Replacing the FitzHugh-Nagumo electrophysiology model by physics-informed neural networks

Yan Barbosa Werneck[1], Rodrigo Weber dos Santos[0000−0002−0633−1391][1], Bernardo Martins Rocha[0000−0002−0508−8959][1], and Rafael Sachetto Oliveira[0000−0003−0800−5984][2]

[1] Graduate Program in Computational Modeling, Federal University of Juiz de Fora, Juiz de Fora, Brazil

[2] Graduate Program in Computer Science, Federal University of São Jõao del Rei, São Jão del Rei, Brazil

Abstract. This paper presents a novel approach to replace the FitzHugh-Nagumo (FHN) model with physics-informed neural networks (PINNs). The FHN model, a system of two ordinary differential equations, is widely used in electrophysiology and neurophysiology to simulate cell action potentials. However, in tasks such as whole-organ electrophysiology modeling and drug testing, the numerical solution of millions of cell models is computationally expensive. To address this, we propose using PINNs to accurately approximate the two variables of the FHN model at any time instant, any initial condition, and a wide range of parameters. In particular, this eliminates the need for causality after training. We employed time window marching and increased point cloud density on transition regions to improve the training of the neural network due to nonlinearity, sharp transitions, unstable equilibrium, and bifurcations of parameters. The PINNs were generated using NVIDIA's Modulus framework, allowing efficient deployment on modern GPUs. Our results show that the generated PINNs could reproduce FHN solutions with average numerical errors below 0.5%, making them a promising lightweight computational model for electrophysiology and neurophysiology research.

Keywords: Neurophysiology · Computational Electrophysiology · FitzHugh-Nagumo · Physics-Informed Neural Networks

1 Introduction

Computational electrophysiology and neurophysiology are fields that study the electrical activity of cells and tissues, such as those in the heart or the brain. The Hodgkin-Huxley (HH) model, proposed in 1952 [4], was a breakthrough in these fields as it was the first model to explain the electrical activity of neurons. The FitzHugh-Nagumo (FHN) model, proposed in 1961 [2], is a simplified version of the HH model that retains its essential features while being much more computationally efficient. The FHN model is a system of two ordinary differential equations that describe the dynamics of a cell action potential (AP). Because of its simplicity, the FHN model has become one of the most important building blocks in computational electrophysiology and neurophysiology, and it is widely used in simulations of neuronal and cardiac activity.

Despite its advantages, the numerical solution of millions of cell action potential models based on the FHN model is still computationally expensive, especially for large-scale simulations, such as whole-organ electrophysiology modeling [7], and drug testing and development [8]. Therefore, there is a need for lightweight computational models, also called emulators, proxies, or surrogates, that can replace the original cell model.

In this work, we present a new approach to replace the FHN model by physics-informed neural networks. This machine-learning technique uses mathematical equations that describe the phenomena during the model development or training phase. The resulting neural network model takes as input the time, initial conditions, and parameters of the FHN model and provides accurate approximations for the two variables of the original equations at the specified time.

Different from classical numerical methods for transient equations, causality is not needed after the network is trained. The computations of $u(t_1)$ and $u(t_2)$ by the network do not have to respect any relation between the time instants t_1 and t_2. For instance, t_1 can be greater than t_2. This further speeds up the calculation of the solutions since fine numerical discretizations are no longer needed.

Due to the nonlinearity of the model, sharp transitions, unstable equilibria of the solutions, and bifurcations of parameters, two techniques were used to improve the training phase of the neural network: Time window marching and increasing point cloud density on transition regions. The new physics-informed neural networks were generated with the NVIDIA Modulus framework [6] and are optimized for deployment in modern GPUs. The results show that the neural networks were able to reproduce the FHN solutions with average numerical errors below 0.5%. This approach has the potential to significantly reduce the computational cost of large-scale simulations in electrophysiology and neurophysiology.

2 Background

The use of neural networks for electrophysiology modeling has become increasingly important and interesting in recent years, particularly for complex and expensive cardiac electrophysiology models. One notable study in this area is [3], which demonstrated the ability of neural networks to emulate the intricate spatial and temporal dynamics of tissue action potential (AP) propagation.

Recently, physics-informed neural networks have emerged as a promising approach to improving data-driven models [1]. Physics-informed neural networks (PINNs) combine deep learning with physical laws to improve data-driven models. In a PINN, a neural network is trained to predict the solution to a physical problem while enforcing the governing differential equations to capture the underlying physical behavior of the system being modeled. This approach has been applied to various fields such as fluid mechanics, heat transfer, and electrophysiology modeling.

