xolotl Documentation

Srinivas Gorur-Shandilya and Alec Hoyland

Nov 08, 2018

Contents:

1 Quickstart 3

2 Frequency Asked Questions (FAQ) 5

3 Installing, Updating and Uninstalling 7

4 How do I... 9

5 Methods 15

6 Compiler Support 29

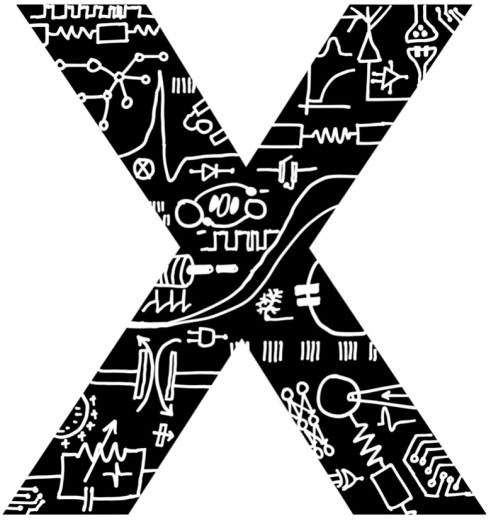
7 Units 33

8 Contributing to xolotl 35

9 Troubleshooting 37

10 License 39

11 Indices and tables 49



`xolotl` is a fast single-compartment and multi-compartment simulator written in `C++` with a `MATLAB` interface. Designed with a focus on ease-of-use, flexibility and speed, `xolotl` simulates conductance-based neuron models and networks.

CHAPTER 1

Quickstart

This quickstart guide will get you started using `xolotl` right away. It will cover installation and basic usage.

1.1 Installing

The easiest way to get `xolotl` is as a MATLAB toolbox. Click here to download it, and click on the downloaded file to install.

1.2 Creating a Hodgkin-Huxley Model

We will now create a `xolotl` object that describes a single-compartment neuron model with a fast sodium conductance, delayed rectifier potassium conductance, and passive leakage. The compartment has a membrane capacitance `'Cm'` of 10 nF/mm^2 and a surface area of 0.01 mm^2. These conductances come from Liu *et al.* 1998. `'gbar'` is the maximal conductance in uS/mm^2 and `'E'` is the reversal potential in mV.

```
% create the xolotl object
x = xolotl;

% add a compartment
x.add('compartment', 'HH', 'Cm', 10, 'A', 0.01);

% add conductances
x.HH.add('liu/NaV', 'gbar', 1000, 'E', 50);
x.HH.add('liu/Kd', 'gbar', 300, 'E', -80);
x.HH.add('Leak', 'gbar', 1, 'E', -40);
```

1.3 Simulate the Model

We simulate the model using the GUI to manipulate the leak conductance.

```
x.t_end = 1000; % ms
x.manipulate('*Leak*')
```

You should get a GUI that pops up showing the voltage trace, with sliders that allow you to vary parameters for the Leak conductance. Play with the sliders and see what happens!

CHAPTER 2

Frequency Asked Questions (FAQ)

2.1 Who Wrote xolotl?

`xolotl` was designed and written by Srinivas Gorur-Shandilya and Alec Hoyland in Eve Marder's laboratory at Brandeis University.

2.2 How Can I Contribute?

`xolotl` is far from complete and contributions are welcome. Check out our guide on how to report bugs, add conductances and mechanisms, or contribute to the code base.

2.3 Something is Broken, What Do I Do?

As the Hitchiker's Guide to the Galaxy reads, "DON'T PANIC". The most common error we see is an issue between `MATLAB`, the `MEX` compiler, and your computer. In that situation, a first step would be to check our compilers guide which has detailed fixes for some of the most common compiler issues we've seen.

Hint: If `GetMD5` fails, then it's definitely at least a compiler issue.

If it's not a compiler issue and you've checked all the typical programming pitfalls (syntax, spelling, your `MATLAB` path, cleared your workspace, etc...), try the following

```
% erases all the compiled binaries of xolotl networks
x.cleanup
```

If you're still having a problem, we might have fixed it already! Reinstall or update by running the `xolotl.update()` if you installed the toolbox or pulling from the repository if you're using `git`. Remember that if you're using `git` you should be sure to pull all the dependencies as well (`srinivas.gs_mtools`, `cpplab`, `puppeteer`).

2.4 How Do I Cite This?

Thanks for thinking of us! We'll be publishing later this year, and a preprint is available on bioRxiv.

CHAPTER 3

Installing, Updating and Uninstalling

There are multiple ways to install this toolbox, based on your level of expertise and the toolchain on your computer.

3.1 Installing: Download a MATLAB Toolbox

`xolotl` is available as a MATLAB toolbox. Click here to download it, and click on the downloaded file to install.

