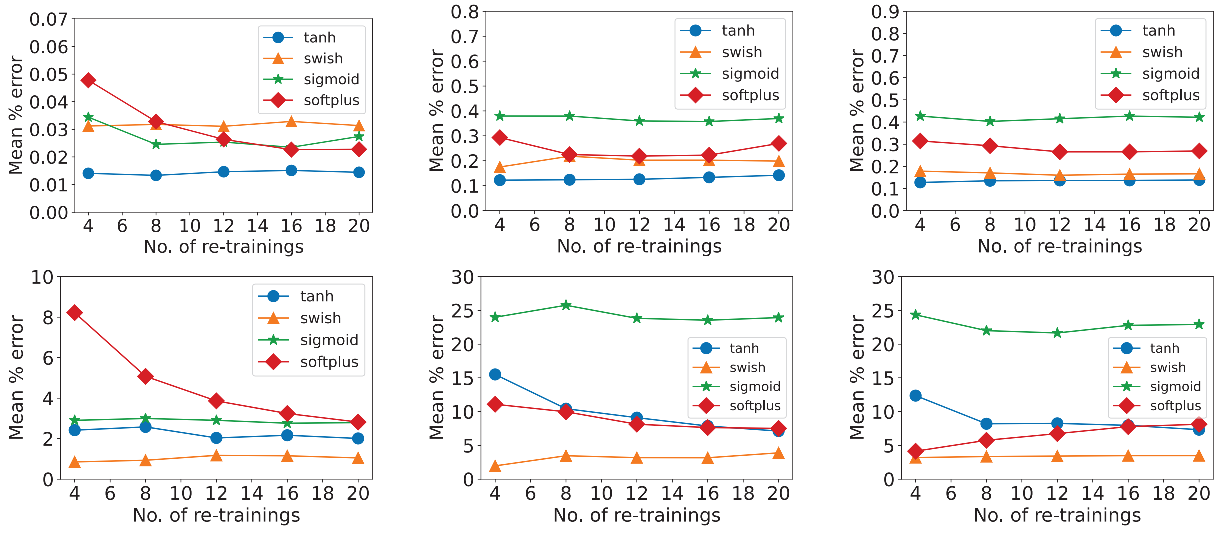

Fig. 3. Mean $\mathcal{R}$ vs number of re-trainings for all four activation functions $\mathbf{top}$: L^2 training $\mathbf{Left(a)}$: one dimension; $\mathbf{Mid(b)}$: two dimension; $\mathbf{Right(c)}$: three dimension. $\mathbf{bottom}$: L^∞ training $\mathbf{Left(d)}$: one dimension; $\mathbf{Mid(e)}$: two dimension; $\mathbf{Right(f)}$: three dimension.

Goal: To find an approximation of the quantity of interest u.

Step 1: Choose the training set ($\Delta = \Delta_b \cup \Delta_i \cup \Delta_{int}$) from the space-time domain $D \times T$.

Step 2: Construct feedforward deep neural network with inputs $(\mathbf{x}, t)$ and output u.

Step 3: Minimize the loss function (4) and (5) with suitable optimization algorithm and find the optimal parameters (weights and biases).

Step 4: Use the optimal parameters to find the approximate solution u^* at $(x_{\text{test}}, t_{\text{test}}) \in D \times T$.

IV. Numerical Experiments

This section describes the considered neural network architecture, along with the test cases. To be consistent in all dimensions, sufficient number of interior training points ($N_{\text{int}} = 16384$) and initial and boundary training points ($N_b + N_i = 16384$) are provided in both trainings [8]. The training points are generated using low discrepancy Sobol sequences [8]. The considered neural network for the study contains 4 hidden layers and 20 neurons in each hidden layer. The residual parameter λ is chosen to be 0.1 [8]. The choice of activation function plays a crucial role in training neural networks. We consider well-used activation functions in PINN literature (tanh, sigmoid and softplus) along with the swish activation function to observe the behaviour of the predicted solution as the dimensionality increases. For the error metric, relative percentage error $\mathcal{R}$ as used in [8] is chosen.

$$\mathcal{R} = \frac{||u^* - u||_2}{||u||_2} \times 100 \tag{6}$$

For each numerical experiment, 20 re-trainings have been performed. Re-training refers to running the PINN algorithm multiple times, randomizing inputs, weights, and biases resulting in a different $\mathcal{R}$ for each run owing to the numerical optimization algorithm. Table [I]-[VI] present the mean and standard deviations of all 20 re-trainings. 10000 max-iteration has been performed for the MSE loss function as second-order optimizer L-BFGS was used. For the maximum absolute norm loss function optimized by the first-order optimizer ADAM, 20000 epochs were performed.