In the context of cardiac electrophysiology, several recent works have demonstrated the potential of PINNs informed by electrophysiological dynamics, using the relationships described by the Eikonal equation, to generate accurate predictions for cell activation time. For example, Bin et al. [12] and Sahli et al. [11] both presented PINN-based methods for predicting cell activation time with reasonable accuracy and performance. Additionally, Ruiz-Baier et al. [10] proposed a method to estimate the cardiac fiber architecture using physics-informed neural networks.

This concept has also been extended to more complex cell activation models. In [3], the authors used PINNs to reconstruct the entire spatiotemporal evolution of the monodomain model, accurately predicting action potential propagation in 1D and 2D meshes with only a sparse amount of data in the training. The resulting PINNs were also used to perform the inverse estimation of parameters using both *in silico* and *in vitro* data, showcasing their potential for clinical applications.

Overall, the use of neural networks and PINNs in electrophysiology modeling has shown great promise in recent years and has the potential to significantly improve our understanding and prediction of complex electrophysiological phenomena. The networks can prove to be a more efficient replacement for the numerical solvers in order to facilitate studies requiring large-scale simulations, such as whole organ simulations of the heart and brain in fine detail.

3 Methods

3.1 FitzHugh-Nagumo model

The FitzHugh-Nagumo model is one the simplest model capable of describing the dynamics of excitable cells such as neurons and cardiac cells. It consists of two coupled ordinary differential equations (ODEs) that capture the dynamics of the cell through the activation and recovery variables. The FHN model is given by:

$$\frac{dU}{dt} = KU(U-\alpha)(1-U) - W, \tag{1}$$

$$\frac{dW}{dt} = \epsilon(\beta U - 0.8W). \tag{2}$$

The model is particularly useful in studying the activation and all-or-nothing behaviors of the action potential generation dynamic in cardiac cells with few equations and parameters. The above equations were numerically solved with the classical Euler method in order to generate data for the training and validation phases of the neural networks.

In this study, the following model parameters are used: $\epsilon = 0.5$, $\alpha = 0.4$, and $\beta = 0.2$. The remaining parameter K as well as the initial conditions for U and W are allowed to vary.

3.2 Neural Network Architecture

In this work, the neural networks were developed with Modulus, NVIDIA's framework for neural networks, capable of incorporating data-driven and physics-informed constraints. Modulus provides a high-level interface for training and deploying PINNS and traditional neural networks. It can handle complex model geometries and parameterization to produce surrogate models for various high-dimensional problems. It is designed to be used with NVIDIA's high-end GPUs to accelerate training and evaluation.

The PINNs in this study were used to replace the FHN model, replicating the temporal evolution of the solutions in the whole parameter space, and for any initial conditions. The proposed NN model has the following form:

$$u_{net}(U_0, W_0, K, t) = U, W. \tag{3}$$

To handle this problem a network topology was designed using a total of 10 hidden layers, each one with 300 nodes, all fully connected, i.e., a Multi-Layer Perceptron (MLP) architecture. The hidden layers connect the input layer, which consists of time, the initial conditions, and the model parameter, to an output layer with nodes for the variables U and V.

The neural network proposed in this study was trained using a loss function that incorporates both physics-informed and data-driven constraints, providing the network with actual data and prior knowledge of the system dynamics. As a result, the network produces more accurate solutions and converges faster.

The physics constraints are incorporated by accounting for the consistency of the solutions predicted by the network in relation to the system's known governing equations, described by the FHN model:

$$f_u(U, W) = KU(U - 0.4)(1 - U) - W \tag{4}$$

$$f_w(U, W) = 0.5(0.2U - 0.8W) \tag{5}$$

The loss function comprises the sum of losses for each variable, evaluated in a domain using a specified batch size and aggregated using the L_2 norm. This interior loss function, L_i is expressed as:

$$L_I(U_0, W_0, K, t) = \frac{1}{N_{batch}} \left(\sum^{batch} \frac{\partial u_{net}(U_0, W_0, K, t)}{\partial t} - f(U, W) \right)^2, \tag{6}$$

where *batch* is a meta-parameter used to define the number of points used to evaluate the above expression during the training phase. In particular, these points will be randomly chosen.

The resulting constraint requires the derivative of the output of the model with respect to the input parameters, such as time. In order to calculate it, the network u_{net} is assumed to be differentiable, which is a reasonable assumption since the activation functions are infinitely differentiable. The derivatives are then calculated by Modulus using Automatic Differentiation [9] .