> **Warning:** If you wish to develop `xolotl` further, you should probably use git (see below).

3.2 Installing: Via Git

If you are comfortable with `git`, you can clone all the code and dependencies yourself:

```
git clone https://github.com/sg-s/srinivas.gs_mtools
git clone https://github.com/sg-s/puppeteer
git clone https://github.com/sg-s/xolotl
git clone https://github.com/sg-s/cpplab
```

You will have to manually set your `MATLAB` paths. Make sure you add the main folder for `puppeteer`, `cpplab`, and `xolotl`, and all subfolders of `srinivas.gs_mtools/src`.

> **Warning:** If you want to develop `xolotl` further, make sure you're running a non-Windows OS, and that you make the *pre-commit* git hook executable: `chmod a+x ./git/hooks/pre-commit`

3.3 Updating

In most cases, `xolotl` can update itself to the latest version using

```
xolotl.update()
```

If you installed using `git`, `xolotl` will attempt to do a `git pull` and update itself. If you installed it as a MATLAB toolbox, `xolotl` will delete the old toolbox, download the new one, and install that.

3.4 Uninstalling

If you installed `xolotl` as a MATLAB toolbox, you can easily uninstall it using

```
xolotl.uninstall()
```

Note that this doesn't do anything if you installed using `git`, or if you manually downloaded the files and linked them.

CHAPTER 4

How do I. . .

4.1 create a empty xolotl object?

```
x = xolotl;
```

This is a prerequisite to doing anything else. (You can name your object whatever you want, but this documentation will assume that you've named it `x`).

4.2 create a new compartment?

You can first create a new compartment using `cpplab`

```
AB = cpplab('compartment','vol',.01);
```

At this point you have created a free-floating object. You can inspect it just like you would any other `MATLAB` object:

```
AB

AB =

 compartment object with:

     hash : 44c3772
       Cm : 10
        A : 0.01
   radius : NaN
      vol : NaN
  Ca_average : NaN
  shell_thickness : NaN
  tree_idx : NaN
        V : -60
```

(continues on next page)

(continued from previous page)

```
  neuron_idx : NaN
  Ca_target : NaN
       Ca : NaN
   Ca_out : 3000
      len : NaN
```

and then add it to the `xolotl` object tree:

```
x.add('AB', AB);
```

A handy shortcut for this is:

```
x.add('compartment', 'AB', 'vol', .01)
```

This shortcut syntax will be used for the rest of the documentation, but remember that you can also do things the "long" way.

4.3 add a mechanism to a compartment?

Assuming you have a compartment called `AB` in your `xolotl` object, you can add the mechanism `CalciumMech1` through the `add` function and specify the value of the parameter `f`,

```
x.AB.add('CalciumMech1', 'f', 1.498)
```

4.4 add a conductance to a compartment?

Assuming you have a compartment called `AB` in your `xolotl` object,

```
x.AB.add('liu/NaV');
```

The string `liu/NaV` specifies the path within the `C++` folder that indicates where the header file for the thing that we want to add is.

Once again, you can inspect this object just like you would any other `MATLAB` object:

```
x.AB.NaV

ans =

 NaV object with:

       hash : 49007f5
          E : NaN
          m : NaN
       gbar : NaN
          h : NaN
```

As always, you can set these properties after-the-fact (e.g. `x.gbar = 100`), or when the conductance is added (e.g. `x.add('liu/NaV', 'gbar', 100)`).

4.5 find out what conductances are available?

Look in the folder yourself! All C++ headerfiles are contained in the C++ folder in the xolotl directory. If you are unsure where that is, type this in your MATLAB prompt:

```
fileparts(fileparts(which('xolotl')))
```

4.6 add a custom conductance?

The quickest way is to use the conductance class. The conductance class expects steady-state gating functions for activation and inactivation variables (m_inf and h_inf) and their respective time-constants (tau_m, tau_h). Whether the channel fluxes calcium (is_Ca) and whether it should use approximations for the gating functions rather than integrating (is_approx) can be set. In addition, you can set the default activation and inactivation variable initial conditions (default_m and default_h), and the default reversal potential (default_E). Finally, you should be sure to set the exponential fit parameters (p and q) so that the instantaneous conductance is gbar * m^p * h^q.

```
newCond = conductance;
newCond.m_inf = @m_inf;
newCond.h_inf = @h_inf;
...
newCond.generateCPPFile('name_of_conductance');
```

Alternatively, you can make your own custom conductances by editing a copy of the conductance templates found in ../xolotl/conductances/templates/. If you think it should be added to xolotl as a permanent feature, send us a message.

4.7 inspect the object I have created?