A. Test Cases

1) One-dimensional telegraph equation: For the first numerical experiment, one space dimensional model is consider in the domain, $D \times T = [0, 1] \times [0, 1]$. The forcing term is assumed to be, $f(x, t) = \cos(t)\sin(x) - 2\sin(t)\sin(x)$, To make the problem well-posed, initial conditions,

$$u(x, 0) = \sin(x), \quad u_t(x, 0) = 0,$$

and Dirichlet boundary conditions are applied as,

$$u(0, t) = 0, \quad u(1, t) = \cos(t)\sin(1).$$

The exact solution for this problem is, $u(x, t) = \cos(t)\sin(x)$ which is used for the calculation of relative error percent ($\mathcal{R}$).

TABLE I

One Dimension	**L^2 training**	
Activation Functions	***Mean***	***Std. Dev***
tanh	1.44×10^{-2}	5.18×10^{-3}
swish	3.13×10^{-2}	1.79×10^{-2}
sigmoid	2.74×10^{-2}	2.41×10^{-2}
softplus	2.28×10^{-2}	1.84×10^{-2}

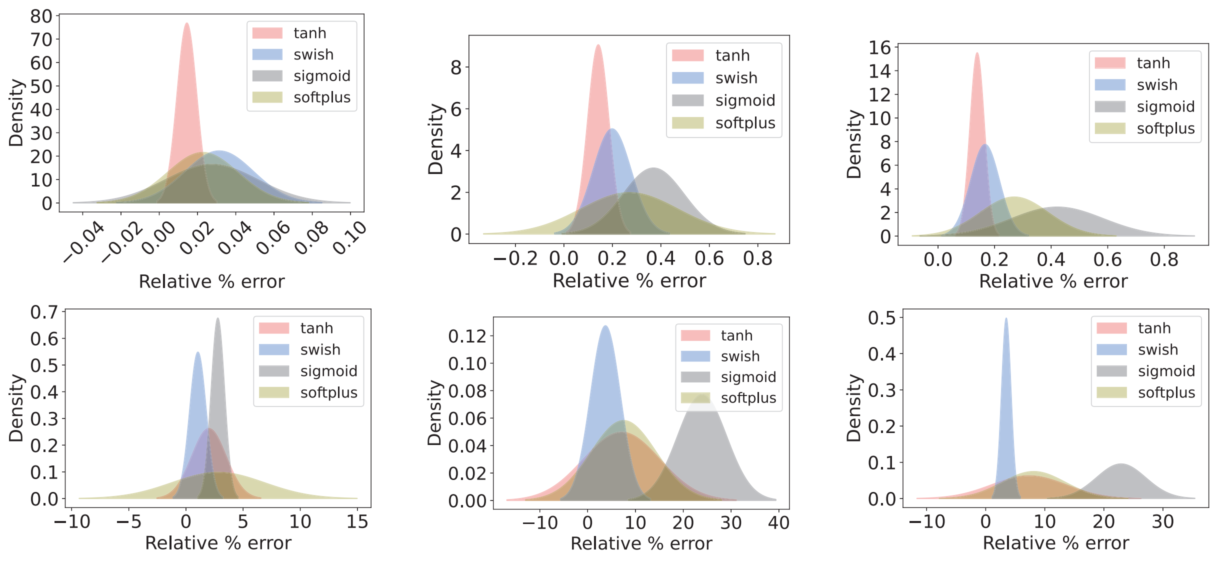

Fig. 4. Distribution plot for all four activation functions **top** : L^2 training **Left**(**a**) : one dimension; **Mid**(**b**) : two dimension; **Right**(**c**) : three dimension. **bottom** : L^∞ training **Left**(**d**) : one dimension; **Mid**(**e**) : two dimension; **Right**(**f**) : three dimension.

