Learning to Spike: Physics-Informed and Numerical Simulation of Dynamic Neuron Models

Mohammad Badawy, Alaa Essam, Bassel Ahmed, Omar Gamal, Ahmed Abdelrahman, Kareem Mohammad, Amat AlRahman, Engy Wael, Omar Ahmed, Kareem Hassan, Ahmed Salem
Biomedical Engineering Department, Faculty of Engineering, Cairo University, Egypt
Author emails available at: https://tinyurl.com/neuronsemails

***Abstract*—This project presents the simulation of a reduced Izhikevich neuron model using classical numerical solvers, a Physics-Informed Neural Network (PINN), and a real-time 3D visualization interface in Unity. We implemented and compared five methods: Explicit Euler, Backward Euler, Midpoint Runge-Kutta (RK2), ExpRESS-Euler, and a PINN trained on the system's governing ODEs. RK2 achieved a balance between accuracy and computational efficiency, while Backward Euler demonstrated robustness under stiffness but failed to capture sharp spike transitions. The PINN qualitatively reproduced the dynamics, particularly in the recovery variable, though it struggled with steep voltage peaks. Additionally, the Unity-based interactive tool enabled users to explore parameter effects on neuron spiking in real time. This hybrid framework integrates numerical modeling, machine learning, and visualization for a comprehensive understanding of neural excitability.**

***Index Terms*—Dynamic neuron model, Izhikevich model, numerical solvers, Euler method, Runge-Kutta, PINNs, Unity 3D, spiking neurons, biomedical modeling**

I. Introduction

The activity of neurons can be observed by measuring their electrical signals. Understanding this behavior through mathematical modeling is essential in biomedical engineering and computational neuroscience. Traditional biophysical models, such as the Hodgkin-Huxley formulation [1], provide a detailed representation of ion channel dynamics and action potential generation. However, these models are computationally intensive and impractical for large-scale or real-time simulations. To address this, reduced models such as the two-dimensional Izhikevich model [2] offer a balance between biological accuracy and computational efficiency by preserving key neuronal dynamics, including spiking and bursting. In this project, we investigate a simplified Izhikevich-type neuron model introduced by Schiesser, which captures spiking behavior through a reduced system of ordinary differential equations (ODEs). The model is solved using six different numerical methods that vary in complexity and stability, and the results are compared based on how well they handle sharp voltage changes and stiff behavior. In addition, a physics-informed neural network (PINN) is trained to learn the dynamics directly from the equations. Finally, the model is integrated into a Unity 3D environment for real-time, interactive visualization and adjustment of input parameters.

II. Background and Related Work

Modeling neuron behavior is a core challenge in computational neuroscience. The Hodgkin–Huxley model remains a benchmark for biological accuracy, but its complexity makes it computationally intensive. Simpler models like FitzHugh–Nagumo preserve key excitability features with reduced dimensionality [3].

Building on this, Izhikevich proposed a reduced two-variable model that balances biological realism and computational efficiency [2]. Schiesser later reformulated this model to support numerical experimentation and solver analysis [4].

Classical solvers such as Euler, Heun's method, and RK4 are widely used due to their simplicity, while more advanced schemes like RKF45 and RKC4 better handle stiffness and sharp transitions in spiking behavior [5]–[7].

Physics-Informed Neural Networks (PINNs) offer an alternative by embedding the governing equations into the loss function of a neural network, enabling it to learn system dynamics without explicit integration [8]. This approach is part of the broader Physics-Based Deep Learning paradigm, which merges physical laws with data-driven models [9].

Finally, tools like Unity 3D support real-time biomedical visualization, providing an intuitive, interactive way to explore model behavior beyond static plots [10].

This project integrates these domains—mathematical modeling, numerical simulation, physics-informed learning, and interactive visualization—into a unified framework for simulating spiking neuron dynamics.

