Methodology for an Intelligent System Constructing for Modeling the Conduction of an Electrical Impulse in Nerve Axons

Michael Z. Zgurovsky[1], Pavlo O. Kasyanov[2], Liudmyla B. Levenchuk[1] and Vladyslav R. Novykov[1]

[1] *National Technical University of Ukraine 'Igor Sikorsky Kyiv Polytechnic Institute', Beresteiskyi Ave. 37 build.35, Kyiv, 03056, Ukraine*
[2] *Institute for Applied System Analysis, National Technical University of Ukraine 'Igor Sikorsky Kyiv Polytechnic Institute', Beresteiskyi Ave., 37 build. 35, Kyiv, 03056, Ukraine*

Abstract

This article presents a methodology for building an intelligent system for approximate solutions of the Hodgkin-Huxley equation, which models the conduction of an electrical impulse in nerve axons. Given the complexity of this nonlinear differential equation and the non-uniqueness of its solutions, we use advanced computational methods, including Physical Information Neural Networks (PINN) and Deep Learning Galerkin Method (DLGM). These methods allow us to transform infinite-dimensional stochastic optimization problems into finite-dimensional ones, providing efficient and accurate numerical simulations. Our approach combines machine learning with classical biological modeling to overcome the limitations of traditional numerical methods. We develop an algorithm that approximates solutions for electrical impulse conduction models by capturing both quantitative and qualitative characteristics of nerve impulse dynamics. Numerical results confirm the effectiveness of our methodology, demonstrating accurate approximations and stability of traveling wave solutions for various parameter settings. This research provides a deeper understanding of neuronal behavior and offers potential applications in the development of new therapeutic strategies.

Keywords

reaction-diffusion equations; multivalued interaction functions; machine learning; physics-informed neural networks; approximate solutions

MSC2020: 35-04, 35R70

1. Introduction

The Hodgkin-Huxley equation describes the conduction of the nerve impulse in the optic nerve. The equation has the form

$$\frac{\partial e}{\partial t} = \frac{\partial^2 e}{\partial x^2} + f,$$

where f is a complicated nonlinear function of e and its past values. A detailed description of the function f is not necessary here. The physiological fact modeled by this equation is as follows: if the nerve is stimulated below the threshold, the disturbance simply dampens out, but if stimulated above the threshold, the disturbance quickly forms a particular shape and moves along the line like a traveling wave. Hodgkin [4] presents the physiological background and compares the results of numerical integration with experimental observations. The mathematical problem is to classify the possible waveforms $\bar{e}(x)$ and prove that any disturbance $e(t,x)$ of appropriate initial size and shape, traveling at the appropriate speed k, stabilizes to a translation of one of these forms:

ProfIT AI 2024: 4th International Workshop of IT-professionals on Artificial Intelligence (ProfIT AI 2024), September 25–27, 2024, Cambridge, MA, USA

mzz@kpi.ua (M. Zgurovsky); kasyanov@i.ua (P. Kasyanov); levenchuk.liudmyla@lll.kpi.ua (L. Levenchuk); vlad.novykov@gmail.com (V. Novykov)

0000-0001-5896-7466 (M. Zgurovsky); 0000-0002-6662-0160 (P. Kasyanov); 0000-0002-8600-0890 (L. Levenchuk); 0009-0003-4249-6454 (V. Novykov)

$$\lim_{t\to\infty} e(t, x + kt) = \bar{e}(x);$$

see [7] and references therein. The complexity of this equation has led to the introduction of simplified models of the Hodgkin-Huxley equation to understand the stabilization mechanism in similar but simpler circumstances. This issue is reviewed in the works of Cohen [2] and Rinzel [9]. The first simplification is the Fitzhugh-Nagumo equation [3]. Considering that the process is described by a reaction-diffusion differential equation with discontinuous nonlinearity, the uniqueness of solutions to the Cauchy problem is not always guaranteed. Therefore, the quantitative analysis of solutions to such equations requires a methodology for finding all solutions to the Cauchy problem or, at the very least, a sufficient variety of approximate methods that can potentially provide an approximation of another solution to the corresponding problem without uniqueness. In this work, we continue the developments laid out in the work on the quantitative study of solutions of a class of differential inclusions using so-called physics-informed neural networks (PINNs) [6]. We note that the qualitative analysis and stability theory for differential inclusions with partial derivatives were developed in [11]. In the future, the developed methodology will be applied to systems of discontinuous nonlinearities with applications to both the original Cohen [2] and Rinzel [9] models, FitzHugh-Nagumo, as well as to other mathematical models such as climatology models, heat-mass transfer, unilateral problems, problems on manifolds with and without boundaries, differential-operator inclusions with pseudomonotone type operators, and evolutionary hemivariational inequalities with possibly nonmonotone potentials. In addition to computational studies, considerable attention will be given in the future to empirically validating the obtained results through a series of experiments