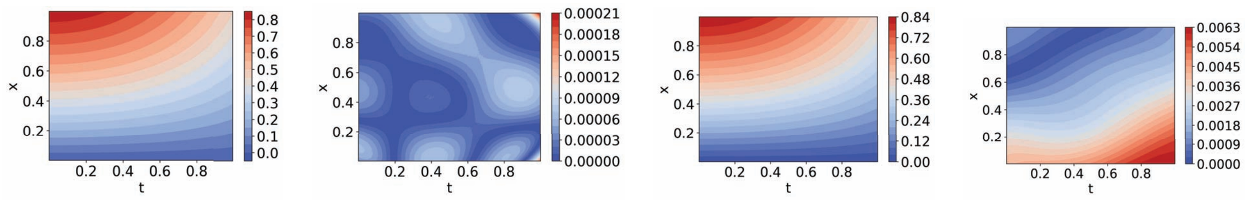

Fig. 5. Test case 1 (**a**) PINN approximation with L^2 training; (**b**) Absolute difference of exact and PINN with L^2 training; (**c**) PINN approximation with L^∞ training; (**d**) Absolute difference of exact and PINN with L^∞ training.

TABLE II

One Dimension	L^∞ training	
Activation Functions	*Mean*	*Std. Dev*
tanh	2.00879766	1.51779322
swish	1.04768208	0.72702900
sigmoid	2.79106137	0.58980483
softplus	2.81767620	4.05375819

TABLE III

Two Dimension	L^2 training	
Activation Functions	*Mean*	*Std. Dev*
tanh	0.14199159	0.04399127
swish	0.19886359	0.07910510
sigmoid	0.36978569	0.12598651
softplus	0.26977729	0.20043524

TABLE IV

Two Dimension	L^∞ training	
Activation Functions	*Mean*	*Std. Dev*
tanh	7.10643780	8.00334820
swish	3.72776193	3.13151448
sigmoid	23.92830481	5.15323669
softplus	7.51133697	6.83848449

2) *Two-dimensional telegraph equation:* Two space dimensional telegraph model (1) is taken as the second test case with the computational domain $D \times T = [0,1]^2 \times [0,1]$. In this case the forcing term is, $f(x,y,t) = 2\cos(t)\sin(x)\sin(y) - 2\sin(t)\sin(x)\sin(y)$. The problem is supplemented with the initial conditions,

$$u(x,0) = \sin(x)\sin(y), \quad u_t(x,0) = 0.$$

and Dirichlet boundary conditions,

$$\begin{aligned} u(x,0,t) &= 0, \quad u(0,y,t) = 0,\\ u(1,y,t) &= \cos(t)\sin(1)\sin(y),\\ u(x,1,t) &= \cos(t)\sin(x)\sin(1). \end{aligned}$$

In this case the analytical solution is $u(x,y,t) = \cos(t)\sin(x)\sin(y)$.

3) *Three-dimensional telegraph equation :* In the final experiment, three space dimensional telegraph equation (1) in the domain $D \times T = [0,1]^3 \times [0,1]$ is considered.

The forcing term taken for this test case is, $f(x, y, z, t)$ = $3\cos(t)\sin(x)\sin(y)\sin(z) - 2\sin(t)\sin(x)\sin(y)\sin(z)$, with the initial conditions,

$$u(x, y, z, 0) = \sin(x)\sin(y)\sin(z), \quad u_t(x, y, z, 0) = 0, \quad (7)$$

and Dirichlet boundary conditions,

$$\begin{aligned}
&u(1, y, z, t) = \sin(1)\sin(y)\sin(z)\cos(t),\\
&u(x, 1, z, t) = \sin(x)\sin(1)\sin(z)\cos(t),\\
&u(x, y, 1, t) = \sin(x)\sin(y)\sin(1)\cos(t),\\
&u(0, y, z, t) = u(x, 0, z, t) = u(x, y, 0, t) = 0.
\end{aligned}$$

TABLE V

Three Dimension	**L^2 training**	
Activation Functions	***Mean***	***Std. Dev***
tanh	0.13847198	0.02568810
swish	0.16615373	0.05135004
sigmoid	0.42155327	0.16187028
softplus	0.26939003	0.12027588

TABLE VI

Three Dimension	**L$^\infty$ training**	
Activation Functions	***Mean***	***Std. Dev***
tanh	7.32334883	6.32465018
swish	3.48192039	0.80039423
sigmoid	22.91893825	4.16512679
softplus	8.12778883	5.33127412

For the calculation of relative error percentage ($\mathcal{R}$), the analytical solution $u(x, y, z, t) = \cos(t)\sin(x)\sin(y)\sin(z)$ is used.