Additionally, to ensure accurate solutions, another constraint is imposed on the neural network to enforce proper initial conditions. This is achieved by measuring the error of the network's prediction at the initial time in relation to a constant or parameterized initial state. The initial condition constraint, L_B has the following form:

$$L_B(U_0, W_0, K) = \frac{1}{N_{batch}} \sum^{batch} (u_{net}(U_0, W_0, K, t = 0) - (U0, W0))^2 \quad (7)$$

Finally, a data-driven constraint is also introduced. It evaluates the similarity of the network predictions in relation to data generated using numerical results of the FHN equations. Similarly, the data constraint, L_D consists of the L_2 aggregation of the error function in a number of points defined by the batch size:

$$L_D = \frac{1}{N_{batch}} \sum^{N_{batch}} \frac{1}{2}(u_{net}(U_0, W_0, K, t) - u_{solver}(U_0, W_0, K, t))^2. \quad (8)$$

For this, a training set was assembled containing the time evolution of different solutions in the parameter space, explored on a regular grid manner, with 10 homogeneously spaced values for each parameter. The solutions are obtained using the simple forward Euler numerical solver with a time step of 0.01.

The total loss function was formulated simply by the weighted sum of the previous constraints, and is given by:

$$L_T = \lambda_D L_D + \lambda_I L_I + \lambda_B W L_B. \quad (9)$$

The λ coefficients control the relative weighting of each constraint in the total loss. In this implementation, the initial condition constraint is heavily imposed with a much larger weighting ($\lambda_B > \lambda_I, \lambda_D$) to ensure its effectiveness in eliminating non-viable solutions. However, this does not mean the solution is overly constrained since the initial condition constraint only evaluates at $t = 0$. By giving greater weight to the initial condition constraint, the network can learn the correct initial conditions for the system, which is crucial for generating reliable solutions. At the same time, the other constraints are still crucial for ensuring that the solutions are consistent with the governing equations and the available data.

A training protocol was formulated for PINNs with a fixed amount of training steps (3000). The loss function was minimized using the classical Adam optimizer. An exponential decaying function was used for the learning rate:

$$l_r = l_{r0}\gamma^{step}. \quad (10)$$

This allows the network to learn more aggressively in the initial training steps and only make minor adjustments to fine-tune the solutions in the final steps.

3.3 Advanced techniques

In this work, we use PINNs to replace the FHN model with increasing dimensions of parameterization. As a result, the complexity of the problem increases at each stage. To tackle the more challenging problems in the later stages, we implemented two techniques to enhance model training efficiency and improve the overall accuracy, which are described next.

Time Window Marching Large time domains, with heterogeneous scales of solution behavior, pose additional challenges during the training phases. In our case, we have a rapid activation, a slow recovery, followed by a long steady state rest, resulting in vastly different behaviors across time.

To mitigate this issue, we implemented the time window marching technique, which is illustrated in Figure 1. The idea of this technique is to divide the time domain into smaller windows and solve them separately. Different networks are trained separately for each time window.

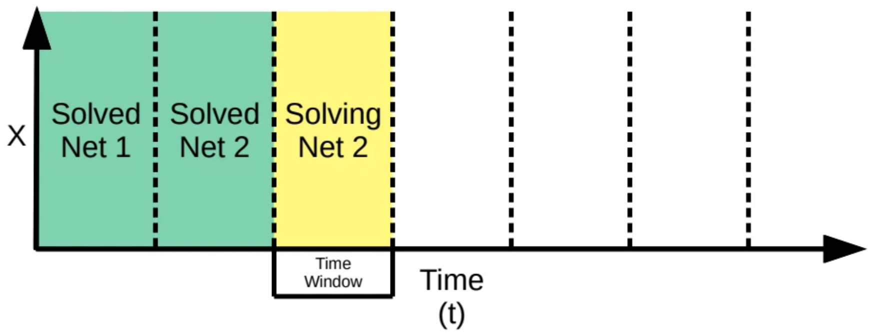

Fig. 1. Time window scheme, each training window consists of a region of the time domain to be trained separately (adapted from [6]).

An additional constraint is considered to connect the solution of each window to the next, which imposes the continuity of the solution on the interface of the time windows. The constraint is given by:

$$L_W(Y) = \frac{1}{N_{batch}} \sum^{batch} (u_{net}(U_0, W_0, K, t)^{prevW} - u_{net}(U_0, W_0, K, t)^{nextW})^2. \tag{11}$$

The points are sampled at the window interface, i.e., the final time of the previous window, and the initial condition of the next. This constraint forces the initial conditions of the next window and thus a weight similar to the one of the initial condition constraint was adopted.