2. Problem definition

In this paper, we study an intelligent system methodology designed to approximate solutions for the model of electrical impulse conduction in nerve axons. We explore the integration of advanced computational techniques with biological modeling, aiming to overcome the limitations of classical numerical methods. Recent advancements in intelligent systems, including machine learning and neural networks, offer promising new avenues for developing approximate solutions to these complex models. By leveraging the computational power and adaptability of intelligent systems, we can enhance the accuracy and efficiency of simulations, providing deeper insights into neuronal behavior and potentially informing the development of novel therapeutic strategies.

Consider the problem:

$$\begin{cases} \dfrac{\partial u}{\partial t} - \dfrac{\partial^2 u}{\partial x^2} + u \in H(u - a), \quad (x,t) \in (0,\pi) \times (0,T), \\ u(0,t) = u(\pi,t) = 0, \quad t \in [0,T], \end{cases} \tag{1}$$

with initial conditions:

$$u(x,0) = u_0(x) \tag{2}$$

where $a \in \left(0, \frac{1}{2}\right)$, and H is the Heaviside step function.

For a fixed $u_0 \in C_0^\infty(\mathbb{R})$ let $supp\ u_0 \subset (0,\pi)$. According to [13] (see the book and references therein), there exists a weak solution $u = u(x,t) \in L^2(0,T;H_0^1(0,\pi))$, with $\frac{\partial u}{\partial t} \in L^2(0,T;H^{-1}(0,\pi))$, of Problem (1) – (2) in the following sense:

$$-\int_0^T\int_0^\pi u(x,t)v(x)\frac{d}{dt}\eta(t)dxdt \quad + \\ +\int_0^T\int_0^\pi (\frac{\partial}{\partial x}u(x,t)\cdot\frac{d}{dx}v(x)+u(x,t)v(x)-d(x,t)v(x))\eta(t)dt=0, \tag{3}$$

for all $v,\eta\in C_0^\infty(0,T)$, where $d\colon \mathbb{R}\times(0,T)\to\mathbb{R}$ be a measurable function such that

$$d(x,t)\in H(u(x,t)-a) \quad for\ a.e.\ \ (x,t)\in\mathbb{R}\times(0,T). \tag{4}$$

The main goal of this paper is to develop an algorithm for approximation of solutions for classes of electrical impulse conduction in nerve axons con with multivalued interaction functions allowing for non-unique solutions of the Cauchy problem (1) – (2) via the PINNs; [1, 5, 6, 8] and references therein.

3. Methodology of Approximate Solutions for Electrical Impulse Conduction Equations with Multivalued Interaction Functions

Fix an arbitrary $T>0$, and a sufficiently smooth function $u_0\colon\mathbb{R}\to\mathbb{R}$ with compact support $supp\ u_0:=\mathrm{cl}\{x\in\mathbb{R}:\ u_0(x)\neq 0\}$. We approximate the function $s\mapsto H(s-a)$ by the following Lipschitz functions:

$$H_k(s):=\begin{cases}0, & s<a;\\ k(s-a), & s\in\left[a,a+\frac{1}{k}\right);\ k=1,2,\ldots.\\ 1, & s\geq a+\frac{1}{k},\end{cases} \tag{5}$$

For a fixed $k=1,2,\ldots$, consider the problem:

$$\frac{\partial u_k}{\partial t}(x,t)=\frac{\partial^2 u_k(x,t)}{\partial x^2}+H_k(u_k(x,t))-u_k(x,t), \quad (x,t)\in\mathbb{R}\times[0,T], \tag{6}$$

with initial conditions:

$$u_k(x,0)=u_0(x), \quad x\in\mathbb{R}. \tag{7}$$

According to [11] and references therein, for each $k=1,2,\ldots$ Problem (6)–(7) has an unique solution $u_k\in C^{2,1}(\mathbb{R}\times[0,T])$. Moreover, [12] implies that each convergent sub-sequence $\{u_{k_l}\}_{l=1,2,\ldots}\subset\{u_k\}_{k=1,2,\ldots}$ of corresponding solutions to Problem (6)–(7) weakly converges to a solution u of Problem (1)–(2) in the space

$$W:=\{z\in L^2(0,T;H_0^1(0,\pi)):\ \frac{\partial z}{\partial t}\in L^2(0,T;H_0^1(0,\pi))\}, \tag{8}$$

endowed with the standard graph norm, where $supp\ u_0\subset(0,\pi)$, $H_0^1(0,\pi)=\{v\in L^2(0,\pi)\ \frac{\partial v}{\partial x}\in L^2(0,\pi), v(0)=v(\pi)\}$.

