

Xolotl: An Intuitive and Approachable Neuron & Network Simulator in MATLAB

Alec Hoyland^{1†}, Srinivas Gorur-Shandilya^{1†}, and Eve Marder^{1*}

¹ *Volen Center and Biology Department
Brandeis University
Waltham, MA 02454
USA*

[†] *These authors have equally contributed to this article.*

Correspondence*:
Srinivas Gorur-Shandilya
Marder Lab
srinivas@brandeis.edu

An Intuitive Neuronal Simulator

2 ABSTRACT

`xolotl` is a free and open-source neuronal simulator written in C++ with MATLAB wrappers. Biophysically-detailed models of 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) 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++, conductance-based, neuron, network, pedagogy

1 INTRODUCTION

Conductance-based models are the simplest biophysical representation of an electrically-excitable cell (Hodgkin and Huxley 1952a). Modeling studies now contribute significantly to mainstream research in small-circuit networks (E. Marder and Abbott 1995; Astrid A. Prinz 2006; Astrid A. Prinz 2010). Additionally, conductance-based models provide an approachable formalism based in electrophysiology for understanding salient principles of neuroscience. To this end, we have developed `xolotl` (<https://github.com/marderlab/xolotl>), a fast single-compartment and multi-compartment simulator in C++ with MATLAB wrappers. Written with an emphasis on ease-of-use, `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 a few lines of code.

Models are specified in MATLAB by a nested structure. The `xolotl` object contains compartments 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.

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`. While automated C++ transpiling from MATLAB using the proprietary `codegen` 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. 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 (Sherfey et al. 2018; Stimberg, D F M Goodman, et al. 2014; Stimberg, D F Goodman, et al. 2013)

`xolotl` comes packaged with visualization functions and 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 GUI 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 DESIGN GOALS

`xolotl` is designed to be easy-to-use without sacrificing speed.

The software has been designed in MATLAB due to its popularity among neuroscientists for pedagogy and research. `xolotl` capitalizes on MATLAB's straightforward structure array syntax to permit rapid prototyping and experimentation, especially for small neuronal networks of complex models. Parameters of conductances, neuronal compartments, and simulations may all be edited in the structure before any calls to integration functions. The underlying code is written in C++ for speed and memory optimization, and while models can indeed be integrated using the compiled binary, symbolic manipulations can be readily performed in MATLAB without ever touching the foundational code.

2.1 FEATURES

MATLAB provides a high level programmatic and graphical interface for implementing, manipulating, and visualizing models without sacrificing the enhancements of the underlying C++ code.

Modular structure. 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.

66 *Interface between C++ and MATLAB.* `xolotl` relies on `cpplab` constructions, which allow the user
 67 to exploit the efficiency of low-level C++ code. MATLAB treats `cpplab` objects as fully-typed variables
 68 allowing for symbolic manipulation using only the high-level programming language and graphical inter-
 69 faces. `xolotl` is fast because all time-intensive code is written in C++. While automated C++ transpiling
 70 from MATLAB using the proprietary `codegen` can drastically improve performance over loops through
 71 strong typing and memory pre-allocation, supervenience of MATLAB over C++ prevents efficient use of
 72 low-level features, such as passing by reference and object-oriented programming. C++ provides speed
 73 improvements beyond the benefits of translating MATLAB features into low-level code. For this reason,
 74 `cpplab` has been designed to provide an interface for constructing, transpiling, and compiling C++ code
 75 to be called from within MATLAB. `xolotl` simulations are run entirely from C++ executables.

76 *Automatic and efficient compiling.* `xolotl` automatically handles transpiling and compiling MATLAB
 77 code into C++. The MD5 algorithm hashes the network to compile a new binary and MEX bridge file only
 78 if needed and to confirm that the correct binary fetched during execution.

2.2 SYNTAX

79 MATLAB can easily control the `cpplab` objects using the standard, flexible data structure notation popular
 80 in high-level scripting languages.

81 *Adding features.* The `add` function creates a compartment, conductance, or controller and affixes it as
 82 a field in the `xolotl` network structure. This function generates a MATLAB `struct` that faithfully
 83 represents the underlying C++ code. Compartments add to the `xolotl` object and conductances add to
 84 compartments. Specific properties can be specified using key-value pair arguments (e.g. Figure 1A).

85 *Finding features.* `cpplab` comes with several features which simplify the handling of complex, nested
 86 models. The `find` function acquires a cell array of all properties of the network which satisfy a search
 87 condition. For example, one can find all paths to maximal conductances within the 'HH' compartment
 88 by:

```
89         x.find('HH*gbar');
```

90 To extract a vector of the maximal conductances:

```
91         gbars = x.get('HH*gbar');
```

92 To set the maximal conductances all at once:

```
93         x.set('HH*gbar', gbars)
```

94 *Compartments.* A model neuron consists of one or more compartments, each representing a section of
 95 membrane with capacitance and surface area. Isopotential models require one compartment, whereas
 96 models with multiple neurons, units, or non-trivial morphology require multiple compartments.

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

102 *Conductances and controllers.* All conductances contain fields for maximal conductance and reversal
 103 potential. Conductances with activation and inactivation variables include them as m and h respectively.

104 Gating functions and their respective time constants are contained within the conductance header file.
 105 `xolotl` comes packaged with conductances from several dozen papers (Dethier et al. 2015)

106 *Creating custom `cpplab` objects.* `xolotl` contains template header files for producing custom con-
 107 ductances. The template contains instructions on how to design novel conductances with arbitrary
 108 specifications.