You can inspect any object by outputting it in the command window. For example, to inspect the whole xolotl object

```
>> xolotl object with
----------------------
+ HH
  > NaV (g=NaN, E=NaN)
----------------------
```

You can click on the linked (blue) text to inspect those properties, or reference them directly (e.g. x.AB.NaV).

4.8 connect two compartments using a synapse?

Connect two compartments with an electrical synapse

```
x.connect('AB', 'PD')
```

Connect two compartments with an electrical synapse and specify properties

```
x.connect('AB' 'PD', 'gbar', 100)
```

Connect two compartments with a glutamatergic synapse

```
x.connect('AB', 'LP', 'prinz/Glut')
```

Connect two compartments with a glutmatergic synapse and specify properties

```
x.connect('AB', 'LP', 'prinz/Glut', 'gbar', 100)
```

4.9 find out what synapse types are available?

Look in the folder yourself! All C++ headerfiles are contained in the `c++/synapses` folder in the `xolotl` directory. If you are unsure where that is, type this in your MATLAB prompt:

```
fileparts(fileparts(which('xolotl')))
```

4.10 inject current into a compartment?

Add a scalar, vector, or matrix to `x.I_ext`. This is interpreted as an injected current in nanoamperes.

Inject a constant current into all compartments

```
x.I_ext = 0.2;
```

Inject a constant current into one of two compartments

```
x.I_ext = [0.2 0];
```

Inject a variable current into one of two compartments

```
nSteps = x.t_end / x.dt;
I_ext  = zeros(nSteps, 2);
I_ext(:,1) = 0.2 * rand(nSteps, 1);
x.I_ext = I_ext;
```

4.11 voltage clamp a compartment?

Add a matrix to `x.V_clamp`. This is interpreted as an `nSteps x nComps` matrix of clamped voltage, where `nSteps` is the number of time-steps in the simulation, and `nComps` is the number of compartments.

Clamp the voltage and step it from -50 mV to 50 mV and back

```
nSteps = x.t_end / x.dt;
V_clamp = -50 * ones(nSteps, 1);
V_clamp(ceil(nSteps/2), 1) = 50;
V_clamp(ceil(nSteps*3/4), 1) = -50;
x.V_clamp = V_clamp;
```

4.12 specify simulation time step and other integration parameters?

Specify the time step by setting `x.dt` in milliseconds. Specify the simulation time by setting `x.t_end` in milliseconds. Injected current and voltage clamp are determined by setting `x.I_ext` and `x.V_clamp`.

Set the simulation time to 5000 ms

```
x.t_end = 5000
```

Set the time step to 0.1 ms

```
x.dt = 0.1
```

4.13 integrate the model and obtain outputs?

Integrate the model

```
[V, Ca, cont_states, currents, syn_currents] = x.integrate
```

`V` is the voltage trace as a matrix `nSteps x nComps`. `Ca` is the intracellular calcium concentration trace. `cont_states` is the controller states and controlled parameters as time series. `currents` and `syn_currents` are the time traces of all the currents and synaptic currents, in the order that they are displayed in the serialized `xolotl` object (e.g. how `x` displays them in the command window).

4.14 debug a model or simulation?

`xolotl` has a debug mode that can be turned on using

```
x.verbosity = 1;
```

CHAPTER 5

Methods

This page lists the methods of the `xolotl` class in `MATLAB`. This can serve as a reference for advanced usage.

In the rest of this documentation we will assume a `xolotl` object named `x` that can be created using

```
x = xolotl;
```

Hint: You can list all the methods of `xolotl` object `x` by

```
x.methods;
```

5.1 show

shows activation functions and timescales of any conductance. Usage

```
x.show('cond_name')
```

'cond_name' must be a string that resolves to a valid C++ file that describes a conductance.

5.1.1 Example

```
    % compare some channels from the Prinz et al. paper
xolotl.show('prinz/NaV')
xolotl.show('prinz/Kd')
xolotl.show('prinz/KCa')
```

5.1.2 See Also

- plot
- getGatingFunctions

5.1.3 Test coverage

`show` is tested in:

5.2 compile

compiles a executable binary form a transpiled `C++` file. These are stored in your `xolotl` directory. `xolotl` automatically compiles when t needs to. You can turn this functionality off by setting

```
x.skip_hash = true;
```

In addition, creating a `xolotl` object through a function call does not utomatically hash and compile. In this case, you should use `x.md5hash`.

If you turn hashing off, `xolotl` might not compile

5.2.1 See Also:

- transpile
- cleanup

5.2.2 Test coverage

`compile` is tested in:

5.3 contributingCurrents

This static method calculates the contributions of each current at every point in a voltage race. This is used internally in `xolotl.plot` to color voltage traces. The syntax is

```
curr_index = xolotl.contributingCurrents(V, I)
```

where V is a vector of voltages, I is the corresponding matrix of currents

5.3.1 See Also

- plot
- manipulate

5.3.2 Test coverage

`contributingCurrents` is tested in:

5.4 matrixCost

a static method to compute the distance between two LeMasson matrices. This is a useful way to determine how similar two voltage traces are.