Thus, the **first step** of the algorithm is to replace the function f in Problem (1)–(2) with H_k considering Problem (6)–(7) for sufficiently large k; cf [6].

For this purpose let us now consider Problem (6)–(7) for sufficiently large k. Theorem 16.1.1 from [5] allows us to reformulate Problem (6)–(7) as an infinite dimensional stochastic optimization problem over a certain function space. More exactly, let $\mathrm{t}\in C([0,T];(0,\infty))$, $\xi\in C(\mathbb{R};(0,\infty))$, let $(\Omega,\mathcal{F},\mathbb{P})$ be a probability space, let $\mathcal{T}\colon\Omega\to[0,T]$ and $\mathcal{X}\colon\Omega\to\mathbb{R}$ be independent random variables. Assume for all $A\in\mathcal{B}([0,T])$, $B\in\mathcal{B}(\mathbb{R})$ that

$$\mathbb{P}(\mathcal{T}\in A)=\int_A \mathrm{t}(t)\,dt \quad \text{and} \quad \mathbb{P}(\mathcal{X}\in B)=\int_B \xi(x)\,dx.$$

Note that $H_k\colon\mathbb{R}\to\mathbb{R}$ be Lipschitz continuous, and let $\mathcal{L}_k\colon C^{2,1}(\mathbb{R}\times[0,T])\to[0,\infty]$ satisfy for all $v=(v(x,t))_{(x,t)\in\mathbb{R}\times[0,T]}\in C^{2,1}(\mathbb{R}\times[0,T])$ that

$$\mathcal{L}_k(v) = \mathbb{E}\left[|v(\mathcal{X},0) - u_0(\mathcal{X})|^2 + \left|\frac{\partial v}{\partial t}(\mathcal{X},\mathcal{T}) - \frac{\partial^2 v}{\partial^2 x}(\mathcal{X},\mathcal{T}) - H_k(v(\mathcal{X},\mathcal{T})) + v(\mathcal{X},\mathcal{T})\right|^2\right].$$

Theorem 16.1.1 from [5] implies that the following two statements are equivalent:

1. It holds that $\mathcal{L}_k(u_k) = \inf_{v \in C^{2,1}(\mathbb{R}\times[0,T])} \mathcal{L}_k(v)$.
2. It holds $u_k \in C^{2,1}(\mathbb{R}\times[0,T])$ is the solution of Problem (6)–(7).

Thus, the **second step** of the algorithm is to reduce the regularized Problem (6)–(7) to the infinite dimensional stochastic optimization problem in $C^{2,1}(\mathbb{R}\times[0,T])$:

$$\begin{cases} \mathcal{L}_k(v) \to \min, \\ v \in C^{2,1}(\mathbb{R}\times[0,T]). \end{cases} \tag{9}$$

However, due to its infinite dimensionality, the optimization problem (9) is not yet suitable for numerical computations. Therefore, we apply the **third step**, the so-called Deep Galerkin Method (DGM) [10], that is, we transform this infinite dimensional stochastic optimization problem into a finite dimensional one by incorporating artificial neural networks (ANNs); see [5, 10] and references therein. Let $a: \mathbb{R} \to \mathbb{R}$ be differentiable, let $h \in \mathbb{N}, l_1, l_2, \ldots, l_h, \mathfrak{d} \in \mathbb{N}$ satisfy $\mathfrak{d} = 4l_1 + [\sum_{k=2}^{h} l_k(l_{k-1}+1)] + l_h + 1$, and let $\mathcal{L}_{k,h}: \mathbb{R}^{\mathfrak{d}} \to [0,\infty)$ satisfy for all $\theta \in \mathbb{R}^{\mathfrak{d}}$ that