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

113 *Numerical integration.* `xolotl` uses the exponential Euler method for single compartment models, for-
 114 ward Euler for gating variables, and a Crank-Nicholson regime for electrically-coupled compartments
 115 (Butcher 2016; Dayan and Abbott 2001; Oh and French 2006) These defaults provide a mix of speed,
 116 accuracy, and stability, and are built into the `cpplab` header files. Custom `cpplab` header files can
 117 be customized with any iterative integration method. The simulation time-resolution can be specified to
 118 target arbitrary precision, and an output time step can be selected to support automatic down-sampling for
 119 memory considerations.

120 *‘Closed-loop’ vs. ‘open-loop.’* Simulations can be run in ‘closed-loop’ mode where each simulation be-
 121 gins by resetting all dynamical variables to their initial conditions at instantiation, or ‘open-loop’ mode
 122 which begins simulation with the current network state.

123 *Using the graphical interface to manipulate parameters.* `xolotl` comes packaged with a graphical user
 124 interface for visualizing parameter changes in real-time. The `manipulate` function opens the GUI,
 125 which displays a figure plotting the membrane potential and intracellular calcium concentration of all
 126 compartments as time series, and a dialog box with customizable sliders for all parameters of the model,
 127 much like the `Manipulate` function in Wolfram Mathematica. Moving the sliders integrates the
 128 model in ‘open-loop’ mode with the new parameters. The parameters available in the sliders can be cus-
 129 tomized by passing a cell array to `manipulate`. For example, to only see sliders for maximal conductances
 130 of the HH compartment, call `x.manipulate('HH*gbar')`. Closing the GUI saves the network state
 131 of the model to the `xolotl` object. This is particularly helpful for rapid prototyping of models.

132 *Optimizing parameters.* `xolotl` can use the Global Optimization toolbox for MATLAB to optimize any
 133 accessible `xolotl` parameters. The toolbox is algorithm-agnostic and accepts any function in MATLAB
 134 with a scalar first output as the objective function. Simulations run on multi-core processors or high-
 135 performance computing clusters using the Parallel Computing toolbox.

2.3 LIMITATIONS

136 The focus on ease-of-use and speed means some features were intentionally neglected in the streamlining
 137 process.

138 *Reliance on compiled C++ code.* While MATLAB comes with robust features for compiling C and C++
 139 code, `xolotl` cannot run without C++ compilation. For users, this necessitates the additional step of
 140 setting up the `mex` compiler which can be problematical, especially for nonstandard (e.g. Arch-based
 141 Linux). Secondly, compilation adds a small amount to total processing time. Longer simulations (> 1000
 142 time-steps) minimizes this effect. Adding new conductances also requires writing some C++ code. For
 143 model conductances in the Hodgkin-Huxley formalism (Dayan and Abbott 2001; Hodgkin, Huxley, and
 144 Katz 1952) adjustments consist of changing default values in a template C++ header file. Implementing a
 145 new integration scheme requires much more in-depth usage of C++.

146 *Limited to conductance-based models.* `xolotl` has been developed specifically for conductance-based
 147 models. It does not currently support rate- or current-based models.

148 *Limited numerical integration strategies.* While the exponential Euler method performs well in neuronal
 149 models (Dayan and Abbott 2001; Oh and French 2006) it may be desirable to use other methods un-
 150 der certain conditions. `xolotl` does not currently support other integration schemes for its built-in
 151 conductances, nor does the software support error-sensitive variable step-sizes ‘out-of-the-box.’

152 *Inefficient tools for handling large networks.* While `xolotl` can integrate large networks (> 1000 com-
 153 partments), `xolotl` uses string-based comprehension for labeling compartments which is suited to
 154 descriptive naming, but prohibits vector operations over compartments.

3 USAGE EXAMPLES

155 In MATLAB, users create a `xolotl` object and populate it with `cpplab`-generated objects which describe
 156 compartments, conductances, synapses, and controllers. The model is integrated with the `integrate`
 157 function where the membrane potential, intracellular calcium concentration, controller states, intrinsic
 158 currents, and synaptic currents can be outputs.

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

3.1 SIMULATING A HODGKIN-HUXLEY MODEL

163 The seminal Hodgkin-Huxley model of action potentials in the squid giant axon (Hodgkin and Huxley
 164 1952b; Hodgkin, Huxley, and Katz 1952) contains a fast inactivating sodium conductance (NaV), a non-
 165 inactivating delayed rectifier (Kd), and a passive leak current (Figure 1A). A compartment, HH, with
 166 membrane capacitance (Cm) and surface area (A) can be specified by Figure 1B. Network properties can
 167 be set during construction or afterwards using dot-notation in MATLAB (e.g. `x.HH.Cm`). Figure 1C shows
 168 the MATLAB command prompt after invoking the `xolotl` object `x`, displaying the hierarchical structure
 169 inherent in conductance-based treatments of neurodynamics.

170 This model was constructed using conductances from Liu et al. 1998 based on electrophysiological
 171 recordings from the lobster stomatogastric ganglion (Turrigiano, LeMasson, and E. Marder 1995) In
 172 the absence of applied positive current, the model is quiescent. When 0.2 nA is injected, the model
 173 tonically spikes (Figure 1D). The `integrate` function takes the applied current as an argument (e.g.
 174 `x.integrate(Iapp)`), so that the `xolotl` object is agnostic to integration-specific perturbations.
 175 The `plot` function generates voltage and intracellular calcium traces, where the voltage trace is colored
 176 by the dominant current. If the membrane potential is increasing, the strongest instantaneous inward cur-
 177 rent colors the trace. Conversely, if the membrane potential is decreasing, the strongest outward current
 178 colors the trace instead. Figure 1F-I display the results of the `show` function. Activation and inactivation
 179 steady-states and the voltage-dependent time constants of these gating variables describe the conductance
 180 dynamics in absence of other channel types.

