Dynamic Ion Channel Modeling Within NeuroVISOR

Carlos Eckert

May 28, 2026

Abstract

NeuroVISOR, a virtual reality platform for neuronal simulation, previously supported only the classical two-channel Hodgkin-Huxley model, limiting exploration of multi-channel dynamics underlying neuronal behavior. We extended NeuroVISOR to a dynamic architecture supporting six physiologically relevant ion channels: fast sodium (Na), delayed-rectifier potassium (K), leak, high-threshold calcium (CaH), low-threshold T-type calcium (CaT), and slow potassium M-current (M). The implementation features real-time channel toggling via an interactive VR panel, enabling users to activate or deactivate channels during simulation to observe immediate effects on membrane dynamics. We developed a Python-based Forward Euler voltage solver with modular channel definitions and validated the Euler Program model against the Yale NEURON reference platform across eight channel configurations. Validation using inter spike interval (ISI) and firing frequency analysis demonstrated ISI agreement within 0.02 ms and frequency matching within 1 Hz, with 100% spike count agreement across all configurations. Additionally, we implemented an injected current electrode stimulus protocol to complement existing voltage clamp and synaptic stimulation methods. The multi-channel framework revealed distinct electrophysiological signatures: calcium channels increased firing frequency and enabled pacemaker activity, while M-current produced spike-frequency adaptation. This advancement transforms NeuroVISOR into a computational tool for researchers and students to seamlessly discover properties of neuronal excitability through interactive exploration of ion channel contributions.

1 Introduction

The quantitative description of neuronal excitability established by Hodgkin and Huxley in 1952 [HH52] revolutionized computational neuroscience by demonstrating that action potentials emerge from the interplay of voltage-gated sodium and potassium ion channels. This foundational model provided the mathematical framework for understanding membrane dynamics, yet real neurons express diverse channel types that shape complex firing patterns beyond the classical two-channel paradigm. High-threshold calcium channels enable plateau potentials and neurotransmitter release, T-type calcium channels mediate rebound excitation and pacemaker activity, and M-type potassium channels govern spike-frequency adaptation and gain control. Understanding how these channels interact to produce neuronal behaviors is essential for both neuroscience education and research.

Traditional computational neuroscience platforms such as NEURON [HC01] and Brian2 [SM19] provide powerful simulation capabilities but require substantial programming expertise, creating a barrier for students and researchers seeking to explore channel dynamics. NeuroVISOR addresses this limitation by providing an interactive virtual reality platform where users manipulate neuronal parameters through spatial interfaces rather than code, enabling direct engagement with biophysical mechanisms. However, the initial NeuroVISOR implementation supported only the classical Hodgkin-Huxley sodium and potassium channels, preventing exploration of calcium dynamics, adaptation mechanisms, and other phenomena that require multi-channel architectures.

The restriction of two hard-coded ion channels limited the users' ability to investigate how channel composition shapes neuronal excitability. Students could not observe how M-current produces spike-frequency adaptation, how calcium channels enable persistent firing, or how channel combinations create varying patterns. Without the ability to toggle individual channels and observe their contributions in real-time, NeuroVISOR could not serve as a comprehensive tool for discovering the biophysical principles governing neuronal behavior. Extending the platform to support arbitrary ion channel configurations was therefore essential to fulfill its educational and research potential.

This work addresses these limitations through three primary contributions:

- **Extensible multi-channel architecture:** We redesigned NeuroVISOR's computational framework to support arbitrary ion channel configurations, implementing six physiologically relevant channels (fast sodium, delayed rectifier potassium, leak, high-threshold calcium, T-type calcium, and M-current) based on the Pospischil et al. [P+08] neuron model.
- **Interactive VR channel control:** We developed an ion channel panel enabling real-time toggling of individual channels during simulation, allowing users to observe immediate effects on membrane dynamics and systematically isolate channel contributions.
- **Rigorous validation framework:** We validated the extended implementation against the Yale NEURON reference platform using interspike interval and firing frequency analysis across eight channel configurations, establishing computational fidelity and identifying channel-specific electrophysiological signatures.

Additionally, we implemented an injected current electrode to complement existing voltage clamp and synaptic stimulation methods - providing researchers, educators, and students with an additional external stimulus.

2 Background

2.1 Neuronal Excitability and the Action Potential

Neurons communicate through rapid changes in membrane voltage known as action potentials. At rest, the neuronal membrane maintains a resting potential of approximately -70 mV, established by the selective permeability of the membrane to potassium ions. When a sufficiently large depolarizing stimulus is applied, the membrane voltage rises past a threshold of approximately -50 mV, triggering a spike in voltage within the neuron, known as an action potential. During an action potential, the neuron fires to around $+50$ mV followed by a hyperpolarization before returning to rest. This all-or-nothing event is the fundamental unit of information transmission in the nervous system.

The biophysical basis of action potential generation was established by Hodgkin and Huxley in 1952 through landmark experiments on the squid giant axon [HH52]. They demonstrated that the action potential arises from the coordinated activity of two voltage-gated conductances: a fast inward sodium current (I_{Na}) responsible for depolarization, and a delayed outward potassium current (I_{K}) responsible for repolarization. Their formulation expressed membrane dynamics as a system of ordinary and partial differential equations, providing the first quantitative framework for neuronal excitability. This Hodgkin-Huxley (HH) framework remains the foundation of modern computational neuroscience.

2.2 Ion Channels and Conductance-Based Models

Ion channels are transmembrane proteins that selectively permit the passage of specific ionic species (Na^+, K^+, Ca^{2+}, Cl^-) down their electrochemical gradients. Their opening and closing is governed by the membrane voltage, giving rise to the voltage-dependent conductances described by HH kinetics. Each channel type contributes a current of the general form:

$$I_j = \bar{g}_j \, m^M h^N (V - E_j) \tag{1}$$

where $\bar{g}_j$ is the maximal conductance density ($\mathrm{S/m^2}$), m and h are dimensionless activation and inactivation gating variables ranging from 0 to 1, M and N are integer powers reflecting the number of independent gating subunits, and E_j is the reversal potential for the ion.

Real neurons express a diverse combination of ion channels beyond the classical sodium and potassium channels. High-threshold calcium channels (I_{CaH}) create burst firing and calcium-dependent neurotransmitter release. T-type calcium channels (I_{T}) activate at subthreshold voltages and mediate rebound excitation and low-threshold burst firing following hyperpolarization. Slow non-inactivating potassium channels (I_{M}, the M-current) activate with depolarization and produce spike-frequency adaptation by progressively hyperpolarizing the membrane during sustained firing. The interplay between these channel types determines the full range of neuronal firing patterns observed in the nervous system, from regular spiking and fast spiking to intrinsic bursting and low-threshold spiking [P+08]. A simulation platform that restricts users to only sodium and potassium channels is incapable of reproducing the majority of physiologically relevant neuronal behaviors.

2.3 The Need for Multi-Channel Extension

Before this work, NeuroVISOR hardcoded sodium and potassium channels as fixed components of the `SparseSolver`, with gating variables m, n, and h explicitly named and updated within the solver loop. While sufficient for demonstrating classic HH dynamics, this architecture consisted of three fundamental limitations.

First, the biological scope of the simulation was constrained to fast-spiking behavior driven by Na^+/K^+ dynamics alone. Phenomena, including spike-frequency adaptation, intense spiking, rebound excitation, and pacemaker activity, were inaccessible. This limited NeuroVISOR's utility as a research and educational tool, as the majority of physiologically distinct neuron classes depend on channels beyond the basic HH pair [P+08].

Second, any extension of the channel set required direct modification of the solver source code, creating a tightly coupled architecture that was fragile, difficult to maintain, and inaccessible to non-programmer users. There was no mechanism by which a researcher or student could add or explore a new channel type without recompiling the Unity project.

Third, the absence of configurable channel toggling prevented the kind of systematic channel manipulation that is standard practice in both experimental neuroscience (pharmacological blockade of specific channels) and computational neuroscience (parameter sensitivity analysis). The addition of a dynamic ion channel architectures allows NeuroVISOR to be a meaningful tool for investigating neuronal excitability.

