Xolotl: An Intuitive and Comprehensible Neuronal Simulator

Alec Hoyland †, **Srinivas Gorur-Shandilya** †, **and Eve Marder** *

Marder Lab, Brandeis University, Biology Department and Volen Center for Complex Systems, Waltham, MA, USA

Correspondence*:
Eve Marder
Volen Center for Complex Systems
Brandeis University
415 South Street
Waltham, MA 02454
USA
marder@brandeis.edu

† Co-first authors

ABSTRACT

`xolotl` is an open-source neuronal simulator written in `C++` with `MATLAB` wrappers. Complex models and networks can be designed efficiently using an intuitive language tightly coupled to the object-based architecture of the underlying `C++` code. Models can be specified by adding conductances to compartment objects. The structure is modular, serialized, and searchable, permitting high-level programmatic control over nearly all features of the models. `C++` templates are provided for developing new conductances, compartments, and integration schemata. It also includes a customizable graphical user interface (GUI, 'puppeteer') for rapid prototyping and hand-tuning conductances in real-time. The modular structure and accessibility to all parameters, variables, and dynamics of the model network in `MATLAB` facilitate rapid construction and assessment of model networks. `xolotl` is freely available at `https://github.com/marderlab/xolotl`. This tool provides straightforward implementation and fast simulation of neuronal models while permitting full control over every aspect of the network and integration.

Keywords: simulator, MATLAB, C++, xolotl, conductance-based, computational, keyword, keyword

1 INTRODUCTION

`xolotl` (`https://github.com/sg-s/xolotl`) is a fast single-compartment and multi-compartment simulator in `C++` with `MATLAB` wrappers (`https://www.mathworks.com/products/matlab.html`). Written with an emphasis on flexibility and speed, `xolotl` can simulate single-compartment conductance-based models, networks of these, and detailed multi-compartment models. `xolotl` exploits a novel automatic type system, `cpplab`, which binds `MATLAB` code to `C++` header files, creating objects and classes *ad libitum* in `MATLAB` which reflect the underlying object-oriented code. `xolotl` implements `cpplab` to represent the nested structure of conductance-based models, and exploits the computational efficiency of the low-level programming language to quickly integrate models. For this reason, models can be implemented entirely in `MATLAB` with few lines of code.

Models are specified in `MATLAB` by a `xolotl` object which contains compartment objects which themselves contain conductances. Synapses belong to the `xolotl` object and connect compartments together. The high-level specification supports arbitrarily large network and multi-compartment morphologies.

`xolotl` provides parameter optimization capabilities through the algorithm-agnostic `procrustes` toolbox. Any network parameters accessible through the `xolotl` structure can be optimized using arbitrary algorithms and objective functions on multi-core computers and high-performance computing clusters.

The software has been implemented in `MATLAB` due to its ease-of-use and popularity among neuroscientists. `cpplab` provides a powerful backend for specifying and integrating models without relying on the significantly slower and limiting `MATLAB codegen`. Minimal experience with `MATLAB` is required to use `xolotl`, and all equations and integration methods are provided transparently to the end user. No string parsing of equations is required.

`xolotl` comes packaged with visualization functions, a graphical user interface (GUI) for real-time manipulation of model parameters. Plotting of voltage, intracellular calcium, conductance gating functions, and time constants is provided by built-in `xolotl` methods. The `puppeteer` toolbox permits real-time tuning of any network parameters using numerical sliders in a graphical interface which displays the resultant membrane potential and intracellular calcium traces. The ease-of-use of these tools lends them to pedagogical applications and rapid exploration of toy models. This tool aims to simplify the investigation of dynamics of complex neural network models, facilitate collaborative modeling, and complement other tools being developed in the neuroinformatics community.

2 ARTICLE TYPES

For requirements for a specific article type please refer to the Article Types on any Frontiers journal page. Please also refer to Author Guidelines for further information on how to organize your manuscript in the required sections or their equivalents for your field

3 WORKED EXAMPLES

Using `xolotl` in `MATLAB`, users create a `xolotl` object and populate it with compartments, synapses, and controllers. Each field is a `cpplab` object constructed by a function call to `add`. The model is integrated with the `integrate` function where the membrane potential, intracellular calcium concentration, controller states, intrinsic currents, and synaptic currents can be outputs.