3.2 PERFORMING A VOLTAGE CLAMP EXPERIMENT *IN-SILICO*

181 `xolotl` can recapitulate the results of voltage clamp experiments (Destexhe and Bal 2009; Swensen and
 182 E. Marder 2000, 2001; Turrigiano, LeMasson, and E. Marder 1995) Figure Figure 2 displays steps in the

183 procedure to clamp the membrane potential of a cell with delayed rectifier potassium conductance. Dur-
 184 ing an *in-vitro* experiment, confounding currents would be pharmacologically-blocked and two-electrode
 185 voltage clamp used to record tail currents at fixed membrane potential (Connor and Stevens 1971a,b)

186 A single-compartment model with a delayed-rectifier conductance is simulated at stepped membrane
 187 potentials. The model is simulated using the `integrate` function. The second argument determines the
 188 clamped voltage and the fourth output is the current trace.

```
189 [V, Ca, ~, I] = x.integrate([], clamped_voltage)
```

190 Currents under voltage clamp approach the steady-state holding current (Figure 2D-E). The current-
 191 voltage relation is the steady-state current over the clamped voltage, and the effective conductance is the
 192 derivative of that relation (Figure 2F-G). Since the effective conductance is the product of the maximal
 193 conductance and the gating variables (Dayan and Abbott 2001; Turrigiano, LeMasson, and E. Marder
 194 1995) and the tail current is monotonically-increasing with time under voltage clamp, the current can be
 195 represented as non-inactivating. Fitting a sigmoid to various powers yields a model for the current dy-
 196 namics (Figure 2H-I). These figures describe graphically the theoretical underpinnings of current analysis
 197 through voltage clamp and can serve as an effective pedagogical tool for computational and quantitative
 198 neuroscience.

3.3 SIMULATING NETWORK MODELS

199 Network models in `xolotl` consist of compartment objects connected by synapses. Synapses are stored
 200 in a vector array as a field of the `xolotl` object in MATLAB. Presynaptic and postsynaptic labels in-
 201 dicate the connectivity of the synapse. Figure 3 implements a model of the triphasic pyloric
 202 rhythm in the stomatogastric ganglion of crustaceans. The pyloric model contains three compartments
 203 and seven synapses (Figure 3A). This structure is reciprocated in the hierarchy of the `xolotl` object,
 204 where conductances are contained within compartments (Figure 3B).

205 Representing the network in `xolotl` requires constructing three compartments and eight conductances
 206 in each using the `add` function.

```
207 x.add('AB', 'compartment', 'Cm', 10, 'A', 0.628, ...)  
208 x.AB.add('prinz/NaV', 'gbar', 1000, 'E', 50)  
209 ...
```

210 Synapses are upper-level properties of the network which point between two compartments (Figure
 211 3C). This exploits vectorized operations in MATLAB and does not require each synapse to possess a
 212 unique name. The `connect` function adds synapses to the network. The first two arguments specify
 213 the presynaptic and postsynaptic compartments. The third dictates the type of synapse. All others follow
 214 the 'keyword', value paradigm and preset parameters of the synapse.

```
215 x.connect('AB', 'LP', 'Chol', 'gbar', 30)
```

3.4 SIMULATING INTEGRAL CONTROL

216 `xolotl` can implement homeostatic tuning rules as integral control. The controller computes an error
 217 signal (typically a function of intracellular calcium concentration), and adjusts the conductance or synapse
 218 it controls accordingly (O'Leary et al. 2013) In `xolotl`, integral controllers are `cpplab` objects added
 219 to the conductance or synapse they regulate.

220 In a demonstration adapted from O'Leary et al. 2013, integral control changes maximal conductances to
 221 bring a neuron from quiescence into a bursting regime. Calcium sensors supervene on maximal con-
 222 ductance density (Figure 4) to change neuronal activity. Each conductance in the `xolotl` structure

contains a calcium-sensitive controller (Figure 4B-C). Maximal conductances increase from random initial conditions to a set which elicits the desired network output by minimizing the error signal (Figure 4D-F).

3.5 USING THE GUI TO MANIPULATE PARAMETERS

The `manipulate` function opens the GUI which permits visualization of changing parameters in real-time. Moving sliders representing the values of network parameters updates a plot (Figure 5B). By default, the function opens a figure displaying the results of the `plot` function, which shows the voltage and intracellular calcium traces for each compartment (Figure 5A). `manipulate` grants slider control over all `xolotl` parameters by default, but specific ones can be selected by passing them as arguments. For example, to manipulate only the maximal conductances and visualize using the `myPlot` function

```
x.manipulate('*gbar', @myplot)
```

4 BENCHMARKS

To assess speed and accuracy, `xolotl`, `DynaSim` (Sherfey et al. 2018) and `NEURON` (Hines and Carnevale 1997) were compared in simulations over varied time resolution, simulation time, and number of compartments (Figure 6).

Single-compartment Hodgkin-Huxley-like models were generated using conductance dynamics from Liu et al. 1998 in the simulation environments. `xolotl` uses the exponential Euler method for integrating membrane potential (Dayan and Abbott 2001) `DynaSim` was implemented with a 2nd-order Runge-Kutta integration scheme as recommended for high-performance in the documentation. `NEURON` used an implicit Euler regime (Hines and Carnevale 1997)

To compare the integration methods, models were simulated for 5 s at varying time-resolution (Figure 6A). The ratio between 'simulated' time and actual runtime was defined as the speed factor. Higher values indicate faster simulations. The coincidence factor determines the correlative overlap between two spike trains (Jolivet et al. 2008) To assess accuracy over decreasing time-resolution for the three simulation environments, spike trains at each resolution were compared to a 'canonical' spike train (exponential Euler at a time-step of $dt = 0.001$ ms).

To assess the performance of the simulators in absence of set-up overhead, models were simulated with a time-resolution of 0.1 ms over increasing simulation time (Figure 6B). The speed factor was defined as the ratio between time represented in the simulation and actual runtime (simulation-time). Therefore, the speed factor represents how many times faster the simulation is than a real-time observation.