III. Mathematical Model

The neuron dynamics are described by the following system of ordinary differential equations (ODEs):

$$\frac{dv}{dt} = \frac{k(v - v_r)(v - v_t) - w + I_n(t)}{C}, \quad \frac{dw}{dt} = a[b(v - v_r) - w] \tag{1}$$

where v is the membrane potential, w is the adaptation variable, and $I_n(t)$ is the input current.

A spike-reset condition is applied: if $v \geq 35$ mV, then

$$v \leftarrow c, \quad w \leftarrow w + d$$

TABLE I
NEURON MODEL PARAMETERS (REGULAR SPIKING)

Parameter	Value	Parameter	Value
C	100	a	0.03
v_r	-60	b	-2
v_t	-40	v_{peak}	35
k	0.7	d	100
c	-50	–	–

The input current $I_n(t)$ is defined as a step function:

$$I_n(t) = \begin{cases} 0 & \text{if } t < 101 \text{ ms} \\ 70 & \text{if } t \geq 101 \text{ ms} \end{cases}$$

IV. NUMERICAL METHODS

We evaluated the system using six numerical solvers:

V. NUMERICAL METHODS AND VISUALIZATION RESULTS

- **Explicit Euler Method**
  A first-order explicit scheme for ODEs:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0$$

It estimates the next value using the slope at the current point. It is simple and fast but conditionally stable, requiring small time steps for stiff systems.
Local error: $\mathcal{O}(h^2)$, Global error: $\mathcal{O}(h)$
Algorithm:
  – Compute slope: $k_1 = f(t_n, y_n)$
  – Update: $y_{n+1} = y_n + h \cdot k_1$

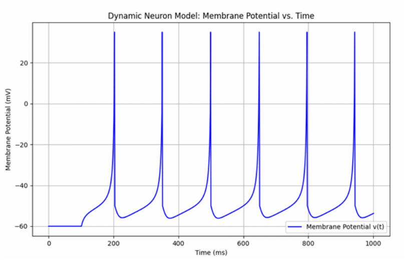

Fig. 1. Explicit Euler: Membrane potential $v(t)$

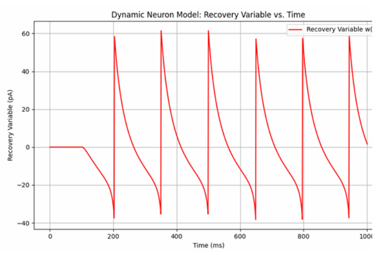

Fig. 2. Explicit Euler: Recovery variable $w(t)$

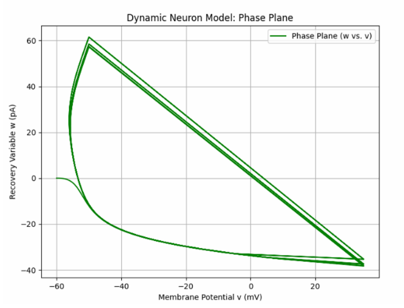

Fig. 3. Explicit Euler: Phase plot w vs v

- **Backward Euler Method**
  An implicit first-order method ideal for stiff problems.

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0$$

Solves for the slope at the end of the step. More stable, but requires solving implicit equations.
Local error: $\mathcal{O}(h^2)$, Global error: $\mathcal{O}(h)$
Algorithm:
  – $y_{n+1} = y_n + h \cdot f(t_{n+1}, y_{n+1})$