$$\begin{aligned} \mathcal{L}_{k,h}(\theta) &= \mathcal{L}_k(\mathcal{N}^{\theta,2}_{\mathfrak{M}_{a,l_1},\mathfrak{M}_{a,l_2},\ldots,\mathfrak{M}_{a,l_h},\mathrm{id}_{\mathbb{R}}}) \\ &= \mathbb{E}\left[\left|\mathcal{N}^{\theta,2}_{\mathfrak{M}_{a,l_1},\ldots,\mathfrak{M}_{a,l_h},\mathrm{id}_{\mathbb{R}}}(\mathcal{X},0) - u_0(\mathcal{X})\right|^2\right. \\ &\quad + \left|\frac{\partial \mathcal{N}^{\theta,2}_{\mathfrak{M}_{a,l_1},\ldots,\mathfrak{M}_{a,l_h},\mathrm{id}_{\mathbb{R}}}}{\partial t}(\mathcal{X},\mathcal{T}) - \frac{\partial^2 \mathcal{N}^{\theta,2}_{\mathfrak{M}_{a,l_1},\ldots,\mathfrak{M}_{a,l_h},\mathrm{id}_{\mathbb{R}}}}{\partial^2 x}(\mathcal{X},\mathcal{T})\right. \\ &\quad \left.\left. - H_k\left(\mathcal{N}^{\theta,2}_{\mathfrak{M}_{a,l_1},\ldots,\mathfrak{M}_{a,l_h},\mathrm{id}_{\mathbb{R}}}(\mathcal{X},\mathcal{T})\right) + \mathcal{N}^{\theta,2}_{\mathfrak{M}_{a,l_1},\ldots,\mathfrak{M}_{a,l_h},\mathrm{id}_{\mathbb{R}}}(\mathcal{X},\mathcal{T})\right|^2\right], \end{aligned} \tag{10}$$

where $\mathfrak{M}_{\psi,d}$ is the d-dimensional version of a function ψ, that is,

$\mathfrak{M}_{\psi,d}: \mathbb{R}^d \to \mathbb{R}^d$

is the function which satisfies for all $x = (x_k)_{k\in\{1,2,\ldots,d\}} \in \mathbb{R}^d$, $y = (y_k)_{k\in\{1,2,\ldots,d\}} \in \mathbb{R}^d$ with $\forall k \in \{1,2,\ldots,d\}: y_k = \psi(x_k)$ that

$\mathfrak{M}_{\psi,d}(x) = y$;

for each $\mathfrak{d}, L \in \mathbb{N}, l_0, l_1, \ldots, l_L \in \mathbb{N}, \theta \in \mathbb{R}^{\mathfrak{d}}$ satisfying $\mathfrak{d} \geq \sum_{k=1}^{L} l_k(l_{k-1}+1)$, and for a function $\Psi_k: \mathbb{R}^{l_k} \to \mathbb{R}^{l_k}, k \in \{1,2,\ldots,L\}$, we denote by $\mathcal{N}^{\theta,l_0}_{\Psi_1,\Psi_2,\ldots,\Psi_L}: \mathbb{R}^{l_0} \to \mathbb{R}^{l_L}$ the realization function of the fully-connected feedforward artificial neural network associated to θ with $L+1$ layers with dimensions $(l_0, l_1, \ldots, l_L)$ and activation functions $(\Psi_1, \Psi_2, \ldots \Psi_L)$, defined as:

$$\mathcal{N}^{\theta,l_0}_{\Psi_1,\Psi_2,\ldots,\Psi_L}(x) = (\Psi_L \circ \mathcal{A}^{\theta,\sum_{k=1}^{L-1} l_k(l_{k-1}+1)}_{l_L,l_{L-1}} \circ \Psi_{L-1} \circ \mathcal{A}^{\theta,\sum_{k=1}^{L-2} l_k(l_{k-1}+1)}_{l_{L-1},l_{L-2}} \circ \ldots$$
$$\ldots \circ \Psi_2 \circ \mathcal{A}^{\theta,l_1(l_0+1)}_{l_2,l_1} \circ \Psi_1 \circ \mathcal{A}^{\theta,0}_{l_1,l_0})(x),$$

for all $x \in \mathbb{R}^{l_0}$; and for each $\mathrm{d}, m, n \in \mathbb{N}, s \in \mathbb{N}_0 := \mathbb{N} \cup \{0\}, \theta = (\theta_1, \theta_2, \ldots, \theta_{\mathrm{d}}) \in \mathbb{R}^{\mathrm{d}}$ satisfying $\mathrm{d} \geq s + mn + m$, the affine function $\mathcal{A}^{\theta}_{s,m,n}$ from $\mathbb{R}^n$ to $\mathbb{R}^m$ associated to (θ, s), is defined as

$$\mathcal{A}^{\theta}_{s,m,n}(x) = \begin{pmatrix} \theta_{s+1} & \theta_{s+2} & \cdots & \theta_{s+n} \\ \theta_{s+n+1} & \theta_{s+n+2} & \cdots & \theta_{s+2n} \\ \theta_{s+2n+1} & \theta_{s+2n+2} & \cdots & \theta_{s+3n} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{s+(m-1)n+1} & \theta_{s+(m-1)n+2} & \cdots & \theta_{s+mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} \theta_{s+mn+1} \\ \theta_{s+mn+2} \\ \vdots \\ \theta_{s+mn+m} \end{pmatrix}$$

for all $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$.