The refactored architecture with `IonChannel` and `GatingVariable` abstractions, dynamic `activeIonChannels` collections, and a real-time VR toggle panel resolves all three limitations. It decouples channel definitions from solver logic, extends the biophysical scope to six physiologically relevant channel types, and provides an intuitive interface for channel manipulation that requires no programming expertise. This transition from a hardcoded two-channel simulator to a dynamic multi-channel architecture represents the core engineering contribution of this work.

3 Models

3.1 Membrane Voltage Dynamics

The membrane voltage $V(t)$ evolves according to the capacitor equation:

$$C_m \frac{dV}{dt} = I_{\text{ext}}(t) - I_{\text{ion}}(V, \mathbf{x}) \tag{2}$$

where $C_m = 1.0 \times 10^{-2}$ F/m^2 is the specific membrane capacitance, $I_{\text{ext}}(t)$ is the summation of external stimuli (A/m^2), I_{ion} is the total ionic current density, and $\mathbf{x}$ represents the vector of gating variables. The total ionic current is the sum of individual channel contributions:

$$I_{\text{ion}} = I_{\text{Na}} + I_{\text{K}} + I_{\text{Leak}} + I_{\text{CaH}} + I_{\text{T}} + I_{\text{M}} \tag{3}$$

where each channel follows the structure of Equation 1.

3.1.1 Ion Channel Structure

All channel kinetics are derived from Pospichil [P+08], which provides minimal Hodgkin-Huxley type models fit to experimental data from cortical and thalamic neurons. We implemented six physiologically relevant channel types as summarized in Table 1.

Channel	Current	$\bar{g}$ (S/m^2)	E (mV)	Gates	Powers
Fast sodium	I_{Na}	50	$+50$	m, h	m^3h
Delayed rectifier K	I_{K}	5.0	-90	n	n^4
Leak	I_{Leak}	0.0001	-70	—	—
High-threshold Ca^{2+}	I_{CaH}	1.0	$+120$	q, r	q^2r
T-type Ca^{2+}	I_{T}	4.0	$+120$	s, u	s^2u
Slow K^+ (M-current)	I_{M}	0.75	-90	p	p

Table 1: Ion channel parameters for all six implemented channels. All kinetics from Pospischil [P+08]. Conductances given in S/m^2; reversal potentials in mV.

The structural uniformity of Equation 1 across all six channel types is what makes the looped architecture of the refactored `SparseSolver` possible: every channel can be evaluated by the same generic computation, differing only in its parameters and gating variable definitions.

The delayed rectifier potassium channel is presented here as a representative example. Its current is:

$$I_{\text{K}} = \bar{g}_{\text{K}}\, n^4 (V - E_{\text{K}}) \tag{4}$$

where the activation gate n evolves according to first-order kinetics with voltage-dependent rate functions. With V in millivolts and threshold shift $V_T = -50.0$ mV:

$$\alpha_n(V) = \frac{-0.032\,(V - V_T - 15)}{\exp[-(V - V_T - 15)/5] - 1}$$

$$\beta_n(V) = 0.5 \exp\left[\frac{-(V - V_T - 10)}{40}\right]$$

All remaining channel equations follow the same structural form with channel-specific rate functions as defined in Pospischil [P+08].

A special case arises in the T-type calcium channel, where the activation variable s is treated as instantaneous rather than integrated. Its steady-state value:

$$s_\infty(V) = \frac{1}{1 + \exp[-(V + V_x + 57)/6.2]}$$

is evaluated directly at each timestep rather than solved through a differential equation. This distinction is consequential for the solver architecture and is addressed explicitly in Section 3.2.

3.1.2 Gating Variable Dynamics

Gating variables are probabilistic rate constants that evolve at a given time t:

$$\frac{dx}{dt} = \alpha_x(V)(1 - x) - \beta_x(V)(x) \tag{5}$$

where $\alpha_x(V)$ and $\beta_x(V)$ are values between 0 and 1. These rate functions determine the steady-state activation:

$$x_\infty(V) = \frac{\alpha_x(V)}{\alpha_x(V) + \beta_x(V)} \tag{6}$$

This steady-state activation function enables NeuroVISOR to initialize the gating variables accurately in accordance with the neuronal voltage at the time of gating variable activation.

3.2 NeuroVISOR Architecture Modifications

Within NeuroVISOR's `SparseSolver`, the HH gating variables m, n, and h are explicitly named `Vector` objects. The `reactF` function computed ionic current by evaluating $\bar{g}_{\mathrm{K}} n^4 (V - E_K)$, $\bar{g}_{\mathrm{Na}} m^3 h (V - E_{\mathrm{Na}})$, and $\bar{g}_{\mathrm{L}} (V - E_L)$ for I_K, I_{Na} and, I_L respectively – returning their sum scaled by $-1/C_m$. In `SolveStep`, the gating variable updates were issued sequentially and by name: the SBDF2 explicit update was called once for m, once for n, and once for h, with their corresponding rate functions written directly into each call. Any additional channel required a programmer to modify `reactF`, insert a corresponding update block in `SolveStep`, and add the resulting equation to the output. There was no abstraction separating what a channel is from how the solver processes it.

We redesigned this architecture around two new class abstractions: `IonChannel` and `GatingVariable`. The `IonChannel` class consists of a channel's maximal conductance $\bar{g}$, reversal potential E, and the list of `GatingVariable` objects associated with it. A `GatingVariable` consists of its name, voltage-dependent $\alpha(V)$ and $\beta(V)$ rate functions, its initial state at the starting voltage, its integer exponent, and a boolean `IsInstant` flag. All six channel definitions were moved to a separate `IonChannelModels` configuration file, which instantiates each channel object and is the only file a future developer needs to modify in order to add a new channel type.

3.2.1 Channel Lists and State Dictionaries

The solver maintains two lists: `ionChannels`, which holds all available channels from the `IonChannelModels` config file, regardless of whether they are currently active, and `activeIonChannels`, which holds only those currently enabled by the user. The explicit vector declarations for m, n, and h were replaced with two generic `Dictionary<string, Vector>` objects - `currentStates` and `previousStates` - keyed by gating variable name. At initialization, `InitializeNeuronCell` iterates over `activeIonChannels` and populates both dictionaries with a vector of size equal to the neuron's node count, set to each variable's initial probability at the starting voltage. This means that adding a new channel with two gating variables requires no changes to the initialization logic; the loop handles it automatically.

3.2.2 Refactored `reactF`

`reactF` is responsible for computing Equation1 for each ion channel and summing the ion contributions to an output. In the original implementation, `reactF` evaluated three explicitly named current terms. In the refactored solver, `reactF` receives the `activeIonChannels` list and `currentStates` dictionary, and computes the total ionic current through a nested loop. For each channel in `activeIonChannels`, the function initializes a unit product vector and iterates over the channel's gating variables. For each variable, it retrieves the corresponding state vector from the dictionary and raises it to the variable's exponent. The product is then multiplied by $\bar{g}$ and $(V - E_j)$, and accumulated into the output vector. The leak channel is handled as a special case, contributing $\bar{g}_L (V - E_L)$ directly without a gating variable product. The final output is scaled by $-1/C_m$. The result is a `reactF` that is fully channel-agnostic: it requires no knowledge of which specific channels are active.

3.2.3 Refactored `SolveStep`

The `SolveStep` function with `SparseSolver` is responsible for constructing the neuronal voltage matrix over time and space. The explicit sequential calls to `stateexplicitSBDF2` for m, n, and h were replaced with a double loop. `SolveStep` now iterates over each channel in `activeIonChannels`, and for each channel iterates over its `GatingVariables`. For each gating variable, it retrieves the current state from `currentStates`, saves a clone to a temporary buffer, calls `stateexplicitSBDF2` with the appropriate α and β functions evaluated at the current and previous voltage, and updates `previousStates` from the saved clone. This loop structure means that the solver's time-stepping logic is completely decoupled from the specific gating variables present in any given simulation.

3.2.4 Channel Toggling

The `ToggleChannel` method enables real-time channel switching during a running simulation. When a channel is deactivated, its `IonChannel` object is removed from `activeIonChannels`. When a channel is activated, it is added to `activeIonChannels` and its gating variable states are reinitialized from the

neurons current voltage at the time of reactivation. The ion channel panel in the VR environment communicates directly with `ToggleChannel` - a function with `SparseSolver` which checks for changes with activeIonChannels - providing a seamless path from user interaction to solver behavior.