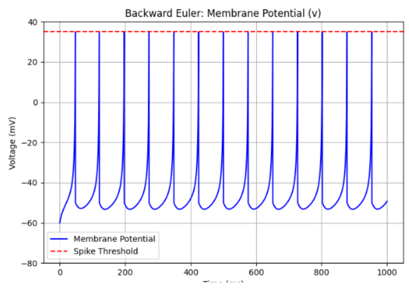

Fig. 4. Backward Euler: Membrane potential $v(t)$

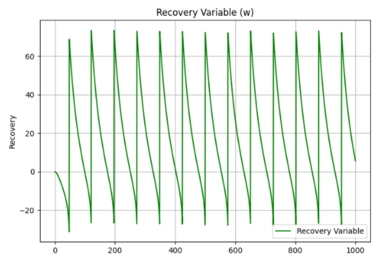

Fig. 5. Backward Euler: Recovery variable $w(t)$

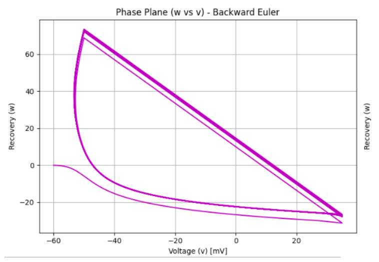

Fig. 6. Backward Euler: Phase plot w vs v

- **Midpoint Runge-Kutta (RK2)**
  A second-order explicit method using midpoint evaluation.

$$k_1 = f(t_n, y_n), \quad k_2 = f(t_n + \tfrac{h}{2}, y_n + \tfrac{h}{2}k_1)$$
$$y_{n+1} = y_n + h \cdot k_2$$

  Local error: $\mathcal{O}(h^3)$, Global error: $\mathcal{O}(h^2)$

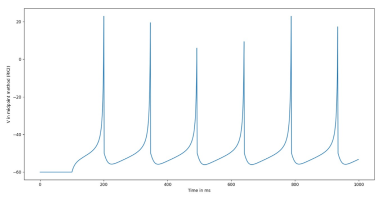

Fig. 7. Rk2 Middle-point: Membrane potential $v(t)$

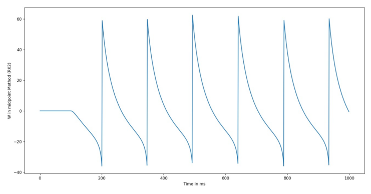

Fig. 8. Rk2 Middle-point: Recovery variable $w(t)$

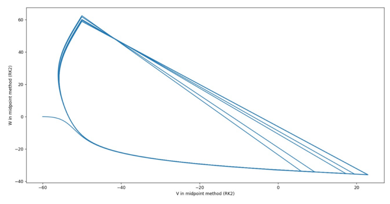

Fig. 9. Rk2 Middle-point: Phase plot w vs v

- **ExpRESS-Euler Method**
  An exponential integrator for stiff systems. It linearizes the system using the Jacobian and applies a matrix exponential.

$$y_{n+1} = y_n + h \cdot \varphi_1(hJ_n) f(t_n, y_n), \quad \varphi_1(z) = \frac{e^z - 1}{z}$$

  Steps:
  - Evaluate $f_n = f(t_n, y_n)$, Jacobian J_n
  - Compute matrix function $\varphi_1(hJ_n)$
  - Update: $y_{n+1} = y_n + h \cdot \varphi_1(hJ_n) f_n$

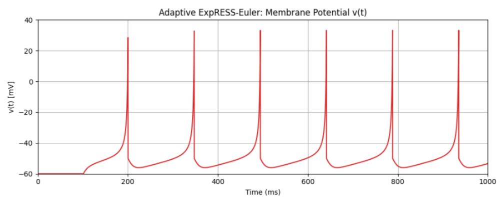

Fig. 10. ExpRESS Euler: Membrane potential $v(t)$

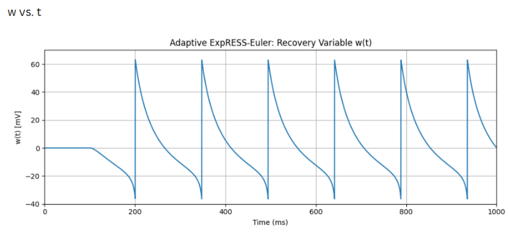

Fig. 11. ExpRESS Euler: Recovery variable $w(t)$

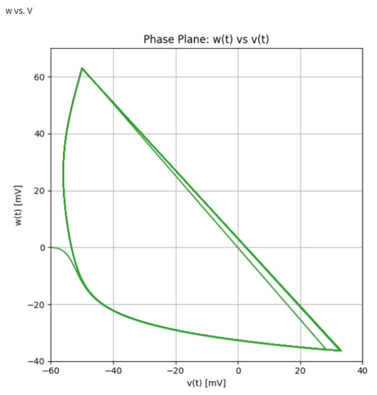

Fig. 12. ExppRESS Euler: Phase plot w vs v

VI. Physics-Informed Neural Network (PINN)

To complement classical solvers, we implemented a Physics-Informed Neural Network (PINN) to learn the time-dependent dynamics of the Izhikevich-type neuron model, directly predicting the state variables $v(t)$ and $w(t)$ from time input.