Many simulators perform well in simulations of many compartments (Brette et al. 2007; Delorme and Thorpe 2003; Sherfey et al. 2018; Vitay, Dinkelbach, and Hamker 2015) To assess how `xolotl` performs in these conditions, networks of up to 1,000 Hodgkin-Huxley cells were simulated for 5 s at a time-resolution of 0.1 ms (Figure 6C).

DESCRIBE BENCHMARK RESULTS

5 DISCUSSION

We envision that `xolotl` will be helpful in teaching students how to interpret cellular biophysics. The modular structure of `cpplab` and the graphical interface simplifies the process of manipulating and analyzing the properties of electrical excitability.

5.1 REPRODUCIBILITY

xolotl fosters reproducibility in science. While the availability of hosting sites with version control (viz. GitHub (<https://github.com>), GitLab (<https://gitlab.com/>), and Open Science Framework (<https://osf.io/>)) and the push for reproducibility in computational science (Baker 2016; Eklund, Nichols, and Knutsson 2016; Stodden et al. 2016) has resulted in the availability of source code, much of this code base is bespoke and difficult to implement (Sedano 2016; W Xu, D Xu, and Deng 2017)

To this end, xolotl provides an environment with readability and reproducibility in mind. Each network is hashed to provide a unique alphanumeric identifier. Conductance header files are easily viewed in the xolotl source files; conductances in MATLAB contain links to the full path of the generating file.

5.2 CIRCUMVENTING LANGUAGE TRADEOFFS

Executing C/C++ code in higher-level languages such as MATLAB or Python often provides speed improvements for iterative code in algorithms.

C is statically-typed, with procedural syntax that provides low-level access to memory (Kernighan and Ritchie 1978) providing significant advantages for time-intensive computations. Unfortunately, automatic code-generation is limited by the supervening language. MATLAB, for instance, cannot use pointers or pass by reference, which limits the efficiency of C code automatically generated from MATLAB. Conversely, custom C/C++ code provides significant increases in performance and memory conservation, but lacks the ease-of-use and flexibility of scripting languages.

xolotl handles this problem through symbolic manipulation of C++ objects in MATLAB. Built from the ground up in C++, xolotl maintains all the advantages of custom compiled code, but can run in MATLAB without the user having to touch the C++ code. xolotl represents compartment, conductance, synapse, and controllers as cpplab objects, which map to underlying C++ header files. In this way, properties of the xolotl network can be examined and changed using object-oriented paradigms. The object specifies the integrate function, not the other way around.

5.3 APPLICATIONS OF CPPLAB

NEED TO WRITE THIS

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

SG-S designed and implemented the core of the xolotl toolbox. AH contributed to the code base, created the online user documentation, and wrote the manuscript. EM supervised the project. All authors reviewed the paper.

FUNDING

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

ACKNOWLEDGMENTS

289 The authors would like to thank Mara CP Rue and Hillary Rodgers for beta-testing the `xolotl` software.
 290 Janis Li helped to prepare some conductance header files.

SUPPLEMENTAL DATA

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

DATA AVAILABILITY STATEMENT

292 The code to generate all figures is available at ([https://github.com/marderlab/](https://github.com/marderlab/xolotl-paper)
 293 `xolotl-paper`). `xolotl` is freely available at (<https://github.com/marderlab/xolotl>).

REFERENCES

- 294 Baker, Monya (2016) “Why Scientists Must Share Their Research Code”. In: *Nature*. DOI: 10.1038/
 295 nature.2016.20504.
- 296 Brette, Romain, Michelle Rudolph, Ted Carnevale, Michael Hines, David Beeman, James M. Bower, et al.
 297 (2007) “Simulation of Networks of Spiking Neurons: A Review of Tools and Strategies”. In: *Journal of*
 298 *Computational Neuroscience* 23.3, pp. 349–398. DOI: 10.1007/s10827-007-0038-6.
- 299 Butcher, J. C. (2016) “Numerical Differential Equation Methods”. In: *Numerical Methods for Ordinary*
 300 *Differential Equations*. Third. Wiley-Blackwell, pp. 55–142. DOI: 10.1002/9781119121534.
 301 ch2.
- 302 Connor, J. A. and C. F. Stevens (1971a) “Inward and Delayed Outward Membrane Currents in Isolated
 303 Neural Somata under Voltage Clamp”. In: *The Journal of Physiology* 213.1, pp. 1–19. pmid: 5575338.
- 304 — (1971b) “Voltage Clamp Studies of a Transient Outward Membrane Current in Gastropod Neural
 305 Somata”. In: *The Journal of Physiology* 213.1, pp. 21–30. DOI: 10.1113/jphysiol.1971.
 306 sp009365.
- 307 Dayan, Peter and L. F. Abbott (2001) *Theoretical Neuroscience*. Computational neuroscience. Cambridge,
 308 Mass.: Massachusetts Institute of Technology Press. xv+460.
- 309 Delorme, Arnaud and Simon J. Thorpe (2003) “SpikeNET: An Event-Driven Simulation Package for
 310 Modelling Large Networks of Spiking Neurons”. In: *Network: Computation in Neural Systems* 14.4,
 311 pp. 613–627. DOI: 10.1088/0954-898X_14_4_301. pmid: 14653495.
- 312 Destexhe, Alain and Thierry Bal (2009) *Dynamic-Clamp: From Principles to Applications*. Springer
 313 Science & Business Media. 428 pp.
- 314 Dethier, Julie, Guillaume Drion, Alessio Franci, and Rodolphe Sepulchre (2015) “A Positive Feed-
 315 back at the Cellular Level Promotes Robustness and Modulation at the Circuit Level”. In: *Journal of*
 316 *Neurophysiology*, jn.00471.2015. DOI: 10.1152/jn.00471.2015. pmid: 26311181.
- 317 Eklund, Anders, Thomas E. Nichols, and Hans Knutsson (2016) “Cluster Failure: Why fMRI Inferences
 318 for Spatial Extent Have Inflated False-Positive Rates”. In: *Proceedings of the National Academy of*
 319 *Sciences*, p. 201602413. DOI: 10.1073/pnas.1602413113. pmid: 27357684.
- 320 Hines, M. L. and N. T. Carnevale (1997) “The NEURON Simulation Environment”. In: *Neural*
 321 *Computation* 9.6, pp. 1179–1209. DOI: 10.1162/neco.1997.9.6.1179.
- 322 Hodgkin, A. L. and A. F. Huxley (1952a) “A Quantitative Description of Membrane Current and Its
 323 Application to Conduction and Excitation in Nerve”. In: *The Journal of Physiology* 117.4, pp. 500–544.
 324 pmid: 12991237.
- 325 — (1952b) “The Components of Membrane Conductance in the Giant Axon of Loligo”. In: *The Journal*
 326 *of Physiology* 116.4, pp. 473–496. pmid: 14946714.