The **final step** in the derivation involves approximating the minimizer of $\mathcal{L}_{k,h}$ using stochastic gradient descent optimization methods [5]. Let $\xi \in \mathbb{R}^{\mathfrak{d}}$, $J \in \mathbb{N}$, $(\gamma_n)_{n\in\mathbb{N}} \subseteq [0,\infty)$, for each $n \in \mathbb{N}$, $j \in \{1,2,\dots,J\}$ let $\mathfrak{T}\colon \Omega \to [0,T]$ and $\mathfrak{X}_{n,j}\colon \Omega \to \mathbb{R}$ be random variables. Let for each $n \in \mathbb{N}$, $j \in \{1,2,\dots,J\}$, $A \in \mathcal{B}([0,T])$, $B \in \mathcal{B}(\mathbb{R})$

$$\mathbb{P}(\mathcal{T} \in A) = \mathbb{P}(\mathfrak{T}_{n,j} \in A) \quad \text{and} \quad \mathbb{P}(\mathcal{X} \in B) = \mathbb{P}(\mathfrak{X}_{n,j} \in B). \tag{11}$$

Let $\ell_{k,h}\colon \mathbb{R}^{\mathfrak{d}} \times \mathbb{R} \times [0,T] \to \mathbb{R}$ is defined as

$$\begin{aligned} \ell_{k,h}(\theta,x,t) = \; & \left|\mathcal{N}^{\theta,2}_{\mathfrak{M}_{a,l_1},\mathfrak{M}_{a,l_2},\dots,\mathfrak{M}_{a,l_h},\mathrm{id}_{\mathbb{R}}}(x,0) - u_0(x)\right|^2 \\ & + \left|\frac{\partial \mathcal{N}^{\theta,2}_{\mathfrak{M}_{a,l_1},\mathfrak{M}_{a,l_2},\dots,\mathfrak{M}_{a,l_h},\mathrm{id}_{\mathbb{R}}}}{\partial t}(x,t) - \frac{\partial^2 \mathcal{N}^{\theta,2}_{\mathfrak{M}_{a,l_1},\mathfrak{M}_{a,l_2},\dots,\mathfrak{M}_{a,l_h},\mathrm{id}_{\mathbb{R}}}}{\partial^2 x}(x,t)\right. \\ & \left. - H_k\left(\mathcal{N}^{\theta,2}_{\mathfrak{M}_{a,l_1},\mathfrak{M}_{a,l_2},\dots,\mathfrak{M}_{a,l_h},\mathrm{id}_{\mathbb{R}}}(x,t)\right) + \mathcal{N}^{\theta,2}_{\mathfrak{M}_{a,l_1},\mathfrak{M}_{a,l_2},\dots,\mathfrak{M}_{a,l_h},\mathrm{id}_{\mathbb{R}}}(x,t)\right|^2, \end{aligned} \tag{12}$$

for each $\theta \in \mathbb{R}^{\mathfrak{d}}$, $x \in \mathbb{R}$, $t \in [0,T]$, and let $\Theta = (\Theta_n)_{n\in\mathbb{N}_0}\colon \mathbb{N}_0 \times (0,\pi) \to \mathbb{R}^{\mathfrak{d}}$ satisfy for all $n \in \mathbb{N}$ that

$$\Theta_0 = \xi \quad \text{and} \quad \Theta_n = \Theta_{n-1} - \gamma_n \left[\frac{1}{J}\sum_{j=1}^{J} \left(\frac{\partial \ell_{k,h}(x,t)}{\partial t}\right)(\Theta_{n-1}, \mathfrak{T}_{n,j}, \mathfrak{X}_{n,j})\right]. \tag{13}$$

Ultimately, for sufficiently large $k,h,n \in \mathbb{N}$, the realization $\mathcal{N}^{\Theta_n,2}_{\mathfrak{M}_{a,l_1},\mathfrak{M}_{a,l_2},\dots,\mathfrak{M}_{a,l_h},\mathrm{id}_{\mathbb{R}}}$ is chosen as an approximation:

$$\mathcal{N}^{\Theta_n,2}_{\mathfrak{M}_{a,l_1},\mathfrak{M}_{a,l_2},\dots,\mathfrak{M}_{a,l_h},\mathrm{id}_{\mathbb{R}}} \approx u$$

of the unknown solution u of (1)–(2) in the space W defined in (8). So, the following theorem is justified.

Theorem 1 *Let $T > 0$, and $u_0 \in C_0^\infty(\mathbb{R})$. Then the sequence of $\{\mathcal{N}^{\Theta_n,2}_{\mathfrak{M}_{a,l_1},\mathfrak{M}_{a,l_2},\dots,\mathfrak{M}_{a,l_h},id_{\mathbb{R}}}\}_{k,h,n}$* defined in (12)–(13) has an accumulation point in the weak topology of W defined in (8). Moreover, each partial limit of the sequence in hands is weakly converges in W to the solution of Problem (1)–(2) in the sense of (3)–(4).