The network architecture consisted of four fully connected layers with 64 neurons each and `tanh` activations. Training was performed over 5000 epochs using the Adam optimizer.

The loss function combined:

- **ODE residual loss:** Penalizing deviations from the governing differential equations.
- **Initial condition loss:** Enforcing correct values at $t = 0$.

We refined our model over three iterations:

- **First attempt:** Trained solely using the ODE residual and initial condition losses. The model captured general trends but failed to accurately learn the spike shape.
- **Second attempt:** Introduced a data loss term by comparing the PINN output to a numerical reference (Euler method). This improved alignment in subthreshold regions, but spike transitions remained poorly learned.
- **Final approach:** Added a thresholding mechanism that manually resets $v(t)$ and adjusts $w(t)$ once a voltage peak is detected, mimicking the spike-reset behavior in the original model. This significantly improved convergence and accuracy near spike events.

Collocation points were sampled uniformly over the time domain, and gradients were computed via automatic differentiation in PyTorch. Training metrics (loss curves and residuals) were monitored throughout to assess convergence.

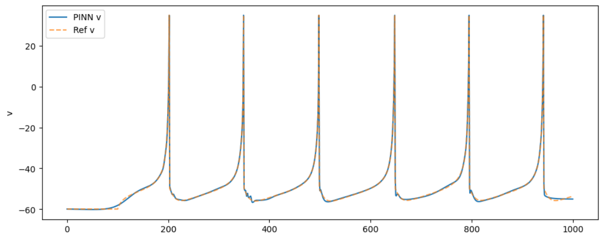

Fig. 13. Predicted $v(t)$

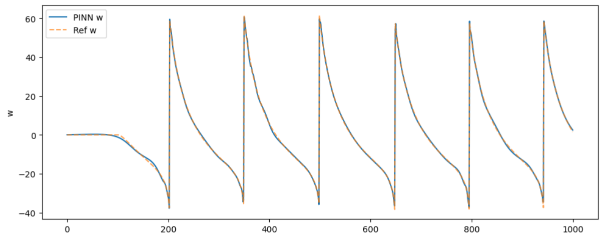

Fig. 14. Predicted $w(t)$

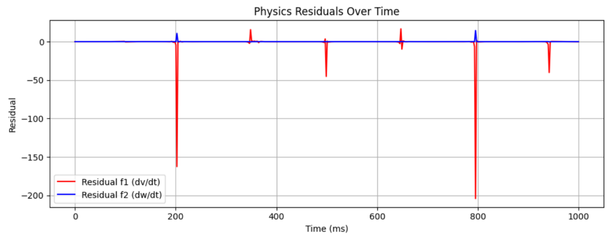

Fig. 15. ODE residual error vs. time

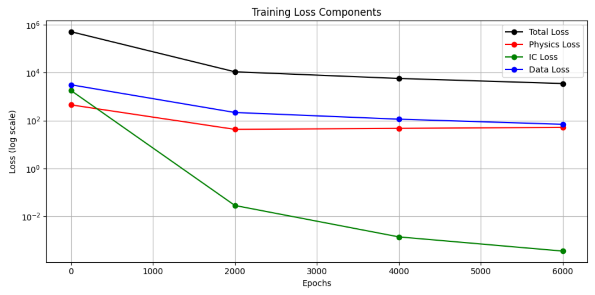

Fig. 16. Training loss convergence

The trained PINN successfully captured the overall spiking dynamics, including the rise and decay timing of and . However, initial versions of the model underestimated peak voltages during rapid transitions — a known challenge due to the stiffness and sharpness of spikes being difficult to learn with smooth activation functions and uniform collocation. In our final iteration, we improved performance by incorporating a data loss term referencing the Euler method solution and applying a reset mechanism that mimicked the model's spike-threshold behavior. This hybrid formulation allowed the network to better align with the model's intrinsic discontinuities and produced a noticeably improved approximation, particularly in the recovery variable and spike timing.