V. Results

Three test cases with two distinct norms on the loss function, making six cases, are considered in this paper. For all these six cases, 20 re-trainings have been performed. The following discussion is based on the statistics of R calculated after 20 re-trainings. As $\mathcal{R}$ could vary on a given run of a PINN algorithm, it is unjustifiable to compare the activation functions based on a random run. Hence, it is necessary to have multiple re-training for each configuration of hyperparameters. Also, the sample size for the hyperparameters should be large enough to avoid outliers. For example, if one ran the experiments with the swish and softplus activation functions only for L^2 training, the conclusion from 1D and 2/3D would be inconsistent, as shown in Fig 3(a-b-c). However, this problem vanishes when a sufficient sample size of four activation functions is chosen. Two subsequent arguments describe the importance of re-training.

First, we illustrate this using the plot of mean $\mathcal{R}$ vs the number of re-trainings as presented in Fig. 3. For instance, in Fig. 3.(a), the mean $\mathcal{R}$ for four re-training of softplus activation function is around 0.05%, which is more than other activation functions, indicating that softplus performs as the worst activation function. However, after 20 re-trainings, the mean $\mathcal{R}$ of softplus decreases nearly to 0.02%, making it the second-best activation function in that case. To further emphasize the importance of 20 re-training, consider Fig. 3.(f). There appears to be a large 10% difference in mean $\mathcal{R}$ after four re-training in approximations using softplus and tanh, but as the re-trainings increase, this difference vanishes. From Fig. 3., it is evident that the mean $\mathcal{R}$ converges to a constant value as re-training increases. This observation indicates that 20 re-trainings are sufficient to compare the performance of activation functions, at least for the considered test cases. An attempt to make inferences and conclusions shall not be made unless an adequate number of re-training for a particular problem has been performed.

Second, to support the importance of 20 re-trainings, it is assumed that the error obtained on each algorithm run is independent and identically distributed. We assume that each $\mathcal{R}$ obtained follows a normal distribution with the mean and standard deviations as mentioned in Table (I-VI). From Fig. 4.(a), it can be inferred that on a random run, there appears to be the slightest chance that the error obtained in the predicted solution with sigmoid activation function will be less than that of tanh. This chance is very low but can not be ignored. Fig. 4.(b)-(f). also indicate that reporting the statistics seems more practical than reporting a random run. Hence, re-training plays a crucial role in ensuring the results and conclusions' reliability.

A discussion of the L^2 training outcomes is now presented. In all three dimensions, the L^2 training proves to provide a very good approximation of the solution, as shown in Tables I, III and V. For all three cases, the mean $\mathcal{R}$ of less than 1% is achieved, which shows PINN successfully removes the curse of dimensionality, as it can be observed that from second dimension to third dimension telegraph equation, the error appears in the same order of magnitude. As it can be observed that from second dimension to third dimension telegraph equation, there is no change in order of magnitude. However, tanh comparatively performed the best in approximating the solution in all dimensions. It is noted that swish gives the maximum mean $\mathcal{R}$ in one dimension, as shown in Table I, but in two and three dimensions (Table III, V), swish performed as the second-best activation function. The performance of softplus and sigmoid showed a gradual fall as the dimensionality increased. Sigmoid proved to be the least performing activation function in approximating the solution in two and three dimensions as the sigmoid-predicted solution exhibits maximum mean $\mathcal{R}$ in those cases. For best performing activation function tanh, Fig. 5., Fig. 6., and Fig. 7. are presented for visualization. The figures are collected at 10000 random testing points in the domain for a random run. Fig. 5.(a), Fig. 6.(a), and Fig. 7.(a) present the PINN predicted solution alongside the exact solution. For a better comparison, the absolute difference between PINN solution and analytical solution is shown in Fig. 5.(b), Fig. 6.(b), and Fig. 7.(b) for L^2 training.