Overall, time window marching is a helpful technique to handle large and complex time domains, especially when the solution behavior is highly heterogeneous across time. It also helps reduce the computational cost and memory usage required for solving the problem.

Increasing point cloud density on transition regions Another common source of complexity, and consequently also a source of error, are the regions in the parameter space where the solution behavior changes rapidly. This happens, for instance, when the solution is near unstable equilibria or when the parameters are near bifurcations. An example of this complexity can be seen in the activation threshold, generated by the term $(u - \alpha)$ of the FHN equations. Solutions with U_0 above the threshold, known as supra-threshold, trigger an action potential, whereas solutions below the threshold, known as sub-threshold, only decay to the steady state. Due to the scarcity of data in these critical regions, the resulting predictions can often contain significant errors.

This problem is addressed by constraints implemented to evaluate additional points in such transition regions, forcing the network to learn more about the solution in those regions with more physics-generated data. This approach accelerates convergence at a small cost, proportional to the batch training size used in the transition regions.

The constraint used has the same form as the regular psychs constraints but is enforced only on points within the critical region. It also has a separate λ coefficient for the weight.

3.4 Validation

In order to assess the quality of the trained networks, validation sets were also generated. This generation was done by sampling the numerical solutions for different parameter sets than the ones used in training and using the same numerical solver (forward Euler). The validation loss was also calculated with the L_2 error for the whole domain, which compares numerical solutions and the predictions of the network.

4 Results

4.1 Basic scenario with time as the single parameter

The first proposed problem is a simple time-only parameterization of the FHN model. Requiring the network to predict the solution of the model for the whole time domain for specific initial conditions and set of parameters. The proposed model has the form:

$$u_{net}(t) \approx FHN(t). \tag{12}$$

The training was done by minimizing the total loss based only on physics-informed constraints of the interior of the time domain, $t \in (0, 10)$ with a batch size of 3000 points, and the boundary of the time domain, $t = 0$, with a batch size of 500.

For such a simple problem, data was unnecessary, and a very good match was obtained using only physics and the time window technique. The time domain was divided into 20-time windows, each with the size of a time unit, and training separately for 3000 steps. This might be a little costly, computationally wise, but allows the network to train for this particular problem only with the prior knowledge of the mathematical equations that describe the dynamics of the system.

Fig 2 shows how the network is capable of solving the problem for the whole time domain, closely matching the numerical solver.

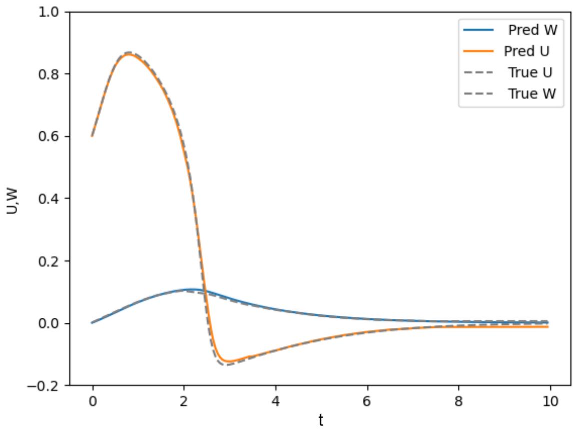

Fig. 2. Scenario with one input parameter: time. Predictions of the neural network for the $FHN(t)$ model (pred) compared to the numerical solution using the forward Euler method (true).

The resulting network prediction achieved a mean error of 0.005, with a relative error of around 0.5%. The maximum error was 0.002. For such a simple problem, high precision is expected; what makes this interesting is that the network managed such precision without the need for data, with only prior knowledge of the system dynamics and the initial state.

4.2 PINN parameterized by time and the initial condition for U

The next step was to increase the problem's complexity by parameterizing part of the initial condition. The initial condition U_0 of the solution was parameterized, requiring the network to predict solutions for any initial condition U_0 and for any time instant, t. The proposed network to solve the problem is described by:

$$u_{net}(t, U_0), \approx FHN(t, U_0). \tag{13}$$

In this case, the increase in complexity required adding a data-driven constraint, as the network did not produce a good fit with only physics-driven constraints and with the limitation, for the sake of comparison, of 3000 iterations. The total loss function contains information on particular solutions, data-driven, and on the governing equations of the FHN model, physics-informed. Additionally, a constraint for initial conditions is also imposed. The time domain was divided into 10 time windows for this problem.