`xolotl` comes packaged with a library of pre-existing conductance and synapse objects which greatly simplify the task of constructing model neurons. These objects can be referenced by name and added directly to a compartment. Novel conductance dynamics can be easily written by modifying a template header file contained in the `xolotl` distribution.

3.1 Simulating a Hodgkin-Huxley Model

The seminal Hodgkin-Huxley model contains a fast inactivating sodium conductance which promotes spiking, a non-inactivating potassium delayed rectifer, and a passive leak current (1A). A compartment named `HH` with compartment properties of membrane capacitance $C_m = 10\ \mu\mathrm{F/mm}^2$ and surface area $A = 0.1\ \mathrm{mm}^2$ can be specified by 1B. Compartment, conductance, synapse, and controller properties can be specified during the call to the `add` function, or after construction, using dot-notation in `MATLAB` (e.g.

`x.HH.Cm`). 1C shows the `MATLAB` command prompt after invoking the `xolotl` object `x`, displaying the hierarchical structure inherent in conductance-based treatments of neurodynamics.

This model was constructed using conductances from Liu *et al* 1998. In the absence of applied positive current, the model is quiescent and tonically spikes under 0.2 nA of applied current (1D). The `integrate` function takes the applied current as an argument (e.g. `x.integrate(AppliedCurrent)`), so that the `xolotl` object is agnostic to integration-specific perturbations. The `plot` function generates voltage and intracellular calcium traces, where the voltage trace is colored by the dominant current. If the membrane potential is increasing, the strongest instantaneous inward current colors the trace. Conversely, if the membrane potential is decreasing, the strongest outward current colors the trace instead. 1F-I display the results of the `show` function. Activation and inactivation steady-states and the voltage-dependent time constants of these gating variables describe the conductance dynamics in absence of other channel types.

3.2 Performing a Voltage Clamp Experiment *in-silico*

`xolotl` can recapitulate the results of voltage clamp experiments. 2 displays the procedure to clamp the membrane potential of a cell with a delayed rectifier potassium conductance. The second argument of the `integrate` function determines the clamped voltage (e.g. `x.integrate([], VoltageClamp)`). Isolated currents under voltage clamp approach the steady-state (2D-E) so that a current-voltage relation at steady-state can be extracted (2F). The derivative of the IV curve is the steady-state conductance (2G). Fitting a sigmoid to various powers yields a model for the current dynamics (2H-I). These figures describe graphically the theoretical underpinnings of current analysis through voltage clamp and can serve as an effective pedagogical tool for computational and quantitative neuroscience.

3.3 Simulating Network Models

Network models in `xolotl` consist of compartment objects connected by synapses. Synapses are stored in a vector array as a field of the `xolotl` object in `MATLAB`. Presynaptic and postsynaptic labels indicate the connectivity of the synapse. Figure 3 implements a model of the triphasic pyloric rhythm in the stomatogastric ganglion of crustaceans. The pyloric model contains three compartments and seven synapses (3A). This structure is reciprocated in the hierarchy of the `xolotl` object, where conductances are contained within compartments (3B). Synapses are upper-level properties of the network which point between two compartments (3C). This exploits vectorized operations in `MATLAB` and does not require each synapse to possess a unique name. The `plot` function generates multiple subplots when called for a network with multiple compartments (3D-F).

3.4 Simulating Integral Control

`xolotl` can implement homeostatic tuning rules as integral control. Figure 4 depicts generation of a bursting neuron model from quiescent conditions. Calcium sensors supervene on maximal conductance density (4). In `xolotl`, integral controllers are properties of conductances (4B-C). They modify properties of the conductance in response to an error signal. In a demonstration adapted from O'Leary *et al.* 2013, integral control changes maximal conductances to bring a neuron from quiescence into a bursting regime. Maximal conductances increase from random initial conditions to a set which elicits the desired network output by minimizing the error signal (4D-F).

4 TECHNICAL DETAILS

4.1 Modeling

Models are specified by adding compartments and synapses to the `xolotl` object. Conductances are added to compartments and controllers can be added to conductances. This modular structure recapitulates the biophysics of the Hodgkin-Huxley formalism and obviates the need to explicitly write out equations, which in `xolotl` are contained within the conductance header files.