The empirical risk minimization problems for PINNs and DGMs are typically solved using SGD or variants thereof, such as Adam [5]. The gradients of the empirical risk with respect to the parameters θ can be computed efficiently using automatic differentiation, which is commonly available in deep learning frameworks such as TensorFlow and PyTorch. We provide implementation details and numerical simulations for PINNs and DGMs in the next section; cf. [5, 6, 8] and references theerein.

4. Numerical Implication

Let us present a straightforward implementation of the method as detailed in the previous Section for approximating a solution $u \in W$ of Problem (1)–(2) with $a = 0.1, 0.15, 0.2, 0.25, 0.3, 0.35$ and the initial condition $u_0(x) := \psi(x)$, where

$$\psi(s) := \begin{cases} \dfrac{2}{\cosh\left(10\left(x - \frac{\pi}{2}\right)\right)}, & s \in [0,\pi); \\ 0, \quad otherwise\,. \end{cases} \tag{14}$$

$x \in \mathbb{R}$. Let $k = 5$. This implementation follows the original proposal by [8], where 2.000 realizations of the random variable $(\mathcal{X}, \mathcal{T})$ are first chosen. Here, $\mathcal{T}$ is uniformly distributed over $[0,1]$, and $\mathcal{X}$ is uniformly distributed over $[0, \pi]$. A fully connected feed-forward ANN with 4 hidden layers, each containing 50 neurons, and employing the Swish activation function is then trained. The training process uses batches of size 256, sampled from the 2.000 preselected realizations of $(\mathcal{X}, \mathcal{T})$. Optimization is carried out using the Adam SGD method [5]. A plot of the resulting approximation of the solution u after 2.000 training steps is shown in Figure 1.