VII. Unity-Based Real-Time Visualization

To enhance interactivity and promote intuitive understanding of spiking behavior, we developed a 3D real-time simulation of the Izhikevich neuron model using the Unity Engine. The simulation allows users to manipulate model parameters and immediately observe how they affect spike generation, providing an engaging educational tool.

Tools Used: Blender, Unity Engine, Visual Studio (C#)

Implementation Steps:

1) A neuron morphology was created in Blender and imported into Unity.
2) The neuron dynamics were implemented in C# using both Euler and RK4 methods for real-time simulation.
3) Parameter sliders were added in the GUI to control key model variables: a, b, c, d, and input current I_n.
4) Spikes were visualized using glowing particles animated along the axon, guided by invisible waypoints placed along the curved 3D path.
5) A toggle system in the UI allowed switching between Euler and RK4 solvers to compare their effect on spike behavior.

Challenges and Solutions:

- *Simulating visual spikes:* Spark-like particle effects were imported from the Unity Asset Store.
- *Animating spikes along curved neuron paths:* A C# script moved particles across empty GameObjects placed along the axon's path.
- *Supporting multiple solvers:* Unity's UI system was used to let users choose the numerical solver at runtime and dynamically update the model.

Output: The final simulation responds instantly to parameter changes, offering users visual feedback on spiking behavior. This makes it especially valuable for visual learners and educators, and provides a platform for exploring how dynamics change under different numerical methods and biological parameter settings.

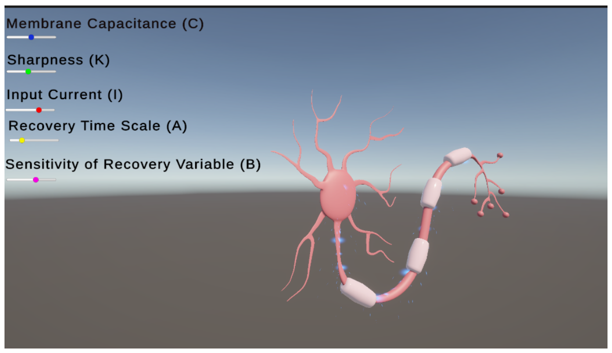

Fig. 17. Unity 3D simulation with parameter sliders and dynamic spike visualization.

VIII. Results and Comparison

The following results compare numerical solvers and the PINN model across five time points, using execution time, error metrics, and qualitative voltage trends. Figure 18 shows membrane potential traces, and Table II summarizes solver performance.

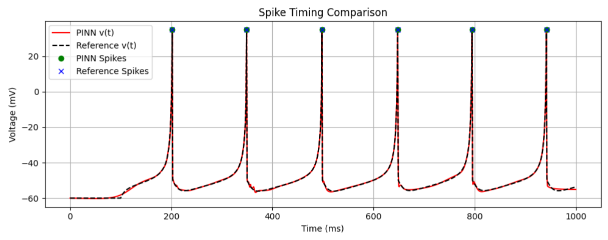

Fig. 18. Spike timing comparison between PINN predictions and reference solution. PINN accurately aligns with spike peaks, demonstrating strong temporal fidelity despite minor deviations during rapid transitions.