`xolotl` relies on `cpplab` constructions, which allow the user to exploit the efficiency of low-level `C++` code. `MATLAB` treats `cpplab` objects as standard variables allowing for symbolic manipulation using only the high-level programming language and graphical interfaces. `xolotl` is fast specifically because all time-intensive code is written in native `C++`. While automated `C++` transpiling from `MATLAB` using the proprietary `coder` can drastically improve performance over loops through strong typing and memory pre-allocation, supervenience of `MATLAB` over `C++` prevents efficient use of low-level features, such as passing by reference and object-oriented programming.

Native `C++` provides drastic speed improvements beyond the benefits of translating `MATLAB` features into low-level code. For this reason, `cpplab` has been designed to provide an interface for constructing, transpiling, and compiling `C++` code to be called from within `MATLAB`. `xolotl` simulations can be run entirely from `C++` executables. To facilitate use in `MATLAB`, `xolotl` uses the MD5 algorithm to automatically hash the network and compile a new binary and `MEX` bridge file only if needed. `MATLAB` provides a high level programmatic and graphical interface for implementing, manipulating, and visualizing models without sacrificing the enhancements of native `C++` code.

4.1.1 Using the `cpplab` Framework

The `add` function will construct a `cpplab` object and affix it to as a field in the `xolotl` structure. All compartments, conductances, synapses, and controllers are `cpplab`. Compartments add to the `xolotl` object and conductances add to compartments. Specific properties can be specified using key-value pair arguments (e.g. 1A).

`cpplab` comes with several features which simplify the handling of high-dimensional models. The `find` function acquires a cell array of all properties of the network which satisfy a search condition.

4.1.2 Compartments and Synapses

A model neuron consists of one or more compartments, each representing a section of membrane with capacitance and surface area. Isopotential models require one compartment, whereas models with multiple neurons, units, or non-trivial morphology require multiple compartments. All specifiable properties of compartments are shown in Supplementary Table 1.

`xolotl` provides some features for generating complex models. Synapses can be added with the `connect` function. At minimum synapses possess identifiers to presynaptic and postsynaptic compartments and default to electrical synapses. All specifiable properties of synapses are shown in Supplementary Table 2. To create axons or transport chains, the `slice` function splits a compartment into n discrete segments and adds these compartments to the network connected by electrical synapses.

4.1.3 Conductances and Controllers

All conductances contain fields for maximal conductance and reversal potential. Conductances with activation and inactivation variables include them as m and h respectively. Gating functions and their respective time constants are contained within the conductance header file. `xolotl` comes packaged with conductances from several dozen papers (Supplementary Table 3).

4.1.4 Creating Custom `cpplab` Objects

`xolotl` contains template header files for producing custom conductances. The template contains instructions on how to design novel conductances with arbitrary specifications.

4.2 Simulation

Models are simulated in `xolotl` with the `integrate` function which outputs as time series the membrane potentials, intracellular calcium concentrations, controller states, intrinsic currents, and synaptic currents. The `integrate` function also accepts an argument which specifies injected current or clamped voltage.

`xolotl` uses the exponential Euler method (Dayan & Abbott 2001) for single compartment models, forward Euler for gating variables, and a Crank-Nicholson regime for electrically-coupled compartments. The simulation time-resolution can be specified to target arbitrary precision, and an output time step can be selected to support automatic down-sampling for memory considerations.

Simulations can be run in ”open-loop” mode where each simulation begins by resetting all dynamical variables to their initial conditions at instantiation, or ”closed-loop” mode which begins simulation with the current network state.

4.3 Using the `puppeteer` graphical interface

`xolotl` comes packaged with a graphical user interface for visualizing parameter changes in real-time. The `manipulate` function begins the

CONFLICT OF INTEREST STATEMENT

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

AUTHOR CONTRIBUTIONS

AH and SG-S wrote the manuscript and the code. EM provided funding and moral support.

FUNDING

Details of all funding sources should be provided, including grant numbers if applicable. Please ensure to add all necessary funding information, as after publication this is no longer possible.