The network was trained with a batch size of 1000 for the data-driven constraint and 2000 for the physics-driven constraint, with the same 500 batch size for the initial condition. It was also set to train for 3000 steps per window.

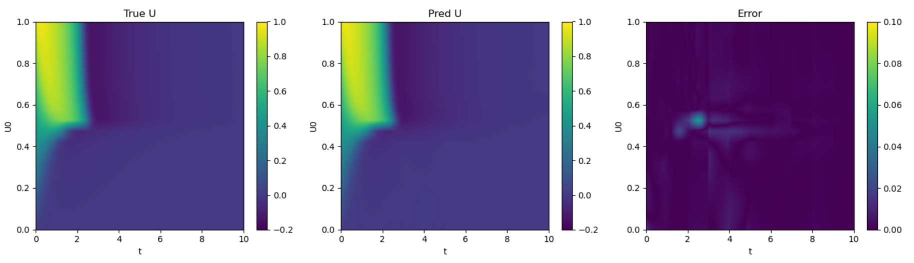

Fig. 3. Neural network's solutions to the $FHN(t, U_0)$ problem. The first column shows the true solution, while in the second the network prediction is presented, and finally, in the third column the error between the two is displayed. In this scenario, t and U_0 parameterize the neural network.

In this case, the network solution had a mean error of 0.0012 for the U variable, representing a good approximation with less than 0.2% relative error. Furthermore, by analyzing individual solutions, as shown in Fig. 4, one can see how the network manages to replicate the action potential dynamics of the model.

The maximum error in the solution space is 0.04, as shown in Fig. 3. It occurs in a very localized region near the threshold $\alpha = 0.4$. This region contains solutions very distinct since this is an unstable equilibrium.

4.3 PINNs for any initial conditions of the FHN equations ($\boldsymbol{U_0}$,$\boldsymbol{W_0}$) and time

Now the network has to solve the whole temporal evolution of the FHN model for a fixed set of parameters and any initial condition. Effectively, replacing the numerical solver for a given set of model parameters. The proposed model has the form:

$$u_{net}(t, U_0, W_0) = U, W(t, U_0, W_0). \tag{14}$$

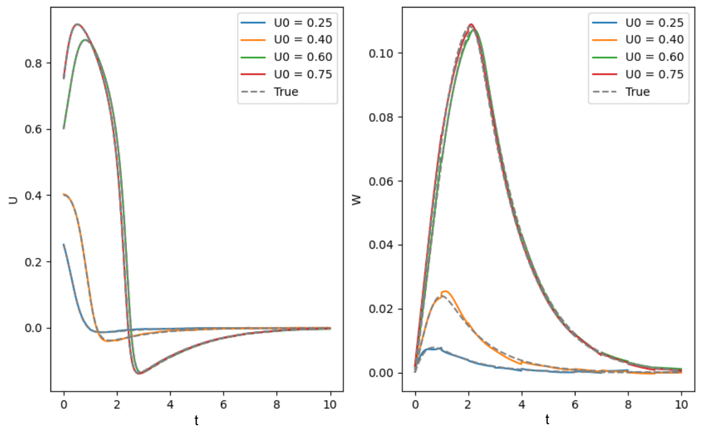

Fig. 4. Specific solutions of the $FHN(t, U_0)$ problem: network predictions compared to the numerical scheme (true) indicated by the dashed line.

In this problem, the same constraints of the last case were used. However, the batch size of each constraint was expanded to account for the higher dimensionality of the problem, which, of course, results in a higher training time when compared to previous cases.

Both the physics and the data constraints had a λ weighting of one and the IC constraint was heavily enforced with a weighting of 1000. The network was trained with batch sizes of 7000 for the data constraints, 5000 for the physics constraints, and 500 for the boundary constraints.

Fig 5 shows how the network is capable of replicating the model in the whole solution space. And that most of the error is still localized in the region immediately above the threshold.

Fig. 6 presents specific network predictions, which had a mean error of only 0.001 and a maximum error of 0.01, meaning a relative mean and maximum error of 0.1% and 1%, respectively. From the results, we note that the neural network yields accurate enough results.