- Hodgkin, A. L., A. F. Huxley, and B. Katz (1952) “Measurement of Current-Voltage Relations in the Membrane of the Giant Axon of Loligo”. In: *The Journal of Physiology* 116.4, pp. 424–448. pmid: 14946712.
- Jolivet, Renaud, Ryota Kobayashi, Alexander Rauch, Richard Naud, Shigeru Shinomoto, and Wulfram Gerstner (2008) “A Benchmark Test for a Quantitative Assessment of Simple Neuron Models”. In: *Journal of Neuroscience Methods* 169.2, pp. 417–424. DOI: 10.1016/j.jneumeth.2007.11.006. pmid: 18160135.
- Kernighan, Brian and Dennis M. Ritchie (1978) *The C Programming Language*. Prentice hall.
- Liu, Z., J. Golowasch, E. Marder, and L. F. Abbott (1998) “A Model Neuron with Activity-Dependent Conductances Regulated by Multiple Calcium Sensors”. In: *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* 18.7, pp. 2309–2320. pmid: 9502792.
- Marder, E. and L. F. Abbott (1995) “Theory in Motion”. In: *Current Opinion in Neurobiology* 5.6, pp. 832–840. pmid: 8805418.
- O’Leary, Timothy, Alex H. Williams, Jonathan S. Caplan, and Eve Marder (2013) “Correlations in Ion Channel Expression Emerge from Homeostatic Tuning Rules”. In: *Proceedings of the National Academy of Sciences* 110.28, E2645–E2654. DOI: 10.1073/pnas.1309966110. pmid: 23798391.
- Oh, Jiyeon and Donald A. French (2006) “Error Analysis of a Specialized Numerical Method for Mathematical Models from Neuroscience”. In: *Applied Mathematics and Computation* 172.1, pp. 491–507. DOI: 10.1016/j.amc.2005.02.028.
- Prinz, Astrid A (2006) “Insights from Models of Rhythmic Motor Systems”. In: *Current Opinion in Neurobiology* 16.6, pp. 615–620. DOI: 10.1016/j.conb.2006.10.001.
- Prinz, Astrid A. (2010) “Computational Approaches to Neuronal Network Analysis”. In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 365.1551, pp. 2397–2405. DOI: 10.1098/rstb.2010.0029. pmid: 20603360.
- Prinz, Astrid A., Cyrus P. Billimoria, and Eve Marder (2003) “Alternative to Hand-Tuning Conductance-Based Models: Construction and Analysis of Databases of Model Neurons”. In: *Journal of Neurophysiology* 90.6, pp. 3998–4015. DOI: 10.1152/jn.00641.2003. pmid: 12944532.
- Prinz, Astrid A., Dirk Bucher, and Eve Marder (2004) “Similar Network Activity from Disparate Circuit Parameters”. In: *Nature Neuroscience* 7.12, pp. 1345–1352. DOI: 10.1038/nn1352. pmid: 15558066.
- Sedano, T. (2016) “Code Readability Testing, an Empirical Study”. In: *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET) 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)* pp. 111–117. DOI: 10.1109/CSEET.2016.36.
- Sherfey, Jason S., Austin E. Soplata, Salva Ardid, Erik A. Roberts, David A. Stanley, Benjamin R. Pittman-Polletta, et al. (2018) “DynaSim: A MATLAB Toolbox for Neural Modeling and Simulation”. In: *Frontiers in Neuroinformatics* 12. DOI: 10.3389/fninf.2018.00010.
- Stimberg, Marcel, Dan F. M. Goodman, Victor Benichoux, and Romain Brette (2014) “Equation-Oriented Specification of Neural Models for Simulations”. In: *Frontiers in Neuroinformatics* 8. DOI: 10.3389/fninf.2014.00006.
- Stimberg, Marcel, Dan FM Goodman, Victor Benichoux, and Romain Brette (2013) “Brian 2 - the Second Coming: Spiking Neural Network Simulation in Python with Code Generation”. In: *BMC Neuroscience* 14 (Suppl 1) P38. DOI: 10.1186/1471-2202-14-S1-P38. pmid: null.
- Stodden, Victoria, Marcia McNutt, David H. Bailey, Ewa Deelman, Yolanda Gil, Brooks Hanson, et al. (2016) “Enhancing Reproducibility for Computational Methods”. In: *Science* 354.6317, pp. 1240–1241. DOI: 10.1126/science.aah6168. pmid: 27940837.
- Swensen, A. M. and E. Marder (2000) “Multiple Peptides Converge to Activate the Same Voltage-Dependent Current in a Central Pattern-Generating Circuit”. In: *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* 20.18, pp. 6752–6759. pmid: 10995818.
- (2001) “Modulators with Convergent Cellular Actions Elicit Distinct Circuit Outputs”. In: *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* 21.11, pp. 4050–4058. pmid: 11356892.