4 Validation

4.1 Validation Strategy

The final objective for validation is for NeuroVISOR, with its adjusted multi-channel architecture, to be computationally equivalent to Yale NEURON [HC97], the established standard in computational neuroscience. The validation of a novel simulation platform against a single external reference introduces the risk of inheriting systematic errors that exist in that reference but not in the underlying model. To address this, we designed a three-platform strategy: a Python Forward Euler solver was constructed as a transparent, equation-level implementation; its outputs were verified against NEURON; and NeuroVISOR was validated against NEURON. Agreement across all three platforms allows discrepancies to be localized to a specific implementation.

The three platforms are not equivalent in their integration approach. Python Euler treats the neuron as a single-point compartment, solving only ordinary differential equations (ODEs) for voltage and gating variables with no spatial dimension. NeuroVISOR solves the cable equation, a partial differential equation (PDE) that accounts for current propagation along neuron geometry using the SBDF2 method - a second-order implicit-explicit backward differentiation formula. For validation purposes, this distinction is deliberately neutralized: the NeuroVISOR neuron is a single-compartment cylindrical soma of one segment, which collapses the PDE to an ODE equivalent. Yale NEURON operates in the same single-compartment regime when `nseg = 1`. The three platforms should therefore converge to the same solution, and the degree of convergence is what the validation quantifies.

4.2 Validation Environment

4.2.1 Soma Geometry

Meaningful cross-platform comparison requires that all three simulators operate on an identical neuron. Membrane current is computed as a current density ($\mathrm{A/m^2}$), meaning that the stimulus amplitude and ionic conductances are inherently area-dependent. If the soma area differs between platforms, the effective per-unit-area current differs, and the resulting spike train cannot be directly compared. Therefore, standardizing geometry was a prerequisite for validation.

We modeled the soma as a cylinder with radius $r = 16.75$ µm and length $L = 0.25187969924812$ µm, yielding a surface area of:

$$A_{\text{soma}} = 2\pi rL = 2.6508664327001058 \times 10^{-11}\ \mathrm{m}^2$$

These dimensions correspond to the soma geometry used in a Pospischil et al. 2008 [P+08] program, which the authors specified as a cylinder of diameter d equal to its length L. In NEURON, the soma was created as a soma with equivalent dimensions. For NeuroVISOR, a morphology file was constructed from scratch in SWC format, defining a neuron with the same cylindrical dimensions and an axon of the size 0.00000001 m^2 since NeuroVISOR returned an error when loading a single-compartment neuron. This ensured that the soma area computed by the SBDF2 solver matched Equation 4.2.1.

4.2.2 Integration Methods and Timesteps

As mentioned, the three platforms differ in their integration strategy, and these differences informed the choice of timestep for each. Python Euler applies the explicit Forward Euler scheme at a fixed timestep of `dt = 50e-4` ms. Yale NEURON integrates the same single-compartment ODE system using a fixed timestep of `dt = 50e-5` ms. The NeuroVISOR timestep is computed adaptively at initialization based on active channel conductances, soma geometry, and membrane capacitance. All three platforms initialized membrane voltage to $V_0 = 0$ mV for testing, but NeuroVISOR and NEURON initialize $V_0 = -70$ mv in the results to simulate biophysically realistic neurons.

4.3 Stimulus Development

The current injection stimulus followed a development arc that proceeded in lockstep with the validation pipeline: it was implemented first in Python Euler, reproduced in Yale NEURON, adapted into NeuroVISOR through a temporary workaround, and finalized as an independent VR feature.

Python Euler. In the Python program, the function `I_stim(t)` evaluates a time condition and returns a fixed current density:

$$I_{\text{stim}}(t) = \begin{cases} 0.15 \times 10^{-11} \,/\, A_{\text{soma}} & 50 \text{ ms} \leq t < 150 \text{ ms} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The total injected current is 1.5×10^{-12} A (1.5 pA), divided by `soma_area` to yield a current density in A/m^2 that is add to the right-hand side of Equation 2. Thus, the stimulus is a conditional expression evaluated at every timestep.

Yale NEURON. To reproduce the same stimulus in NEURON, we placed an `IClamp` object at the midpoint of the soma section (`soma(0.5)`). The total injected current of 1.5 pA was converted to nA for the NEURON interface (`amp = 0.0015` nA), with `delay = 50` ms and `dur = 100` ms to match the Euler window. Because NEURON works in physical current units (nA) rather than current density, no area normalization was required at the stimulus level; the conversion is handled internally by NEURON's area-based conductance formulation.

NeuroVISOR - Synapse Adaptation. Introducing a current stimulus into NeuroVISOR during the validation phase required a practical interim solution before a dedicated electrode feature existed. The existing synaptic current pathway in `SynapseCurrentFunction` passes a list `Icurrs` containing current and previous timestep current values to the right-hand side of the voltage equation via `SetSynapseCurrent`. By setting both entries of `Icurrs` to the same fixed amplitude matching the Euler stimulus value and returning immediately, the synapse infrastructure became a de facto current clamp. This approach required placing two soma objects in the VR scene - one to serve as the presynaptic source, one as the postsynaptic target - which was adequate for controlled validation experiments, but not suitable as a general-purpose user feature.

NeuroVISOR - Independent Electrode. The finalized stimulus implementation is a dedicated current injection electrode, visible as a physical object in the VR environment. The user places it on any vertex of the neuron mesh, and upon activation it contributes a constant I_{stim} directly to the right-hand side of Equation 2 without routing through the synaptic pathway.

4.4 Python Forward Euler Implementation

With the soma geometry and stimulus established, the Python Forward Euler solver integrates the full state vector:

$$\mathbf{y} = [V,\ m,\ n,\ h,\ p,\ q,\ u,\ r]^T \tag{8}$$

using the explicit scheme:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \cdot f(\mathbf{y}_n, t_n) \tag{9}$$

where $f(\mathbf{y}_n, t_n)$ evaluates the right-hand sides of Equations 2 and 5 simultaneously.

Channel toggling is implemented via boolean flags (`USE_NA`, `USE_K`, `USE_LEAK`, `USE_CaH`, `USE_T`, `USE_M`). When a flag is false, that channel's gating variable derivatives are held at zero and its contribution is excluded from I_{ion}, mirroring the `activeIonChannels` mechanism in NeuroVISOR. Each simulation run exports a CSV file containing time in milliseconds and voltage in millivolts, and generates a two-panel plot showing the voltage trace and active gating variable trajectories. These CSV files serve as direct inputs to the MATLAB comparison pipeline described in Section 4.7.

4.5 Yale NEURON

Yale NEURON simulations were conducted using custom channel mechanisms (`.mod` files) implementing the same Pospischil kinetics as the Python and NeuroVISOR implementations. Each mechanism inserts its channel type into the soma section and exposes its conductance as a tunable parameter. Conductances were converted from S/m^2 to mS/cm^2 for the NEURON interface. Channel toggles are implemented by selectively calling `sec.insert()` in the setup script, matching the boolean flag structure of the Python solver. Simulation output is recorded and exported as a CSV file in the same format as the Python Euler output, enabling the MATLAB comparison pipeline to treat both files identically.

4.6 Initial Conditions

Gating variables were initialized at steady-state for the resting potential $V_0 = 0$ mV across all three platforms:

$$x_0 = x_\infty(V_0) = \frac{\alpha_x(V_0)}{\alpha_x(V_0) + \beta_x(V_0)} \tag{10}$$

Consistent initialization across all three platforms is a prerequisite for meaningful comparison: differing initial conditions introduce immediate deviations at simulation onset that are unrelated to solver accuracy.

4.7 Quantitative Metrics and Comparison Pipeline

4.7.1 MATLAB Comparison Pipeline

Cross-platform comparison was performed using a MATLAB script (`voltage_comparison.m`) written for this project. The script iterates over all eight channel configurations, loading the corresponding pair of CSV files (e.g., `nv_na_k_leak.csv` and `yale_na_k_leak.csv`), interpolating both traces onto a common time axis, and computing all validation metrics. For each configuration, the script generates a three-panel figure: the top panel overlays the two voltage traces, the middle panel plots ISI across successive spike intervals for both platforms, and the bottom panel shows the ISI drift. All figures presented in this section are outputs of this pipeline and Euler's voltage/gating states plot.