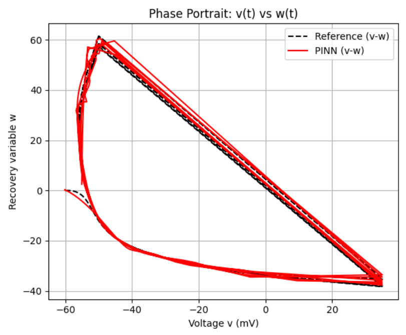

Fig. 19. Phase plane comparison : w vs v

TABLE II
Comparison of Solvers and PINN at Selected Time Points

Method	Time (s)	t (ms)	$v(t)$	$w(t)$	Err v	Err w
Euler	0.558	0	-60.00	0.0000	–	–
		250	-54.48	6.2834	–	–
		500	-50.6154	59.0910	–	–
		750	-49.55	-12.476	–	–
		1000	-53.70	1.5649	–	–
Backward Euler	0.0099	0	-60.00	0.0000	0.000	0.000
		250	-47.52	0.7778	6.965	5.506
		500	35.00	-27.621	85.615	86.712
		750	-52.94	28.878	3.387	41.355
		1000	-49.27	5.6717	4.428	4.107
RK2 Midpoint	0.294	0	-60.00	0.0000	0.000	0.000
		250	-54.37	5.7360	0.198	8.714
		500	-53.59	47.671	5.881	19.326
		750	-48.97	-13.528	1.170	8.433
		1000	-53.18	-0.951	0.956	160.80
ExpRESS-Euler	0.113	0	-60.00	0.0000	0.0000	0.0000
		250	-55.0123	5.950	0.5304	0.3334
		500	-51.0000	60.000	0.3846	0.9090
		750	-50.0000	-13.000	0.4470	0.5237
		1000	-54.0000	2.000	0.3027	0.4351
PINN (ML)	0.0024	0	-59.99	-0.211	0.017	21.055
		250	-54.63	5.3610	0.258	12.662
		500	-51.94	58.359	2.557	1.217
		750	-49.41	-12.014	0.285	3.433
		1000	-54.09	2.2450	0.711	26.524

RK2 balanced speed and accuracy well, making it suitable for real-time applications. Backward Euler managed stiffness but visibly flattened spike amplitudes — a known limitation of implicit solvers. The PINN captured spike timing and general shape but struggled at sharp peaks, as seen in elevated error spikes. However, its accuracy in the slower variable $w(t)$ was strong, highlighting PINNs' potential when supported with methods like adaptive sampling or weighted residual loss.

IX. Conclusion and Future Work

We evaluated numerical and learning-based methods for simulating a reduced Izhikevich neuron model. RK2 offered a favorable balance between accuracy and runtime. In contrast, Backward Euler produced smoothed subthreshold oscillations, reflecting limitations of implicit solvers in capturing fast dynamics. The PINN model effectively tracked slower trends like $w(t)$, but underperformed near spike peaks due to stiffness and sharp transitions.

Future directions include exploring more expressive models (e.g., AdEx), enhancing PINN training with adaptive techniques, and extending to networked neurons with synaptic dynamics. Hybrid solvers and structure-aware learning may further improve accuracy and generalization in stiff regimes.

References

[1] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[2] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[3] E. Schneider and J. Schultheis, "Fitzhugh–nagumo model," Scholarpedia, http://www.scholarpedia.org/article/FitzHugh-Nagumo_model, 2008.

[4] W. E. Schiesser, *Differential Equation Analysis in Biomedical Science and Engineering: Ordinary Differential Equation Applications with R*. John Wiley & Sons, 2014.

[5] E. Fehlberg, "Low-order classical runge-kutta formulas with step size control and their application to some heat transfer problems," NASA Technical Report R-315, Tech. Rep., 1969. [Online]. Available: https://maths.cnam.fr/IMG/pdf/RungeKuttaFehlbergProof.pdf

[6] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer-Verlag, 1993.

[7] P. J. van der Houwen and B. P. Sommeijer, "Stabilized explicit methods for stiff odes: applications to circuit simulation," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 77–86, 1980.

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

[9] N. Thuerey, P. Holl, X. Hu, T. Pfaff, K. Um, and S. Wiewel, "Physics-based deep learning," *arXiv preprint arXiv:2109.05237*, 2021.

[10] R. Kazmi *et al.*, "Using unity for immersive biomedical visualization," *IEEE Computer Graphics and Applications*, vol. 41, no. 5, pp. 78–87, 2021.