- 379 Turrigiano, G., G. LeMasson, and E. Marder (1995) “Selective Regulation of Current Densities Under-
- 380 lies Spontaneous Changes in the Activity of Cultured Neurons”. In: *The Journal of Neuroscience: The*
- 381 *Official Journal of the Society for Neuroscience* 15 (5 Pt 1) pp. 3640–3652. pmid: 7538565.
- 382 Vitay, Julien, Helge Ülo Dinkelbach, and Fred H. Hamker (2015) “ANNarchy: A Code Generation Ap-
- 383 proach to Neural Simulations on Parallel Hardware”. In: *Frontiers in Neuroinformatics* 9. DOI: 10 .
- 384 3389/fninf.2015.00019.
- 385 Xu, W., D. Xu, and L. Deng (2017) “Measurement of Source Code Readability Using Word Concrete-
- 386 ness and Memory Retention of Variable Names”. In: *2017 IEEE 41st Annual Computer Software and*
- 387 *Applications Conference (COMPSAC)* 2017 IEEE 41st Annual Computer Software and Applications
- 388 Conference (COMPSAC) vol. 1, pp. 33–38. DOI: 10.1109/COMPSAC.2017.166.

FIGURE CAPTIONS

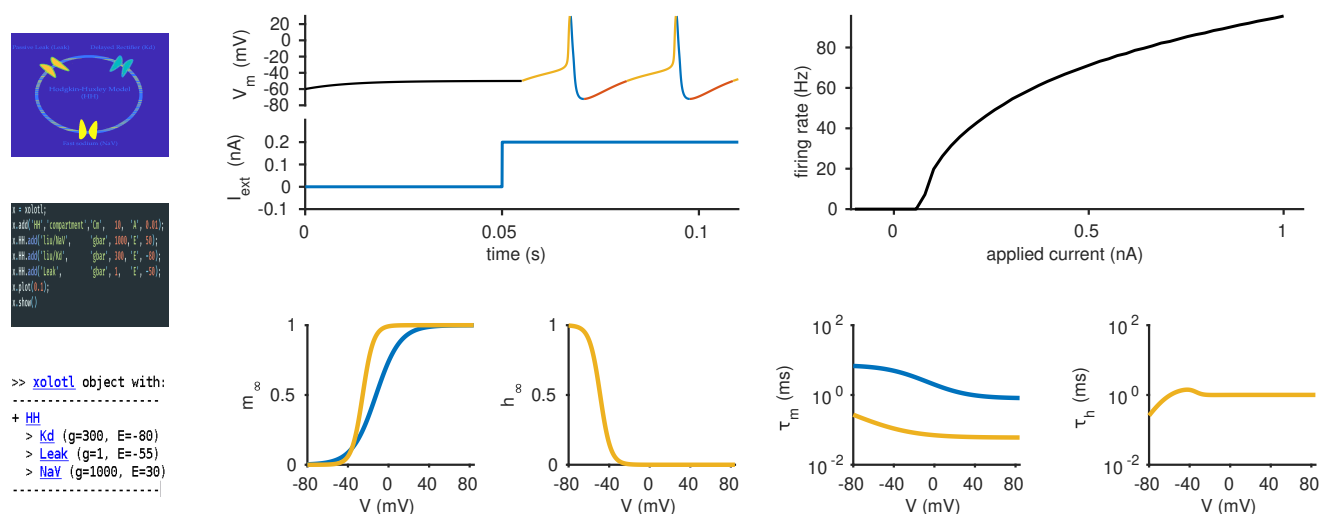


Figure 1: `xolotl` can quickly set up and simulate conductance-based models. (A) Cartoon of a Hodgkin-Huxley single-compartment neuron model with fast sodium, delayed rectifier, and leak currents. (B) Code snippet in MATLAB used to implement D, F-I. (C) `xolotl` schematic displayed in the MATLAB command prompt. (D) Simulated voltage trace of a Hodgkin-Huxley model with three conductances and 0.2 nA of injected current. Colors indicate the dominant current (gold is fast sodium, blue is delayed rectifier, red is leak). (E) Firing rate-input relation displaying firing rate as a function of injected current. (F-G) Steady-state gating functions for activation (m) and inactivation (h) gating variables. (H-I) Voltage-dependence of time constants for activation (m) and inactivation (h) gating variables.