During development, we initially compared voltage traces by visual inspection of the overlaid plots - an eye-norm approach that was sufficient for rapid iteration and identifying gross discrepancies. We subsequently introduced root-mean-square error (RMSE) as a quantitative supplement. However, RMSE is sensitive to amplitude and phase differences that arise from differing integration methods and timesteps, and does not directly reflect the functional output of the neuron model. We therefore transitioned to interspike interval analysis as the primary metric, which is both more physiologically meaningful and more widely used in the computational neuroscience community for comparing contrasting spike trains.

4.7.2 Validation Metrics

Interspike Interval Analysis Spikes were detected via threshold crossing at $V > 35$ mV in both traces. ISI was computed as the time elapsed between consecutive spike threshold crossings. Mean ISI difference quantifies systematic timing drift between platforms:

$$\Delta\text{ISI} = \text{ISI}_{\text{NV}} - \text{ISI}_{\text{ref}} \tag{11}$$

$$\text{ISI}_{\text{mean}} = \frac{\sum \text{ISI}}{\text{ISI}_{\text{total}}} \tag{12}$$

Firing Frequency Instantaneous firing frequency for each interspike interval i was calculated as:

$$f_i = \frac{1000}{\text{ISI}_i} \text{ Hz} \tag{13}$$

where ISI_i is in milliseconds. Mean firing frequency was computed by averaging f_i across all intervals.

Statistical Reporting For each channel configuration, we report mean ISI, mean firing frequency, and ISI difference between platforms. No formal hypothesis testing was performed.

5 Results

5.1 Validation Results

Figures 1–3 present the Python Forward Euler output for three representative channel configurations, showing membrane voltage and active gating variable trajectories over the 200 ms simulation window with $V_0 = 0$ mV, reflecting the initial development configuration used to verify channel dynamics before the final validation setup was established. These figures establish the expected physiological behavior of each configuration before cross-platform comparison. Figures 4–12 present the cross-platform validation results for all eight configurations between Yale NEURON and NeuroVISOR with $V_0 = -70$ mV. Figure 4 shows the voltage trace overlay between NEURON and NeuroVISOR for the baseline configuration. Figures 5–12 present the full three-panel MATLAB output for all eight configurations: voltage overlay in the top panel, ISI per spike interval in the middle panel, and ISI drift ($\Delta\text{ISI} = \text{ISI}_{\text{NV}} - \text{ISI}_{\text{Yale}}$) in the bottom panel. All figures are outputs of the Python Euler solver and `voltage_comparison.m`, respectively.

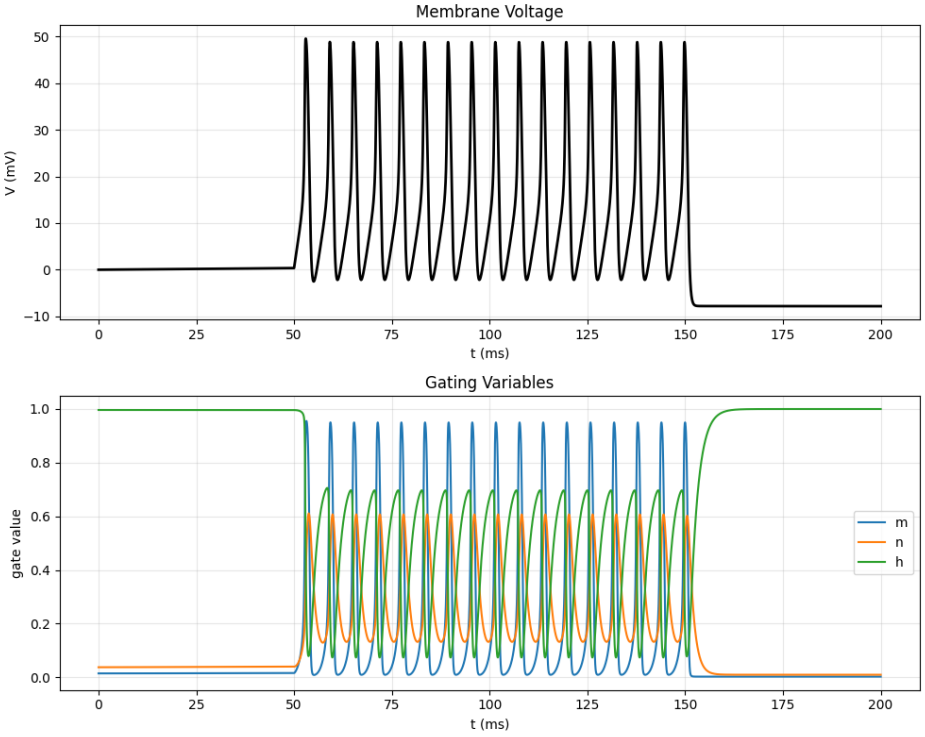

Figure 1: Python Forward Euler output for the Na+K+Leak configuration. Top panel: membrane voltage over 200 ms with stimulus applied 50–150 ms. Bottom panel: active gating variable trajectories (m, n, h). This is the baseline Hodgkin–Huxley configuration.

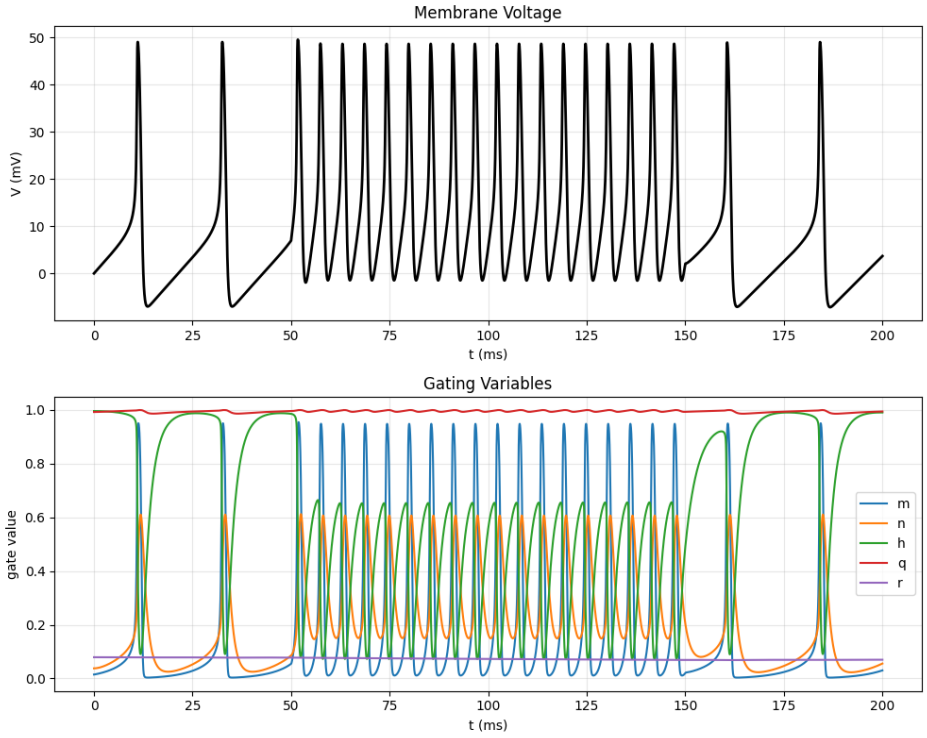

Figure 2: Python Forward Euler output for the Na+K+Leak+CaH configuration. Top panel: membrane voltage showing increased firing frequency relative to baseline. Bottom panel: active gating variables including high-threshold calcium gates q and r.