AH received funding from National Institute on Drug Abuse (NIDA) through the undergraduate training grant in computational neuroscience (1R90DA033463-01).

ACKNOWLEDGMENTS

The authors would like to thank Mara CP Rue and Hillary Rodgers for beta-testing the `xolotl` software.

SUPPLEMENTAL DATA

Table including all conductances packaged with `xolotl` should be put in the supplementary material.

DATA AVAILABILITY STATEMENT

The code to generate all figures can be found in the `xolotl` repository (`https://github.com/marderlab/xolotl`).

REFERENCES

[Dataset] LastName1, A., LastName2, A., and LastName3, A. (2011). Data title. doi:10.000/55555

LastName1, A., LastName2, A., and LastName3, A. (2013). Article title. *Frontiers in Neuroscience* 30, 10127–10134. doi:10.3389/fnins.2013.12345

Name, A. (1993). *The title of the work* (The city: The name of the publisher)

Name, C., Surname, D., and LastName, F. (1996). The title of the work. In *The title of the conference proceedings*, eds. E. Name1 and E. Name2 (The name of the publisher), 41–50

Surname, B. (2002). The title of the work. In *The title of the book*, ed. E. Name (The city: The name of the publisher). 201–213

Surname1, H. (2010). *The title of the work* (Patent country: Patent number)

FIGURE CAPTIONS

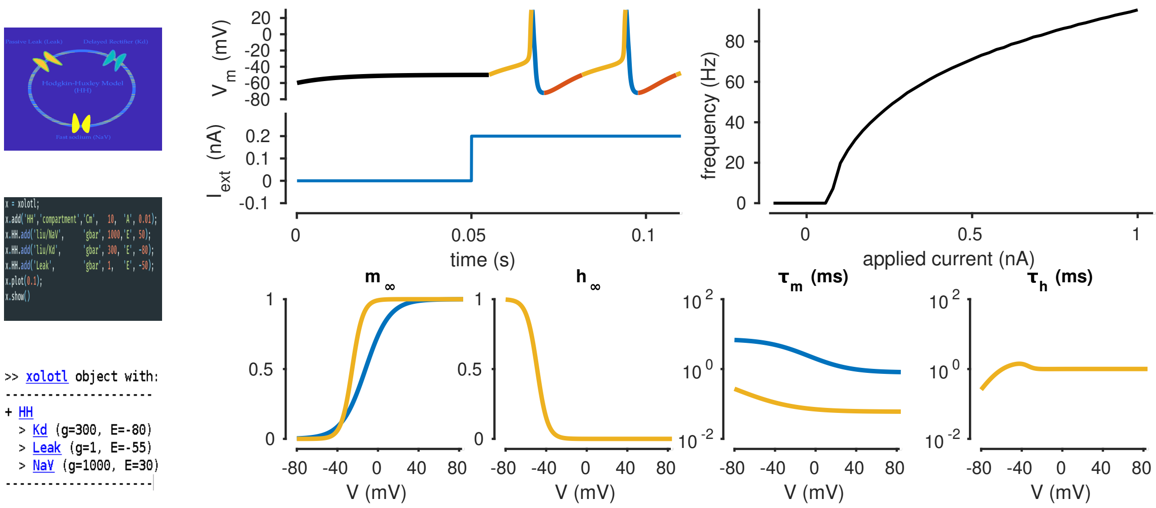

Figure 1. `xolotl` can quickly set up and simulate conductance-based models. (A) Cartoon of a Hodgkin-Huxley single-compartment neuron model with fast sodium, delayed rectifier, and leak currents. (B) Code snippet in `MATLAB` used to implement D, F-I. (C) `xolotl` schematic displayed in the `MATLAB` command prompt. (D) Simulated voltage trace with 0.1 nA applied current. Colors indicate the dominant current (gold is fast sodium, blue is delayed rectifier, red is leak). (E) Frequency-input relation displaying firing rate as a function of applied current. (F-G) Steady-state gating functions for activation (m) and inactivation (h) gating variables. Variables not plotted are unity for all voltage. (H-I) Voltage-dependence of time constants for activation (m) and inactivation (h) gating variables. Variables not plotted are unity for all voltage. Colors indicate conductance type (gold is fast sodium, blue is delayed rectifier, red is leak).