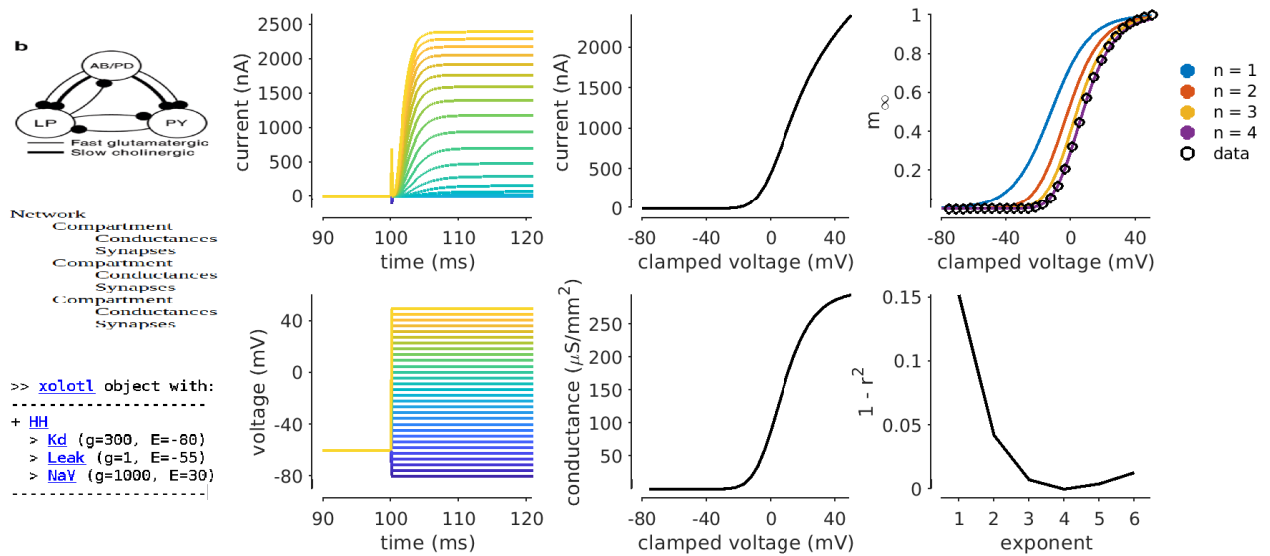


Figure 2: Simulating a voltage-clamp experiment. (A) Cartoon of a cell with delayed rectifier potassium conductance (Liu et al. 1998) with experimentally-fixed voltage. (B) Structure of `xolotl` object in A. (C) Code snippet depicting integration under voltage clamp. (D-E) Current response to steps in 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) Goodness of fit vs. exponent n , suggesting $n = 4$ as the best fit to the data.

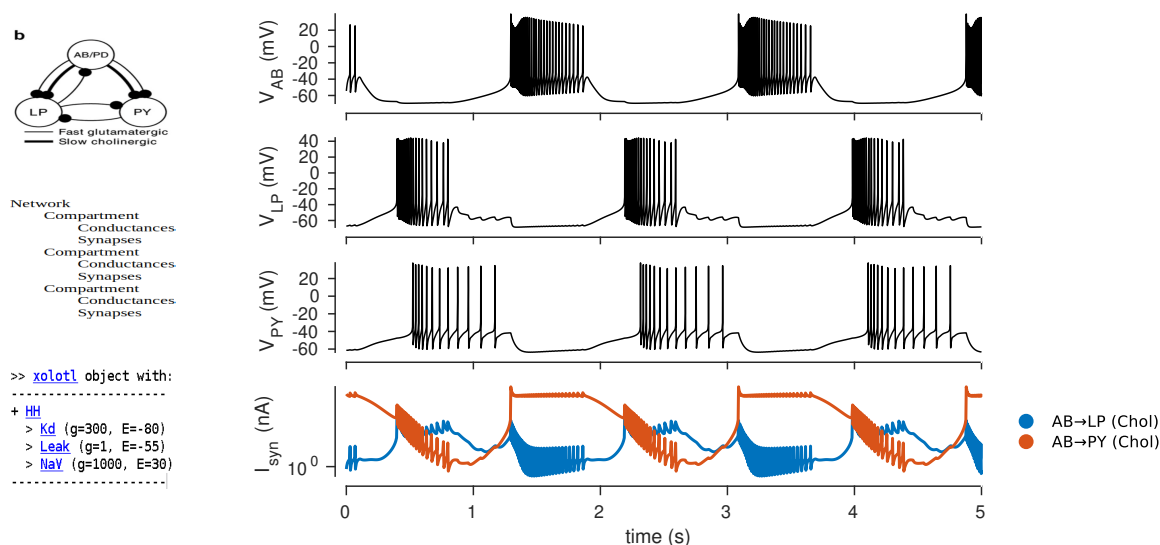


Figure 3: Simulating a network of conductance-based model neurons. (A) Diagram of a network model of the pyloric rhythm in the crustacean stomatogastric ganglion (Prinz *et al.* 2004). (B) Each neuron is modeled as a single compartment with 7-8 intrinsic conductances and 1-3 post-synaptic conductances. (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 can be obtained from the integration.