4.4 PINNs replace the FHN model parameterized by any initial conditions, the parameter K, and time

Our final experiments further increase the complexity of the problem by adding an extra dimension for the parameterization of the FHN model (the parameter K), which represents the velocity of the dynamics of the action potential. Now the network not only has to solve for any initial condition, but it also has to produce different solutions for the K parameter. The proposed model has the following form:

$$u_{net}(t, U_0, W_0, K) = U, W(t, U_0, W_0, K). \tag{15}$$

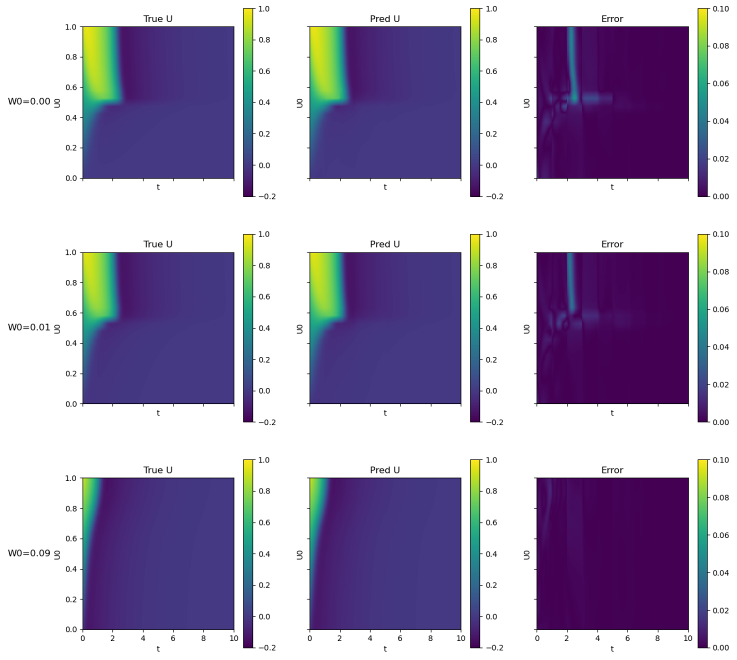

Fig. 5. Colormaps showing the evaluations of the Neural Network, FHN numerical solution and the respective error for three different values of W_0. U_0 varies along the vertical axis and time along the horizontal axis.

For this problem, initially, the error in the critical region near the unsteady equilibrium was too high. Therefore, an additional constraint was explicitly added for evaluating the consistency of the solution in this region, i.e., for $U_0 \in (0.45, 0.65)$. This constraint double-enforces the solution of the FHN equations in this region.

The physics constraints were evaluated with a batch size of 2000 for the constraint enforced in the whole domain and 1000 for the one enforced on the critical region. Furthermore, the initial condition constraint had a batch size of 500, and the data-driven had a batch size of 2000. The time domain was divided in 10 windows, like in the previous examples. In this case, data and physics constraints had a λ weighting of one and the IC constraint was heavily enforced with a weight of 1000.

The network's performance for this problem is worse than the previous examples, with a maximum error of 0.18 (18% of maximum relative error). However,

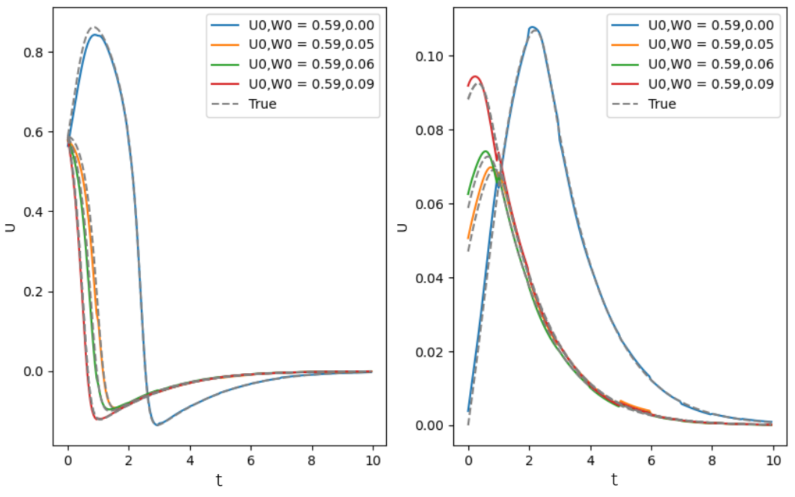

Fig. 6. Graphics showing the specific network predictions (solid lines) and the numerical solutions of the FHN model (dashed) for different values of W_0.

we can see in Figure 7 that this high error is highly localized. The relative mean error measured was 0.1%, showing that the network has good overall accuracy.