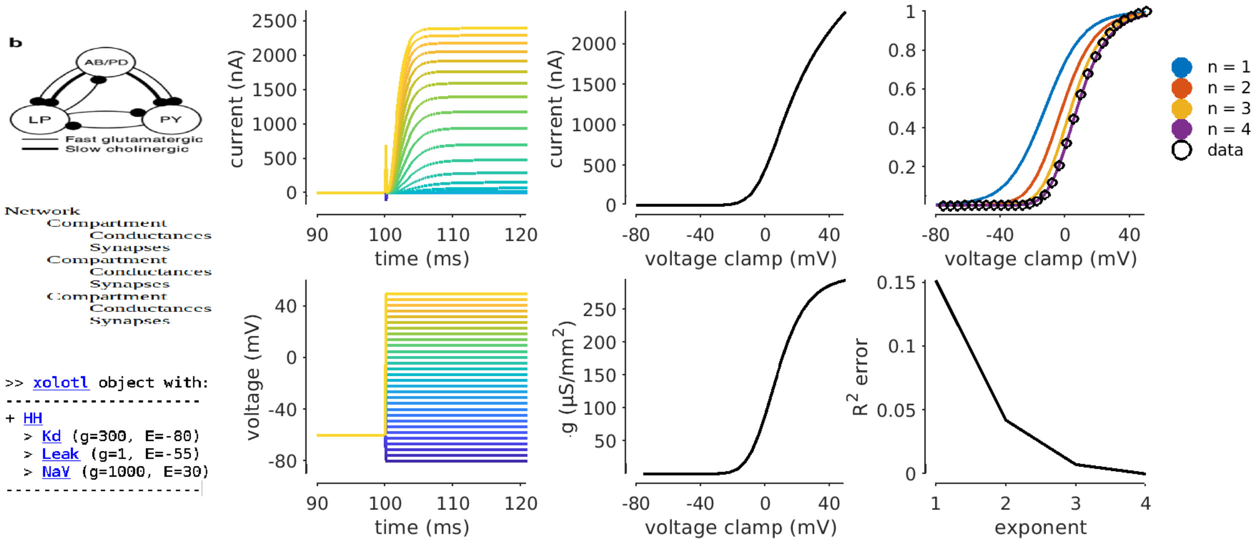

Figure 2. `xolotl` readily implements voltage clamp. (A) Cartoon of a cell with potassium conductance with experimentally-fixed voltage. (B) Structure of `xolotl` object in A. (C) Code snippet depicting integration under voltage clamp. (D-E) Current response to steps in voltage from a holding potential of $V_m = -60$ mV. (F) Current-voltage relation of the steady-state current ($t = 400$ ms) indicating a reversal potential of $E = -80$ mV and no inactivation. (G) Conductance-voltage relation at steady-state takes the form of a sigmoid. (H) Sigmoids m fit to the model as m^n data indicating that $n = 4$ is the best fit. (I) R^2 correlation of the sigmoid fits at various powers where $n = 4$ is an exact fit.