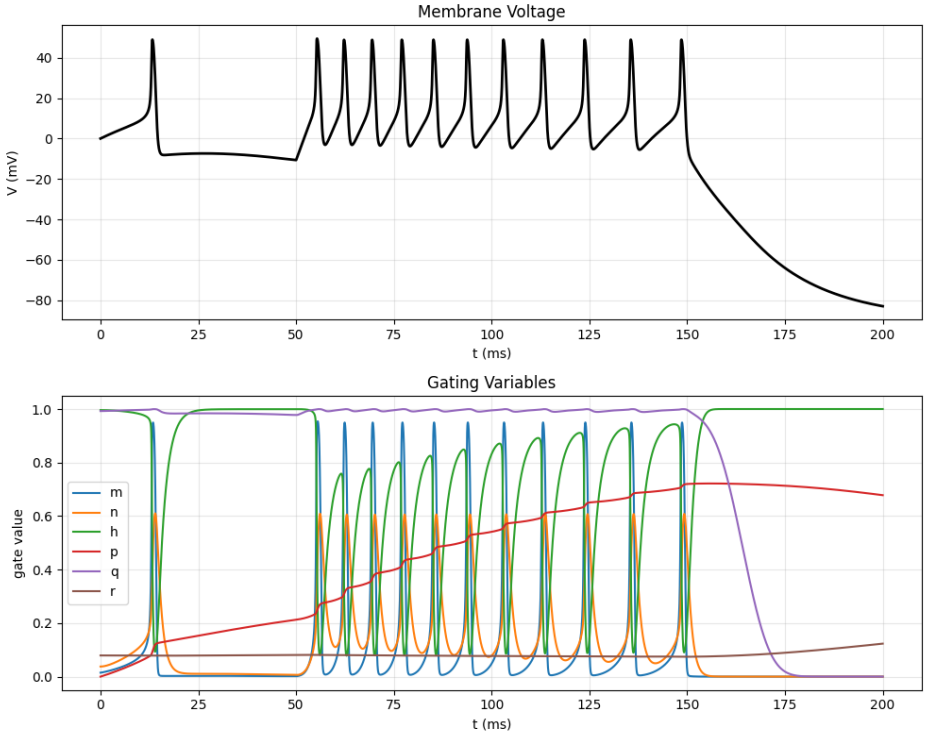

Figure 3: Python Forward Euler output for the full six-channel configuration (Na+K+Leak+CaH+T+M). Top panel: membrane voltage showing the combined effect of all six channels. Bottom panel: all active gating variables (m, n, h, p, q, r, u).

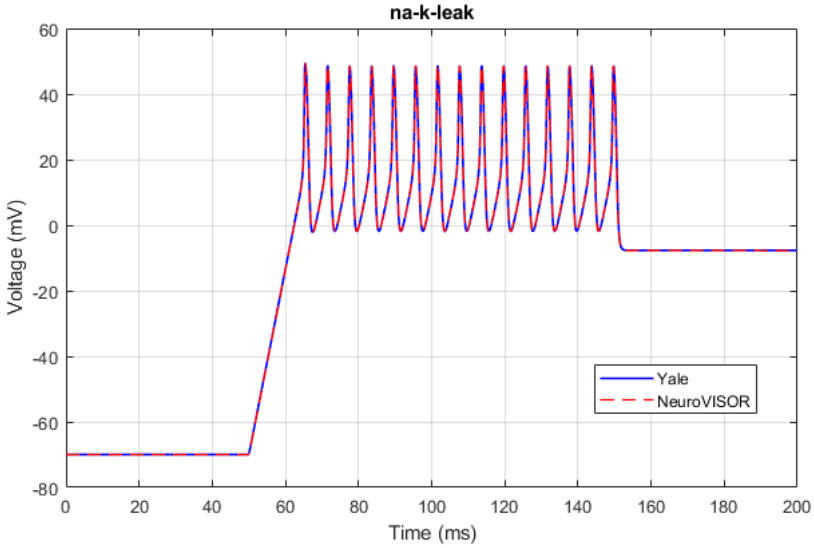

Figure 4: Voltage trace comparison between Yale NEURON (solid blue) and NeuroVISOR (dashed red) for the Na+K+Leak configuration over 200 ms. Stimulus applied 50–150 ms. The two traces are visually indistinguishable across the full simulation window.

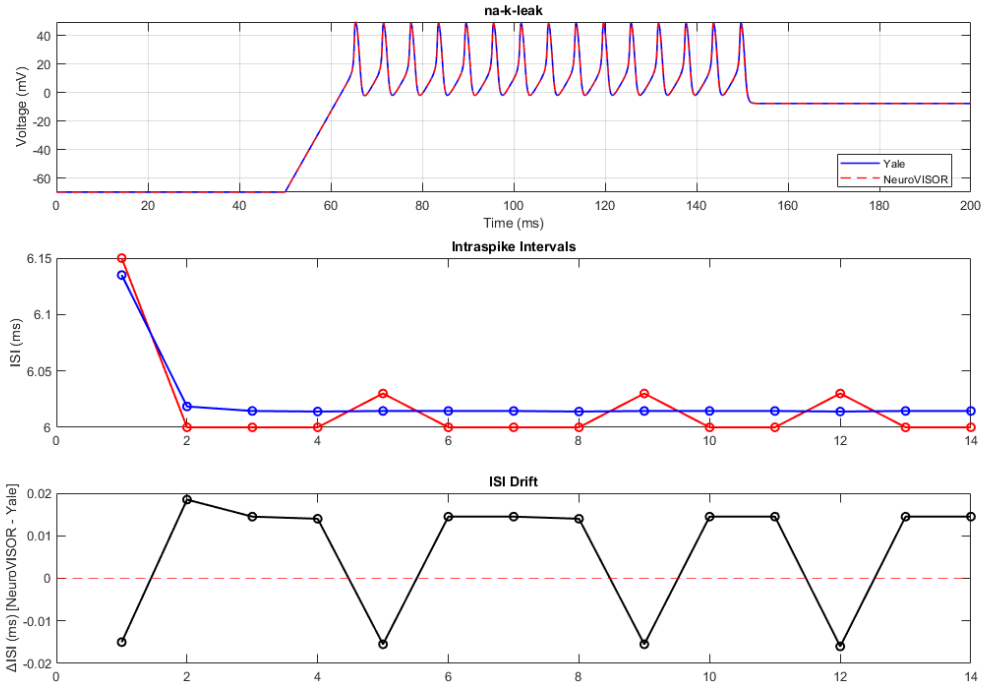

Figure 5: Three-panel validation for Na+K+Leak. Top: voltage overlay (Yale NEURON solid blue, NeuroVISOR dashed red). Middle: ISI per spike interval for both platforms. Bottom: ISI drift (ΔISI = $\mathrm{ISI_{NV}} - \mathrm{ISI_{Yale}}$). Stimulus applied 50–150 ms.

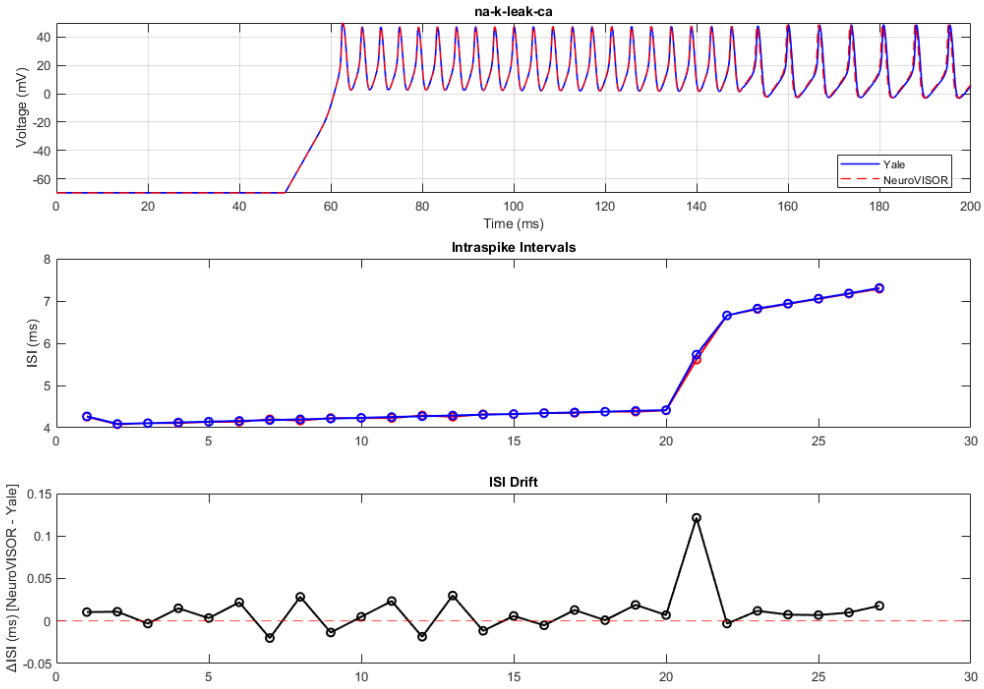

Figure 6: Three-panel validation for Na+K+Leak+CaH. Increased firing frequency relative to baseline is consistent across both platforms. Stimulus applied 50–150 ms.