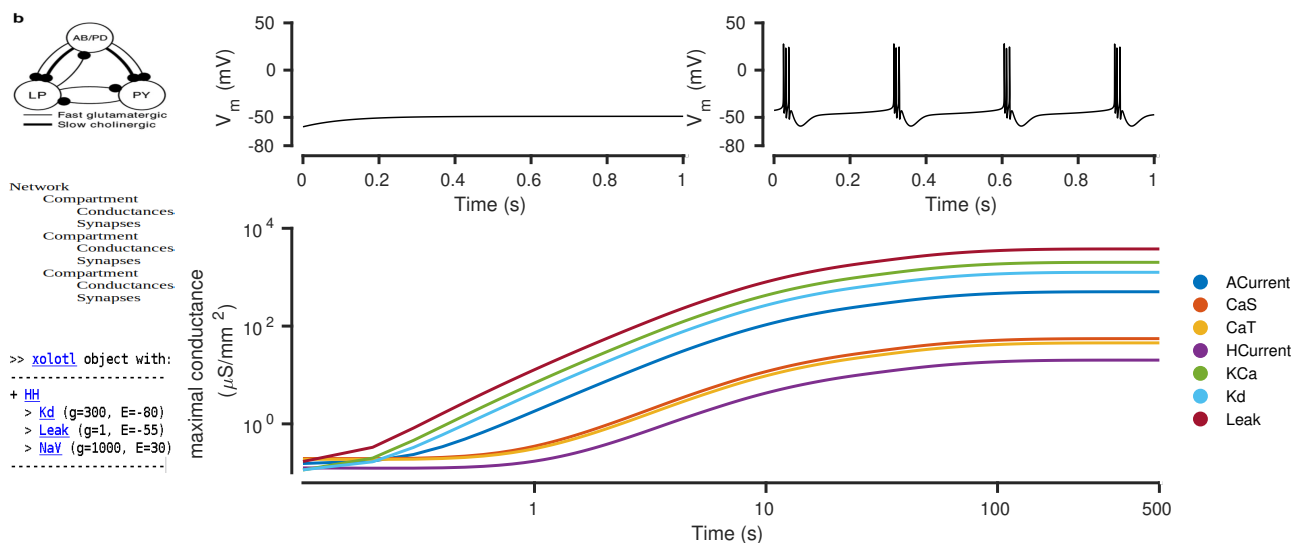


Figure 4: Simulating neurons under homeostatic regulation. (A) Cartoon of a model neuron (Liu et al. 1998) with integral control (O'Leary et al. 2013). (B) Hierarchical structure of a neuronal network considers controllers as components of compartments which act on conductances. (C) *xolotl* implements controllers as properties of conductances and synapses. (D) Calcium sensors change maximal conductances to move a neuron from quiescence to a bursting state. (E) Voltage trace shows regular bursting activity after integral control.

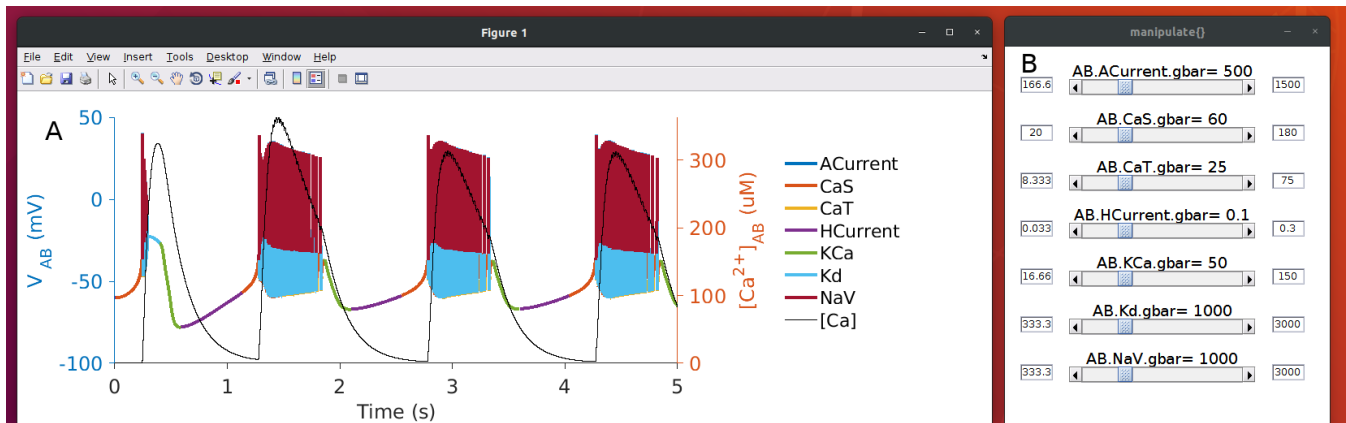


Figure 5: Using the GUI to manipulate neuron parameters. (A) Real-time output of the `plot` function displaying voltage (colored) and intracellular calcium (black) traces of a bursting neuron model (Astrid A. Prinz, Billimoria, and Eve Marder 2003; Astrid A. Prinz, Bucher, and Eve Marder 2004). Colors indicate the dominant current. (B) Sliders control the maximal conductances, which updates on the figure.

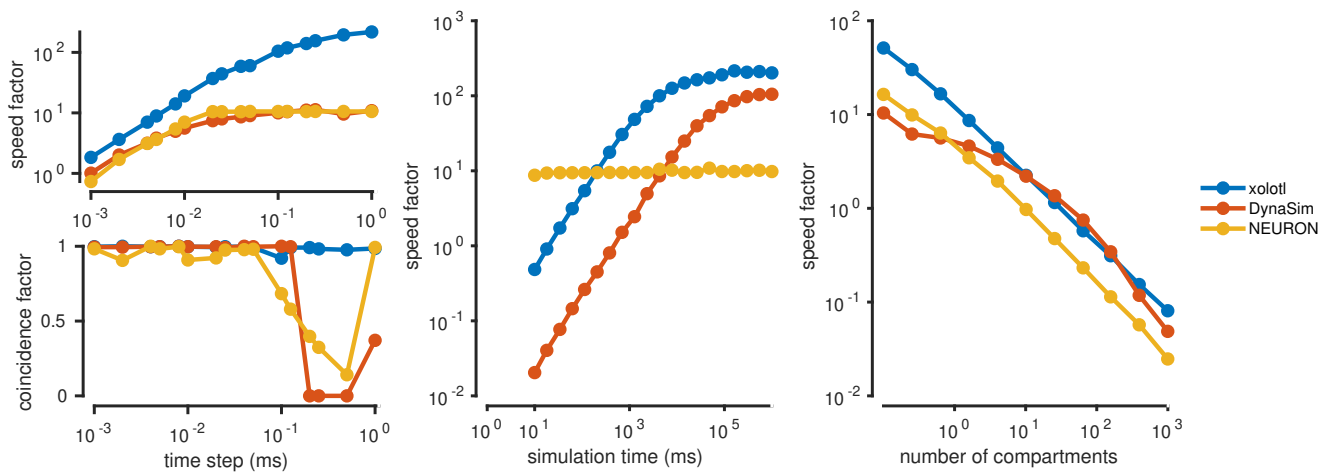


Figure 6: `xolotl` benchmarked against `DynaSim` and `NEURON`. (A) Ratio of 'simulated' time to runtime (speed factor) and accuracy, measured by spike train coincidence plotted against decreasing time-resolution. (B) Speed factor for models at increasing simulation times. (C) Speed factor over number of compartments.