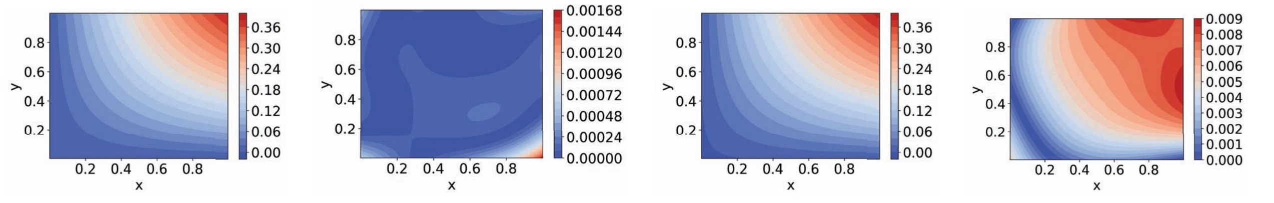

Fig. 6. Test case 2 at time $t = 1$ ($\mathbf{a}$) PINN approximation with L^2 training; ($\mathbf{b}$) Absolute difference of exact and PINN with L^2 training; ($\mathbf{c}$) PINN approximation with L^∞ training; ($\mathbf{d}$) Absolute difference of exact and PINN with L^∞ training.

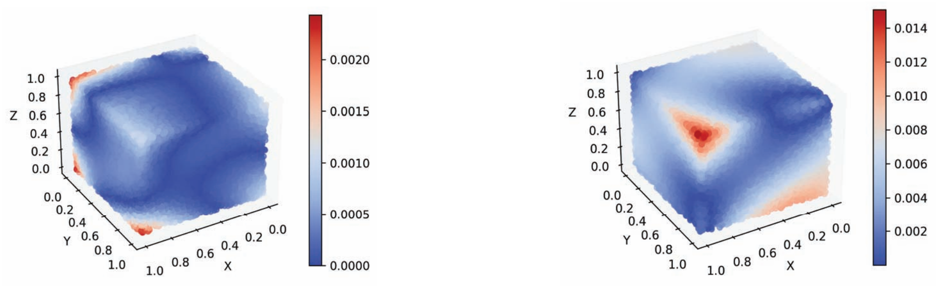

Fig. 7. Test case 3 at time $t = 1$ ($\mathbf{a}$) : Absolute difference of exact and PINN with L^2 training; ($\mathbf{b}$) : Absolute difference of exact and PINN with L^∞ training.

As shown in Tables II, IV, and VI, for L^∞ training, PINN can predict the solution in all three dimensions. The swish activation function in all dimensions obtains the mean $\mathcal{R}$ of less than 4%. It is observed that apart from swish, only tanh and softplus could approximate the solution with under 10% error. As dimensionality rose, softplus and tanh's performance began to deteriorate. Sigmoid proves to be the worst-performing activation function with around 22% mean $\mathcal{R}$ in the case of three-dimension, as shown in Table VI. Fig. 5., Fig. 6., and Fig. 7. are illustrated for the best performing activation function swish. Figures are presented for a random run on the domain's 10000 random testing points. The PINN predicted solution is shown alongside the absolute difference in Fig. 5.(c), Fig. 6.(c), and Fig. 7.(c). For comparison, Fig. 5.(e), Fig. 6.(e), and Fig. 7.(e) for L^∞ training exhibit the absolute difference between PINN and analytical solutions.

It is worth noting that the purpose of this study is not to compare the results obtained by L^2 and L^∞ training. Rather, the aim is to explore the potential of L^∞ training for the transmission line model and provide another option for the training process.

VI. Discussion

The presented numerical experiments show that the physics-informed neural networks efficiently predict the traction return current. On-field experiments for traction return current are cost-intensive due to expensive prototypes. In this work, we leveraged the available knowledge in the form of physical equations to predict the return current at every temporal location for one, two and three space dimensions. Results show that the method successfully addresses the curse of dimensionality by simulating the three-dimensional problem with errors comparable to lower dimensions.

Most railways research is based on numerical and data-based methods. There are several challenges to these methods, such as prototypes, noisy data, predictions only on certain locations, and the optimal location of sensors. This research provides a way forward to leverage the underlying knowledge and use it for further research in railways. The neural network's capacity to efficiently approximate higher dimensional/multivariable functions strengthens the motivation. The universal function approximation theorem guarantees the function mapping for such a problem. However, the conditions under which (neural network configuration, activation function, etc.) the function could be learned are uncertain.