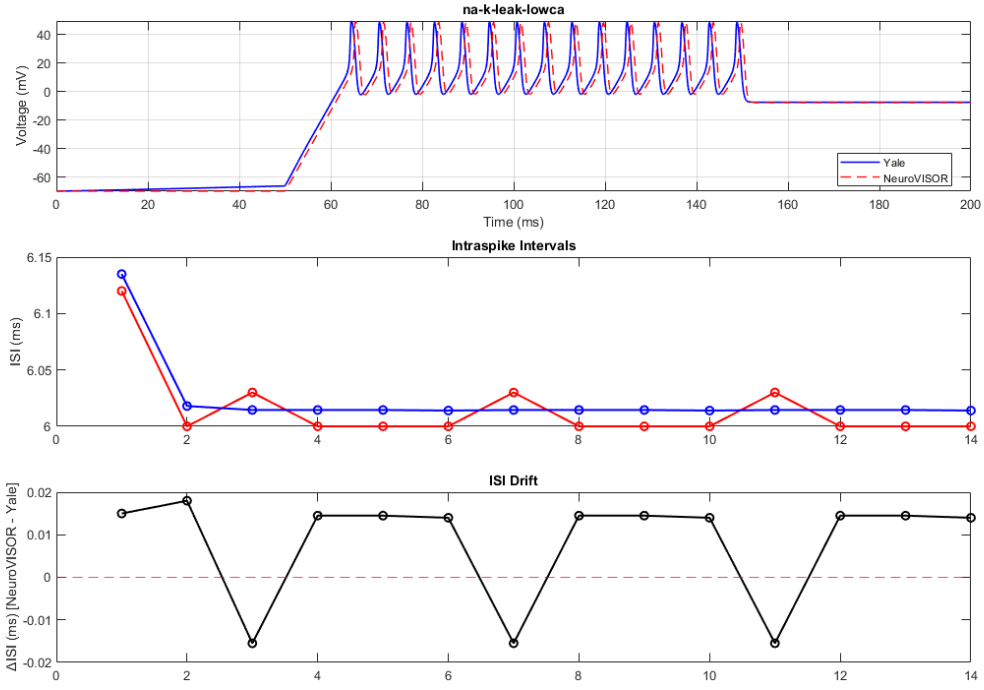

Figure 7: Three-panel validation for Na+K+Leak+T (T-type calcium). Stimulus applied 50–150 ms.

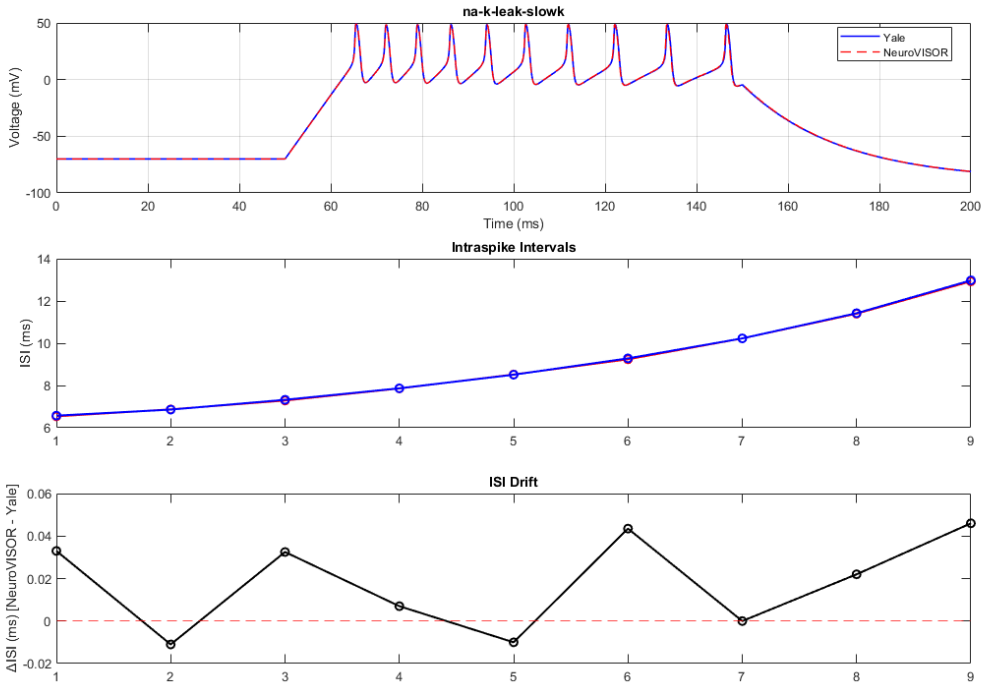

Figure 8: Three-panel validation for Na+K+Leak+M (M-current). Progressive increase in ISI across the stimulus window is visible in the middle panel, consistent with spike-frequency adaptation. Stimulus applied 50–150 ms.

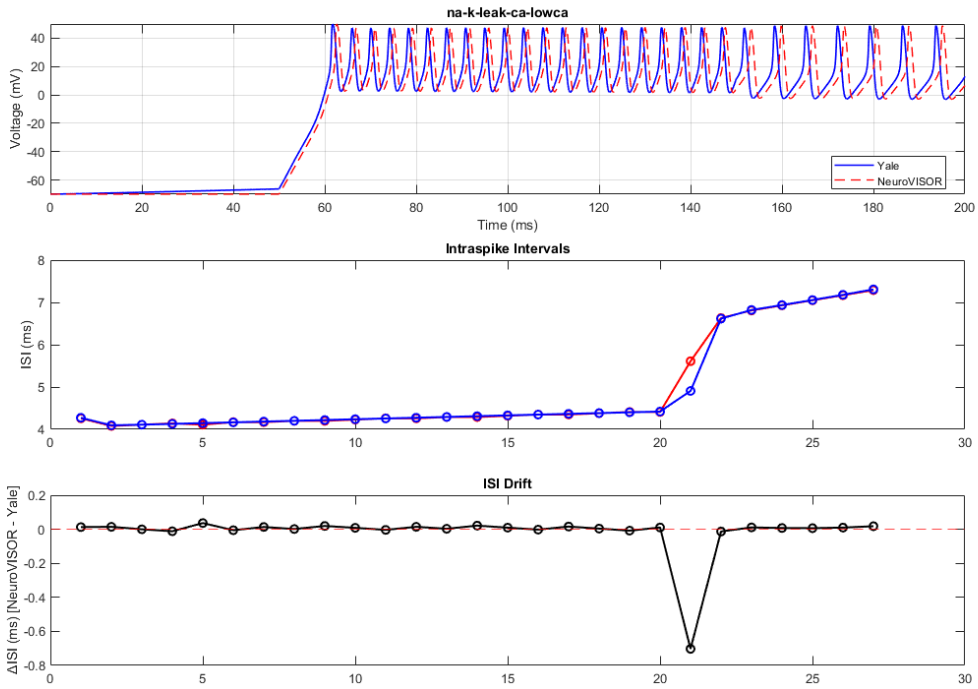

Figure 9: Three-panel validation for Na+K+Leak+CaH+T. Stimulus applied 50–150 ms.