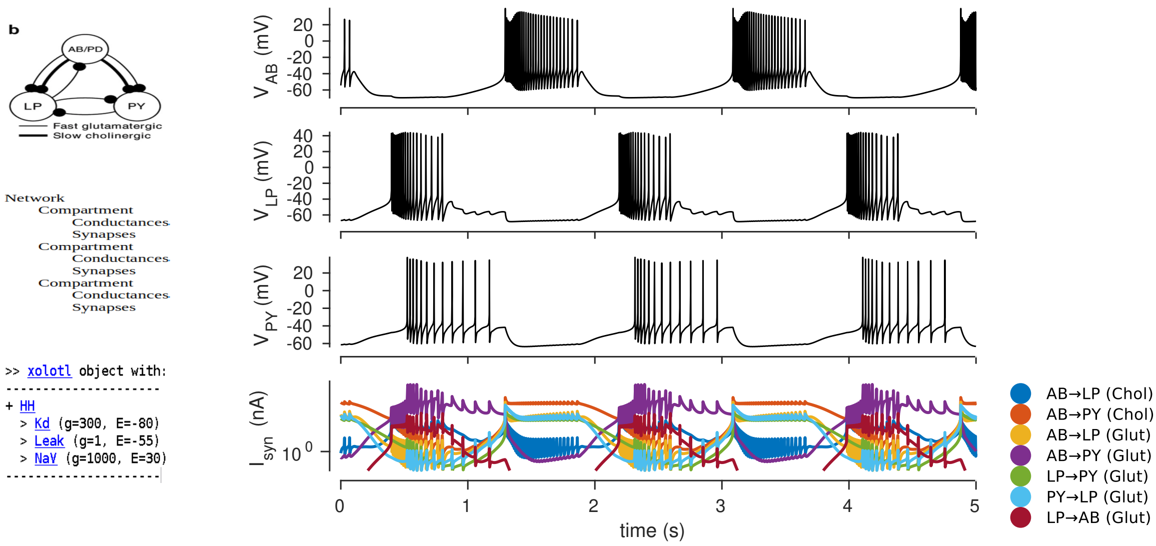

Figure 3. `xolotl` readily implements conductance-based network models. (A) Diagram of a network model of the pyloric rhythm in the crustacean stomatogastric ganglion (Prinz *et al.* 2004). (B) Hierarchical structure of a neuronal network considers compartments as components of the network and conductances and synapses as components of compartments. (C) *xolotl* implements conductances as fields of compartments and synapses as connections between compartments. (D-F) Simulated voltage trace of a model network for the three compartments. (G) Time series of synaptic currents in the simulated network with model cholinergic (chol) and glutamatergic (glut) synapses are outputs of the integration.

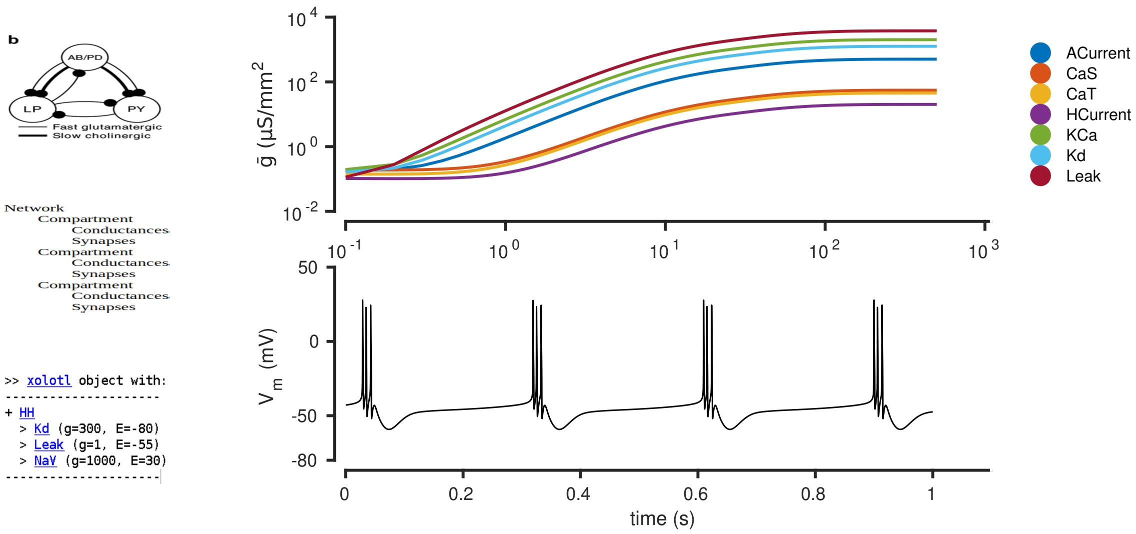

Figure 4. `xolotl` can implement homeostatic tuning rules as integral control. (A) Cartoon of a model neuron with integral control. (B) Hierarchical structure of a neuronal network considers controllers as components of compartments which act on conductances. (C) `xolotl` implements controllers XYZ. (D) Calcium sensors change maximal conductances to move a neuron from quiescence to a bursting state. (E) Voltage trace of the controlled neuron shows regular bursting activity.