One of such uncertain hyperparameters is the choice of the activation function. For the optimal choice of activation function in the higher dimensional problem, we proposed to use Monte Carlo for low-dimensional problems and then use the obtained activation function in high-dimensional problems. This framework provides a way to mitigate the need to perform ensemble training in higher dimensions, which is computationally expensive. It is also worth noting that this Monte Carlo based framework could be employed to choose other hyperparameters like the neuron and layer composition, among others; however, it is not the scope of this paper. In general, the proposed algorithm works because for linear PDEs like the telegraph equation considered for our study; the dimensionality increase is similar to extrapolating the dynamics in higher dimensions compared to the previous lower dimensions. This knowledge allows us to use the same activation function found in the lower dimensions. However, choosing a hyperparameter on a single run seems non-intuitive as it depends on the seed and differs from system to system, necessitating an algorithm like Monte Carlo, which computes the statistics of multiple re-trainings.

Numerical experiments provided the optimal choice of activation function under two different norms, L^2 and L^∞. Although the L^2 norm is most widely employed to train the neural network for PINNs [10], the L^∞ norm could potentially be of interest in cases where the mathematical theories of convergence and stability for the PDEs under consideration have been developed and established under L^∞ norm. A few prominent examples of such PDEs include fully nonlinear PDEs such as the Hamilton-Jacobi-Bellman and the Monge-Ampere equations, among others [28]. The results of the experiments show that the trend for the optimal choice of activation function using the Monte Carlo method is not dependent on norms. For instance, the activation function found for the L^2 norm remains valid throughout all dimensions. A similar trend is observed for the L^∞ norm as well. The reason for this behaviour comes from the definition of the Monte Carlo algorithm. The Monte Carlo algorithm works on the obtained results after the norm has been applied. Hence, the study offers a general framework for any loss function and norm choice, at least for the return traction current problem modelled by the telegraph equations.

VII. Conclusion

This work proposes employing PINNs, a deep learning approach for predicting traction return current using second-order hyperbolic telegraph equations in several space dimensions. The presented experiments in all three dimensions suggest that PINN approximates the solution accurately and efficiently. In addition, the results lead to the following conclusions.

First, the traction return current for railways could be approximated using only physical equations without the need for additional data. We provided the data only for the initial time and the domain boundaries. The current flow is predicted in the whole space and time domain with minimal error. This clearly shows that the PINN algorithm approximates the solution very well. Even for three dimensions, PINN approximates the solution with under five percent error, removing the curse of dimensionality.

Second, Numerical experiments demonstrate that the optimal activation function found using the Monte Carlo framework in the lower dimension continues to perform the best when training the neural network in higher dimensions. This hyperparameter (activation function) tuning trend is advantageous since tweaking it in lower dimensions is less expensive than tuning it repeatedly for every higher dimension.

Third, The activation function selection is norm-specific, even for the same model. The optimal activation function found for one loss function does not guarantee performance for other norms on the same loss functions. Results show swish is possibly a better activation function than tanh when approximating the traction return current using the telegraph equation. Although swish did not outperform tanh in L^2 training, the results are not bad. Moreover, in the case of L^∞, training swish provides far better results than other activation functions. Finding the reasoning behind such a trend is beyond the scope of this work.

In future, the accurate performance of PINNs in estimating the traction return current can be utilized in the field of protection relay, especially the travelling wave protection. According to the travelling wave value collected from the sensors at the transmission line terminals, we can leverage the physical equation to know where and when a fault occurs on the transmission line. In addition, solving the inverse problem to estimate the parameters of the transmission line could assist in monitoring the health status of the transmission line.

References

[1] T. Abdallah, “Sustainable mass transit: Challenges and opportunities in urban public transportation,” 2017.

[2] Z. Liu, Y. Song, Y. Han, H. Wang, J. Zhang, and Z. Han, “Advances of research on high-speed railway catenary,” *Journal of modern transportation*, vol. 26, no. 1, pp. 1–23, 2018.

[3] A. Mariscotti, P. Pozzobon, and M. Vanti, “Distribution of the traction return current in at electric railway systems,” *IEEE Transactions on Power Delivery*, vol. 20, no. 3, pp. 2119–2128, 2005.

[4] A. Mariscotti, “Distribution of the traction return current in AC and DC electric railway systems,” *IEEE Transactions on power delivery*, vol. 18, no. 4, pp. 1422–1432, 2003.