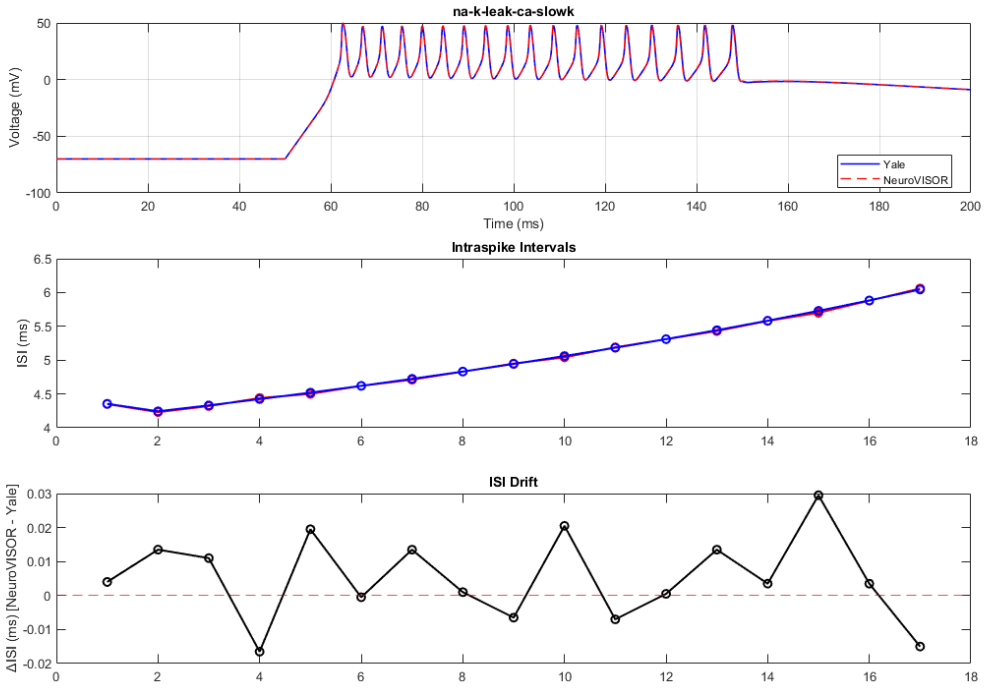

Figure 10: Three-panel validation for Na+K+Leak+CaH+M. The competing excitatory (CaH) and adaptive (M-current) effects produce an intermediate firing frequency consistent across both platforms. Stimulus applied 50–150 ms.

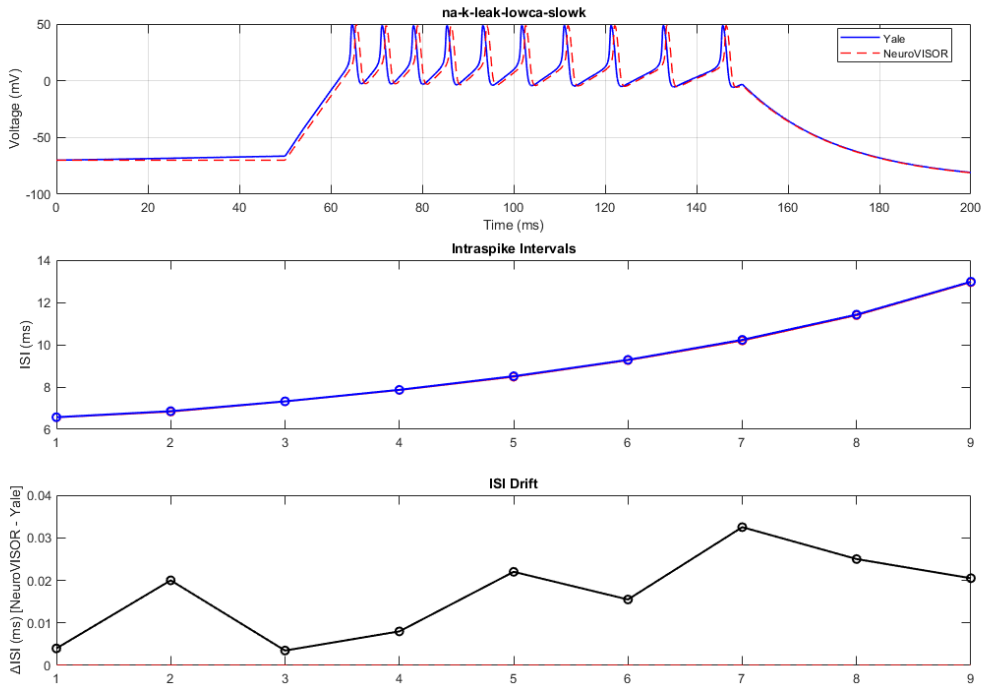

Figure 11: Three-panel validation for Na+K+Leak+T+M. Stimulus applied 50–150 ms.

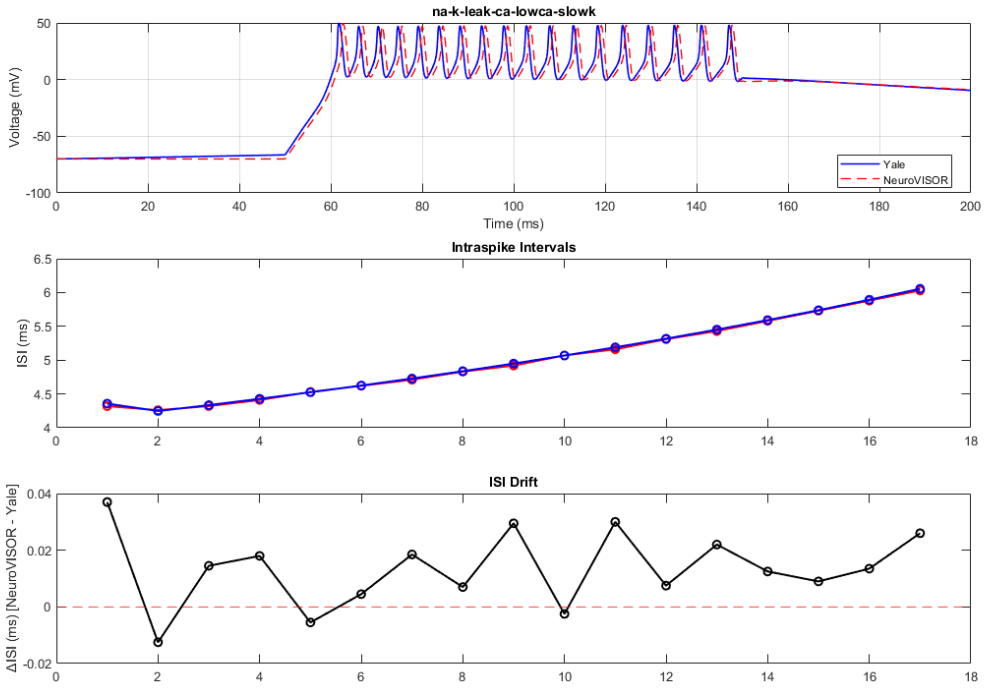

Figure 12: Three-panel validation for the full six-channel configuration (Na+K+Leak+CaH+T+M). Strong ISI agreement is maintained across all spike intervals despite the complexity of six simultaneous channel types. Stimulus applied 50–150 ms.

Across all eight configurations, NeuroVISOR achieved ISI agreement within 0.02 ms and firing frequency matching within 1 Hz, with 100% spike count agreement over the full 200 ms simulation window. The baseline Na+K+Leak configuration achieved the closest agreement (ΔISI $= -0.006$ ms, $\Delta f = 0.18$ Hz), while the full six-channel configuration maintained strong agreement (ΔISI $= -0.014$ ms, $\Delta f = +0.56$ Hz) despite the additional complexity of three simultaneous supplementary channels. Table 2 summarizes ISI and firing frequency metrics across all eight configurations.

Configuration	Yale ISI (ms)	NV ISI (ms)	ΔISI (ms)	Yale Freq (Hz)	NV Freq (Hz)	ΔFreq (Hz)
Na+K+Leak	6.024	6.017	−0.006	166.0	166.2	+0.18
Na+K+Leak+CaH	4.918	4.907	−0.011	212.5	212.9	+0.43
Na+K+Leak+T	6.024	6.015	−0.009	166.0	166.3	+0.24
Na+K+Leak+M	9.003	8.987	−0.017	116.6	116.8	+0.21
Na+K+Leak+CaH+T	4.888	4.907	+0.019	213.5	212.9	−0.60
Na+K+Leak+CaH+M	5.014	5.008	−0.005	201.9	202.1	+0.24
Na+K+Leak+T+M	9.007	8.990	−0.017	116.5	116.8	+0.21
Full (6 channels)	5.021	5.007	−0.014	201.7	202.2	+0.56

Table 2: Validation of NeuroVISOR against Yale NEURON across eight ion channel configurations. ISI values represent mean interspike interval over the full 200 ms simulation. Frequency computed as $f_i = 1000 \,/\, \text{ISI}_i$ (Hz), averaged across all intervals.