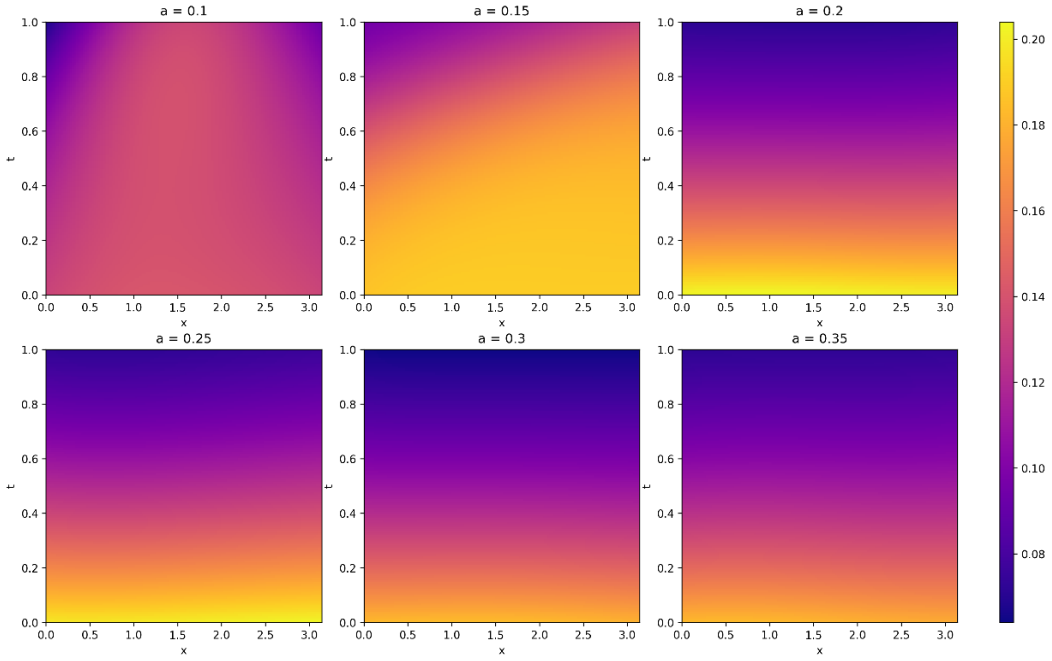

Figure 1: The birth of a soliton: plots for the functions $(0, \pi) \ni x \mapsto U(x,t) \in \mathbb{R}$, where $a \in \{0.1, 0.15, 0.2, 0.25, 0.3, 0.35\}$ and $U \in C^{2,1}(\mathbb{R} \times [0,1])$ is an approximation of the solution u of Problem (1)–(2) with $u_0(x) := \psi(x)$, where ψ is defined in (14), computed by means of the PINN method as implemented in Source code 1.

```
import os
import torch
import matplotlib.pyplot as plt
from torch.autograd import grad
from matplotlib.gridspec import GridSpec
from matplotlib.cm import ScalarMappable

dev = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

T = 1.0  # the time horizon
M = 2000  # the number of training samples
k = 5  # the parameter

torch.manual_seed(0)

x_data = torch.rand(M, 1).to(dev) * torch.pi  # Sampling x from (0,
\pi)
t_data = torch.rand(M, 1).to(dev) * T

# The initial value
def phi(x):
    return 2 / torch.cosh(10 * (x - torch.pi/2)).unsqueeze(1)
```

```
# The interaction function H_k(s)
def H_k(s, a, k):
    return torch.where(
        s < a, torch.tensor(0.0, device=s.device),
        torch.where(s < a + 1/k, k * (s - a), torch.tensor(1.0,
device=s.device))
    )

# Define the network
def create_network():
    return torch.nn.Sequential(
        torch.nn.Linear(2, 50), torch.nn.SiLU(),
        torch.nn.Linear(50, 50), torch.nn.SiLU(),
        torch.nn.Linear(50, 50), torch.nn.SiLU(),
        torch.nn.Linear(50, 50), torch.nn.SiLU(),
        torch.nn.Linear(50, 1),
    ).to(dev)

optimizer = torch.optim.Adam

J = 256  # the batch size
a_values = [0.1, 0.15, 0.2, 0.25, 0.3, 0.35]

# Function to train the model for a specific value of `a`
def train_for_a(a, epochs=M):
    model = create_network()
    optim = optimizer(model.parameters(), lr=1e-3)
    for i in range(epochs):
        if i % 100 == 0:
            print(f"Iteration {i} for a = {a}")

        # Choose a random batch of training samples
        indices = torch.randint(0, M, (J,))
        x = x_data[indices, :]
        t = t_data[indices, :]

        x.requires_grad_()
        t.requires_grad_()

        optim.zero_grad()

        # Compute u(0, x) for each x in the batch
        u0_pred = model(torch.hstack((x, torch.zeros_like(t))))
        # Compute the loss for the initial condition
        initial_loss = (u0_pred - phi(x)).square().mean()

        # Compute the partial derivatives using automatic
differentiation
        u = model(torch.hstack((x, t)))
        ones = torch.ones_like(u)
        u_t = grad(u, t, ones, create_graph=True)[0]
        u_x = grad(u, x, ones, create_graph=True)[0]
        u_xx = grad(u_x, x, ones, create_graph=True)[0]

        # Compute the loss for the PDE
        H_pred = H_k(u, a, k)
        pde_loss = (u_t - u_xx - H_pred + u).square().mean()

        # Compute the total loss and perform a gradient step
        loss = initial_loss + pde_loss
```

```
        loss.backward()
        optim.step()

        if i % 100 == 0:
            print(f"Loss at iteration {i} for a = {a}: {loss.item()}")

    return model

# Function to plot the solution at different times
def plot_solution(a_index, a, model):
    mesh = 128
    x = torch.linspace(0, torch.pi, mesh).to(dev).unsqueeze(1)
    t = torch.linspace(0, T, mesh).to(dev).unsqueeze(1)
    x_grid, t_grid = torch.meshgrid(x.squeeze(), t.squeeze(),
indexing="xy")
    x_flat = x_grid.reshape(-1, 1)
    t_flat = t_grid.reshape(-1, 1)
    z = model(torch.cat((x_flat, t_flat), 1))
    z = z.detach().cpu().numpy().reshape(mesh, mesh)
    return a_index, a, z

def save_plot(results):
    gs = GridSpec(2, 4, width_ratios=[1, 1, 1, 0.05])
    fig = plt.figure(figsize=(16, 10), dpi=300)

    # Find the min and max values for color normalization
    z_min = min(result[2].min() for result in results)
    z_max = max(result[2].max() for result in results)

    for a_index, a, z in results:
        ax = fig.add_subplot(gs[a_index // 3, a_index % 3])
        ax.set_title(f"a = {a}")
        im = ax.imshow(
            z, cmap="plasma", extent=[0, torch.pi, 0, T],
aspect='auto', origin='lower', vmin=z_min, vmax=z_max
        )
        ax.set_xlabel('x')
        ax.set_ylabel('t')

    # Add the colorbar to the figure
    sm = ScalarMappable(cmap="plasma", norm=plt.Normalize(vmin=z_min,
vmax=z_max))
    cax = fig.add_subplot(gs[:, 3])
    fig.colorbar(sm, cax=cax, orientation='vertical')

    # Create the directory if it does not exist
    output_dir = "../plots"
    os.makedirs(output_dir, exist_ok=True)
    fig.savefig(os.path.join(output_dir, "ceur.pdf"),
bbox_inches="tight")

if __name__ == '__main__':
    results = []
    for i, a in enumerate(a_values):
        model = train_for_a(a)
        result = plot_solution(i, a, model)
        results.append(result)
    save_plot(results)
```

Source code 1: Modified version of source code from Section 16.3 of [5].