[5] B. Allotta, L. Pugi, and F. Bartolini, “An active suspension system for railway pantographs: the t2006 prototype,” *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, vol. 223, no. 1, pp. 15–29, 2009.

[6] D. Dujic, A. Mester, T. Chaudhuri, A. Coccia, F. Canales, and J. K. Steinke, “Laboratory scale prototype of a power electronic transformer for traction applications,” in *Proceedings of the 2011 14th European Conference on Power Electronics and Applications*. IEEE, 2011, pp. 1–10.

[7] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, “Physics-informed machine learning,” *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[8] S. Mishra and R. Molinaro, “Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for pdes,” *IMA Journal of Numerical Analysis*, 2021.

[9] Z. Hidayat, R. Babuška, A. Nunez, and B. De Schutter, “Identification of distributed-parameter systems from sparse measurements,” *Applied Mathematical Modelling*, vol. 51, pp. 605–625, 2017.

[10] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, “Scientific machine learning through physics-informed neural networks: Where we are and what’s next,” *arXiv preprint arXiv:2201.05624*, 2022.

[11] C. Beck, E. Weinan, and A. Jentzen, “Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations,” *Journal of Nonlinear Science*, vol. 29, no. 4, pp. 1563–1619, 2019.

[12] T. P. Miyanawala and R. K. Jaiman, “An efficient deep learning technique for the navier-stokes equations: Application to unsteady wake flow dynamics,” *arXiv preprint arXiv:1710.09099*, 2017.

[13] M. Raissi and G. E. Karniadakis, “Hidden physics models: Machine learning of nonlinear partial differential equations,” *Journal of Computational Physics*, vol. 357, pp. 125–141, 2018.

[14] S. Mishra, “A machine learning framework for data driven acceleration of computations of differential equations,” *arXiv preprint arXiv:1807.09519*, 2018.

[15] D. Ray and J. S. Hesthaven, “An artificial neural network as a troubled-cell indicator,” *Journal of computational physics*, vol. 367, pp. 166–191, 2018.

[16] M. H. Hassoun *et al.*, *Fundamentals of artificial neural networks*. MIT press, 1995.

[17] I. E. Lagaris, A. Likas, and D. I. Fotiadis, “Artificial neural networks for solving ordinary and partial differential equations,” *IEEE transactions on neural networks*, vol. 9, no. 5, pp. 987–1000, 1998.

[18] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, "Neural-network methods for boundary value problems with irregular boundaries," *IEEE Transactions on Neural Networks*, vol. 11, no. 5, pp. 1041–1049, 2000.

[19] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112789, 2020.

[20] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: A navier-stokes informed deep learning framework for assimilating flow visualization data," *arXiv preprint arXiv:1808.04327*, 2018.

[21] J. C. Wong, C. Ooi, A. Gupta, and Y.-S. Ong, "Learning in sinusoidal spaces with physics-informed neural networks," *arXiv preprint arXiv:2109.09338*, 2021.

[22] S. Mishra and R. Molinaro, "Physics informed neural networks for simulating radiative transfer," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 270, p. 107705, 2021.

[23] J. Fu, G. Song, and B. De Schutter, "Influence of measurement uncertainty on parameter estimation and fault location for transmission lines," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 1, pp. 337–345, 2020.

[24] G. Böhme, *Non-Newtonian fluid mechanics*. Elsevier, 2012.

[25] H. Pascal, "Pressure wave propagation in a fluid flowing through a porous medium and problems related to interpretation of stoneley's wave attenuation in acoustical well logging," *International journal of engineering science*, vol. 24, no. 9, pp. 1553–1570, 1986.

[26] R. Mohanty, "New unconditionally stable difference schemes for the solution of multi-dimensional telegraphic equations," *International Journal of Computer Mathematics*, vol. 86, no. 12, pp. 2061–2071, 2009.

[27] M. Dehghan and A. Ghesmati, "Combination of meshless local weak and strong (mlws) forms to solve the two dimensional hyperbolic telegraph equation," *Engineering Analysis with Boundary Elements*, vol. 34, no. 4, pp. 324–336, 2010.

[28] L. C. Evans, *Partial differential equations*. American Mathematical Soc., 2010, vol. 19.