5.2 Multi-Channel Implementation

The extended NeuroVISOR framework successfully supports six physiologically relevant ion channels – voltage-dependent Na^+ (I_{Na}), "delayed-rectifier" K^+ (I_K), leak (I_{Leak}), high-threshold Ca^{2+} (I_{CaH}), low-threshold Ca^{2+} (I_T), and slow non-inactivating K^+ (I_M) – with real-time toggling via an interactive VR panel. Figure 13 shows the ion channel panel as it appears in the NeuroVISOR environment, where individual channels can be activated or deactivated during a running simulation.

All six channels were implemented and verified to produce physiologically consistent action potential

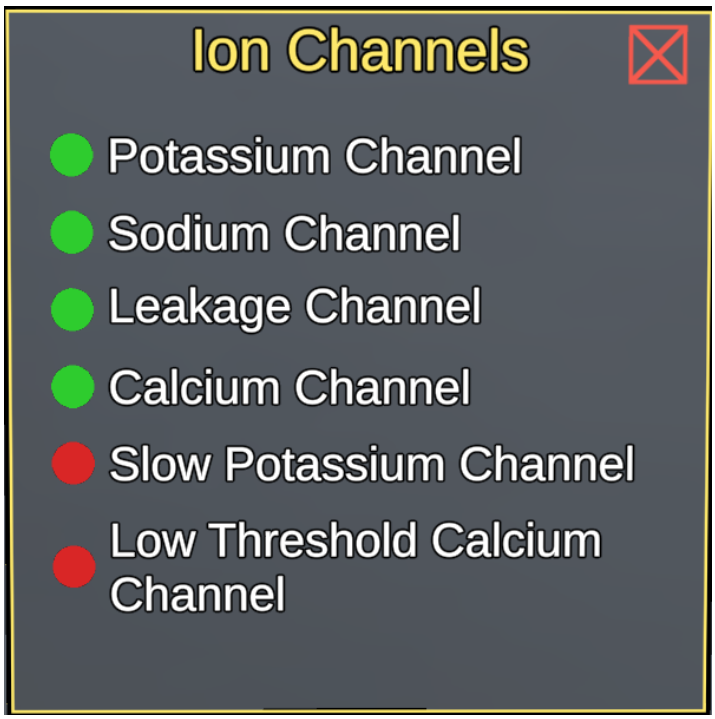

Figure 13: The NeuroVISOR ion channel panel displaying six configurable ion channels. Green indicators denote active channels; red indicators denote inactive channels. Channels can be toggled in real time during simulation without restarting the solver.

dynamics across a 200 ms simulation window, with stimulus injection active from 50 ms to 150 ms. The modular architecture allows any combination of the six channels to be enabled simultaneously, with the solver dynamically updating the active channel set and associated gating variables at each timestep.

6 Discussion

6.1 Findings

The extended NeuroVISOR framework successfully reproduces the distinct electrophysiological signatures of six ion channel types, each contributing uniquely to the neuronal firing patterns. We discuss the key findings below.

6.1.1 Spike-Frequency Adaptation via M-Current

The addition of the M-current (I_{M}) to the baseline Na+K+Leak configuration produced a pronounced reduction in mean firing frequency from 166.0 Hz to 116.6 Hz, accompanied by progressive lengthening of interspike intervals across the spike train. The first interspike interval of the Na+K+Leak+M configuration was 6.573 ms, increasing to 12.976 ms by the final spike, yielding an adaptation index of 1.97 over the 100 ms stimulation window.

This near-doubling of the interspike interval is a hallmark of spike-frequency adaptation and is consistent with the known physiology of M-current. As the membrane depolarizes during sustained firing, I_{M} activates progressively, providing an accumulating hyperpolarizing influence that lengthens each successive interspike interval. This behavior is characteristic of regular-spiking cortical neurons [P+08] and was directly observable in the NeuroVISOR VR environment through the ion channel panel, where deactivating I_{M} mid-simulation immediately restored regular firing at 166 Hz.

6.1.2 Pacemaker Activity via Calcium Channels

High-threshold calcium channels (I_{CaH}) increased mean firing frequency from 166.0 Hz to 212.5 Hz and enabled sustained action potential generation beyond the stimulus window. Following stimulus termination at 150 ms, the Na+K+Leak+CaH configuration produced 7 additional action potentials between 150 ms and 200 ms, while the Na+K+Leak baseline ceased firing immediately upon stimulus removal. This post-stimulus firing constitutes pacemaker activity, in which the persistent inward calcium current depolarizes the membrane autonomously without external input. This property is fundamental to rhythmically active neurons such as cardiac pacemaker cells, and represents a class of neuronal behavior entirely inaccessible in the original two-channel NeuroVISOR implementation.

6.1.3 Non-Additive Channel Interactions

When both I_{CaH} and I_{M} were active simultaneously, the resulting firing frequency of 201.9 Hz fell between the individual effects of each channel alone (212.5 Hz for +CaH, 116.6 Hz for +M). This non-additive interaction reflects the opposing nature of the two currents: calcium provides persistent inward depolarization while M-current supplies progressive outward hyperpolarization. Their co-activation establishes a dynamic equilibrium that moderates both the firing rate and the degree of adaptation, highlighting that multi-channel configurations produce emergent behaviors that cannot be predicted by examining channels in isolation. The NeuroVISOR toggle panel makes this interaction directly explorable, allowing users to activate and deactivate channels in real time to observe these competing influences on membrane dynamics.

6.2 Future Work

6.2.1 Predefined Simulation Experiments

A natural extension of the current framework is the development of a predefined simulation library, enabling users to launch pre-configured experiments from a single button press within the VR environment. In this paradigm, channel configurations, stimulation protocols, and recording parameters would be specified in advance, and upon activation, the simulation would initialize, run, and present results without requiring manual channel toggling or electrode placement. This design shifts the user's role from operator to observer, freeing cognitive resources for biological interpretation rather than parameter management.

6.2.2 Extended Ion Channel Library

The modular `IonChannel` architecture introduced in this work provides a foundation for expanding the channel library beyond the current six implementations. Adding more ion channels from equations that have been experimentally acquired is a natural and significant next step. The modular architecture removes the barrier to extension – any developer or user familiar with the codebase can introduce a new channel type by adding a single entry to `IonChannelModels.cs`.

6.2.3 User-Adjustable Channel Parameters

Currently, channel conductances and reversal potentials are fixed at values derived from Pospischil [P+08]. A natural extension would expose these parameters to the user through the ion channel panel, enabling real-time manipulation of maximal conductances ($\bar{g}$) and reversal potentials (E_j) via sliders or direct numerical input. This would allow users to explore how parameter variation affects firing behavior.

6.2.4 Simulation Recording and Playback

The current implementation exports voltage traces as CSV files for post-hoc analysis in external tools such as MATLAB and Python. A dedicated recording and playback system built into NeuroVISOR would enable users to capture full simulation states – including voltage trajectories, gating variable dynamics, and channel configurations – and replay them within the VR environment at variable speeds. Slow-motion playback of action potential propagation, for instance, would provide an intuitive visual

understanding of the temporal sequence of channel activation and inactivation that underlies the action potential waveform without the need for an external software, such as MATLAB and Python.

References

[HC97] Michael L. Hines and Nicholas T. Carnevale. The NEURON simulation environment. *Neural Computation*, 9:1179–1209, 1997.

[HC01] M L Hines and N T Carnevale. Neuron: a tool for neuroscientists. *The Neuroscientist*, 7:123 – 135, 2001.

[HH52] 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*, 117:500–544, 1952.

[P+08] Martin Pospischil et al. Minimal Hodgkin–Huxley type models for different classes of cortical and thalamic neurons. *Biological Cybernetics*, 99:427–441, 2008.

[SM19] Brette R Stimberg M, Goodman DF. Brian 2, an intuitive and efficient neural simulator. *eLife*, 8, 2019.