5 Discussion

Table 1 summarizes all the presented results. Note that in each different scenario presented the generation of the PINNs took less than one hour of execution time on a single NVIDIA Tesla T4 with 16GB of memory. Also, note that the number of advanced techniques used to improve the results steadily increased from only physics-informed constraints and the time window technique (PINNs parameterized only by time) to physics-informed and data-driven constraints, time window technique and an increased point cloud density on a critical region near an unsteady equilibrium (PINNs parameterized by time, the initial conditions, and the parameter K of the FHN model).

It's important to note that although several techniques were employed to improve model accuracy and training rate there is much more room to improve. One conceivable way is to explore the use of different neural network architectures. Recurrent architectures can be particularly suited for this problem due to their time-aware nature that propagates the solution through time, at the cost of being traditionally more expensive than MLP [13]. Another architecture worth mentioning is the Fourier-based one [5]. By exploring different neural network architectures, it may be possible to find more efficient and effective models for

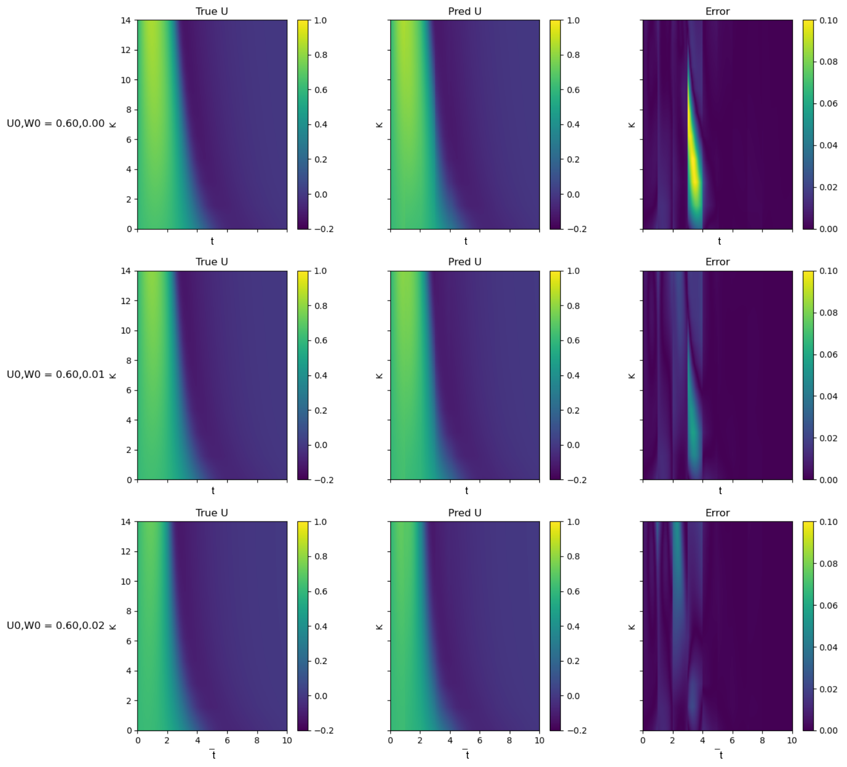

Fig. 7. Figures showing the evaluations of the Neural Network in the whole solution space, and its respective error for 3 parameter sets, U_0 , W_0, and K. The results show a reduced localized error after the special constraint was imposed in the region near the unsteady equilibrium.

solving this problem. However, it is important to carefully consider the trade-offs between model accuracy and computational cost, which is a topic of future research.

6 Conclusions

Together with the Hodgkin-Huxley model, the FitzHugh-Nagumo model is an essential building block in computational electrophysiology and neurophysiology. The FHN is a simple model based on two ordinary differential equations that can reproduce the dynamics of a cell action potential. Despite its simplicity,

Table 1. Table showing each studied scenario, if physics or data constraints were used, the performance techniques employed, the PINNs training time, and the resulting accuracy.

Input parameters	Mean Error	Max error	Training time	Employed techniques
Time	0.0058	0.041	23 min	Physics + Window,
Time, U_0	0.0012	0.042	12 min	Physics+Data+Window
Time, U_0 and W_0	0.0014	0.113	43 min	Physics+Data+Window
Time, U_0 , W_0 and K	0.0007	0.164	25 min	All previous + region

the FHN model is widely used in many research fields, including neuroscience, physiology, and cardiology.

However, the numerical solution of millions of cell models is computationally expensive, particularly in tasks such as whole-organ electrophysiology modeling for the brain or the heart and drug testing and development.

Machine learning techniques, particularly neural networks, have emerged as promising tools for generating emulators in recent years. Physics-informed neural networks (PINNs) are a type of neural network that use the mathematical equations that describe the phenomena during the model development or training phase, making them particularly well-suited for modeling complex physical systems, such as the FHN model.