5. Conclusions

In this paper, we have developed and validated an intelligent system methodology for approximating solutions to models of electrical impulse conduction in nerve axons. The core of our approach integrates advanced computational techniques, notably Physics-Informed Neural Networks (PINNs) and the Deep Learning Galerkin Method (DLGM), with biological modeling to address the complexities inherent in these systems.

Our methodology transforms the infinite-dimensional stochastic optimization problems associated with the Hodgkin-Huxley and FitzHugh-Nagumo equations into tractable finite-dimensional problems based on the use of artificial neural networks (ANNs). This approach not only enhances the accuracy and efficiency of the numerical simulations but also provides a robust framework for accounting for the qualitative characteristics of the solutions, such as stability and traveling wave phenomena.

The numerical implementation and empirical results confirm the effectiveness of our method. By successfully approximating the solutions for various parameter values, our method demonstrates its capability to overcome the limitations of classical numerical methods. The results visually and quantitatively affirm that the proposed computational method captures both the quantitative approximation and qualitative behaviors of the model solutions.

Future work will aim to refine the methodology further, exploring more complex models and extending the approach to other types of partial differential equations encountered in biological and medical applications. The adaptability and computational power of intelligent modeling systems promise to further expand the horizons of understanding complex biological processes.

To summarize, the methodology for constructing an intelligent system for approximating solutions to the conduction equations of electrical impulses in nerve axons presented in this study represents the next step forward in the numerical analysis of neuronal dynamics, offering a powerful tool for both theoretical research and practical applications in computational neuroscience.

Acknowledgements

All authors contributed equally to this work. The third author was partially supported by NRFU project No. 2023.03/0074 "Infinite-dimensional evolutionary equations with multivalued and stochastic dynamics." The authors express their gratitude to Prof. Oleksiy Kapustyan for a thorough and productive discussion on the potential use of machine learning methods for approximate solutions of partial derivative equations that admit non-uniqueness of solutions to Cauchy problems.

References

[1] Christian Beck, Martin Hutzenthaler, Arnulf Jentzen, and Benno Kuckuck. An overview on deep learning-based approximation methods for partial differential equations. Discrete and Continuous Dynamical Systems-B, 28(6):3697–3746, 2023.

[2] Hirsh Cohen. Mathematical developments in hodgkin-huxley theory and its approximations. Lect. Math. Life Sci, 8:89–123, 1976.

[3] Richard FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. Biophysical journal, 1(6):445–466, 1961.

[4] Alan L Hodgkin. The conduction of the nervous impulse. Liverpool University Press, 1964.

[5] Arnulf Jentzen, Benno Kuckuck, and Philippe von Wurstemberger. Mathematical introduction to deep learning: methods, implementations, and theory. arXiv preprint arXiv:2310.20360, 2023.

[6] Pavlo O. Kasyanov, Oleksiy V. Kapustyan, Liudmyla B. Levenchuk, and Vladyslav R. Novykov. Machine learning method for approximate solutions for reaction-diffusion equations with multivalued interaction functions. Preprints, July 2024.

[7] Henry P McKean. Stabilization of solutions of a caricature of the fitzhugh-nagumo equation. Communications on Pure and Applied Mathematics, 36(3):291–324, 1983.

[8] Maziar Raissi, Paris Perdikaris, and George 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.

[9] John Rinzel. Simple model equations for active nerve conduction and passive neuronal integration. Lectures on mathematics in the life sciences, 8:125–164, 1976.

[10] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. Journal of computational physics, 375:1339–1364, 2018.

[11] Michael Z Zgurovsky and Pavlo O Kasyanov. Qualitative and quantitative analysis of nonlinear systems. Springer, 2018.

[12] Mikhail Z Zgurovsky, Pavlo O Kasyanov, Oleksiy V Kapustyan, José Valero, and Nina V Zadoianchuk. Evolution Inclusions and Variation Inequalities for Earth Data Processing III: Long-Time Behavior of Evolution Inclusions Solutions in Earth Data Analysis, volume 27. Springer Science & Business Media, 2012.

[13] Mikhail Z Zgurovsky, Valery S Mel'nik, and Pavlo O Kasyanov. Evolution Inclusions and Variation Inequalities for Earth Data Processing I: Operator Inclusions and Variation Inequalities for Earth Data Processing, volume 24. Springer Science & Business Media, 2010.