In this work, we present how to replace the FHN model with PINNs, using the NVIDIA Modulus framework to generate the neural network models. The new neural network model receives input time, t, initial conditions, and parameters of the FHN and offers accurate approximations for the two variables of the original equations at the specified time.

To deal with the nonlinearity of the model, sharp transitions, unstable equilibria, and bifurcations of the parameters, two techniques were essential to improve the training phase of the neural network: Time window marching and increasing point cloud density on transition regions. These techniques allowed us to generate accurate neural network models that reproduce FHN solutions with average numerical errors below 0.5%.

In conclusion, our work shows how PINNs can generate accurate emulators for the FHN model, a widely used model in computational electrophysiology and neurophysiology. The new neural network models are computationally lightweight, making them well-suited for whole-organ electrophysiology modeling, drug testing, and development. Moreover, our work demonstrates the potential of PINNs as a tool for emulating other complex physical models and highlights the importance of combining data-driven and physics-based approaches in machine learning for scientific applications.

Acknowledgements

This work was supported by NVIDIA (project "Patient-specific models of the heart for precision medicine", NVIDIA Academic Hardware Grant Program), the

Federal University of Juiz de Fora, Brazil, through the scholarship "Coordenação de Aperfeiçoamento de Pessoal de Nível Superior" (CAPES) -Brazil-Finance Code 001; by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)-Brazil Grant numbers 423278/2021-5, 308745/2021-3 and 310722/2021-7; by Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) TEC APQ 01340/18

References

1. Cai, S., Mao, Z., Wang, Z., Yin, M., Karniadakis, G.E.: Physics-informed neural networks (PINNs) for fluid mechanics: A review. Acta Mechanica Sinica **37**(12), 1727–1738 (2021)
2. FitzHugh, R.: Impulses and physiological states in theoretical models of nerve membrane. Biophysical journal **1**(6), 445–466 (1961)
3. Herrero Martin, C., Oved, A., Chowdhury, R.A., Ullmann, E., Peters, N.S., Bharath, A.A., Varela, M.: Ep-pinns: Cardiac electrophysiology characterisation using physics-informed neural networks. Frontiers in Cardiovascular Medicine **8**, 2179 (2022)
4. Hodgkin, A., Huxley, A.: A quantitative description of membrane current and its application to conduction and excitation in nerve. The Journal of physiology **117**(4), 500–544 (1952)
5. Ngom, M., Marin, O.: Fourier neural networks as function approximators and differential equation solvers. Statistical Analysis and Data Mining: The ASA Data Science Journal **14**(6), 647–661 (2021)
6. NVIDIA: Nvidia modulus (2022), https://www.nvidia.com/en-us/docs/developer/modulus/
7. Oliveira, R.S., Alonso, S., Campos, F.O., Rocha, B.M., Fernandes, J.F., Kuehne, T., Dos Santos, R.W.: Ectopic beats arise from micro-reentries near infarct regions in simulations of a patient-specific heart model. Scientific reports **8**(1), 1–14 (2018)
8. Passini, E., Britton, O.J., Lu, H.R., Rohrbacher, J., Hermans, A.N., Gallacher, D.J., Greig, R.J., Bueno-Orovio, A., Rodriguez, B.: Human in silico drug trials demonstrate higher accuracy than animal models in predicting clinical pro-arrhythmic cardiotoxicity. Frontiers in physiology p. 668 (2017)
9. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
10. Ruiz Herrera, C., Grandits, T., Plank, G., Perdikaris, P., Sahli Costabal, F., Pezzuto, S.: Physics-informed neural networks to learn cardiac fiber orientation from multiple electroanatomical maps. Engineering with Computers **38**(5), 3957–3973 (2022)
11. Sahli Costabal, F., Yang, Y., Perdikaris, P., Hurtado, D.E., Kuhl, E.: Physics-informed neural networks for cardiac activation mapping. Frontiers in Physics **8**, 42 (2020)
12. bin Waheed, U., Haghighat, E., Alkhalifah, T., Song, C., Hao, Q.: Pinneik: Eikonal solution using physics-informed neural networks. Computers & Geosciences **155**, 104833 (2021)
13. Wu, B., Hennigh, O., Kautz, J., Choudhry, S., Byeon, W.: Physics informed rnn-dct networks for time-dependent partial differential equations. In: Computational Science–ICCS 2022: 22nd International Conference, London, UK, June 21–23, 2022, Proceedings, Part II. pp. 372–379. Springer (2022)