Usage

```
C = matrixCost(M1,M2)
```

where `M1` and `M2` are two matrices returned by xolotl.V2matrix() and represent discretized probability distributions of a derivative-embedded attractor of the voltage trace.

5.4.1 See Also

LeMasson G, Maex R (2001) Introduction to equation solving and parameter fitting. In: De Schutter E (ed) Computational Neuroscience: Realistic Modeling for Experimentalists. CRC Press, London pp 1–21

- V2matrix

5.4.2 Test coverage

`matrixCost` is tested in:

5.5 checkCompartmentName

is used internally by `xolotl` to verify that the compartment name you are using is valid and legal. This method is called every time you add a compartment to a `xolotl` object. Usage

```
ok = checkCompartmentName(self,comp_name)
```

5.5.1 See Also

- add

5.5.2 Test coverage

`checkCompartmentName` is tested in:

5.6 benchmark

performs a quick benchmarking of a given `xolotl` model. `benchmark` first varies the simulation time step, and measures how quickly the model integrates. It then varies `t_end`, and measures how fast it integrates at a fixed `sim_dt`. Usage

```
x.benchmark;
```

5.6.1 Test coverage

`benchmark` is tested in:

5.7 V2matrix

a static method that converts a voltage trace into a LeMasson matrix. Usage

```
[M, V_lim, dV_lim] = V2matrix(V, V_lim, dV_lim)
```

where V is a vector (a voltage time series), and `V_lim` and `dV_lim` are two-element vectors that specify the lower and upper bounds of `V` and `dV`

This static method allows you to create a delay-embedding of a voltage trace, and then discretize the space and count the number of points in each bin. The resultant matrix is sometimes called a LeMasson matrix. `M` is the LeMasson matrix, which is always of size `101x101`.

If you do not specify `V_lim` and `dV_lim`, they will be computed automatically and returned.

5.7.1 See Also

LeMasson G, Maex R (2001) Introduction to equation solving and parameter fitting. In: De Schutter E (ed) Computational Neuroscience: Realistic Modeling for Experimentalists. CRC Press, London pp 1–21

- matrixCost

5.7.2 Test coverage

`V2matrix` is tested in:

5.8 uninstall

A static method that uninstalls your installation of xolotl in place. If you installed using git, `xolotl` will attempt to use git to uninstall itself. Usage

```
xolotl.uninstall
x.uninstall
```

5.8.1 Test coverage

`uninstall` is tested in:

5.9 plotgbars

makes a stem plot of conductance densities in a given compartment. Usage

```
x.plotgbars('compartment_name');
x.plotgbars(axes_handle,'compartment_name');
```

5.9.1 Test coverage

`plotgbars` is tested in:

5.10 update

A static method that updates your installation of xolotl in place. If you installed using git, `xolotl` will attempt to use git to update itself. Usage

```
xolotl.update
x.update
```

5.10.1 Test coverage

`update` is tested in:

5.11 copy

copies a xolotl object. `copy` creates an identical copy of a xolotl object that can be manipulated seperately. Both copies will use the same binary to integrate, unless you add a new component to one of them. Syntax

```
x2 = copy(x);
```

Some read-only properties in a xolotl object may not be copied over.

Warning: Do not make vectors of `xolotl` objects, as it may lead to undefined behavior.

5.11.1 Test coverage

`copy` is tested in:

5.12 reset

Resets a xolotl object to some default state. Usage

```
x.reset()
x.reset('snap_name')
```

reset called without any arguments resets the model as best as it can – voltages are set to -60 mV, Calcium in every compartment is set to the internal value, and the gating variables of every conductance are reset.

`reset` can also be called with a string argument, which is the name of a snapshot previously stored in the model object. Then, `reset` reconfigures the parameters of the model to match that snapshot. This is useful for working with a model, changing parameters, evolving it, and then coming back to where you started off from.

5.12.1 Example

```
% assuming a xolotl object is set up
x.integrate;
x.snapshot('base');
x.set('*gbar') = 1e-3; % turn off all conductances
x.integrate;
% now go back to original state
x.reset('base')
```

5.12.2 See Also

- snapshot

5.12.3 Test coverage

`reset` is tested in:

5.13 snapshot

Saves the current state of a `xolotl` object for future use. Usage

```
x.snapshot('snap_name')
```

Creating two snapshots with the same name will overwrite the first.

5.13.1 Example

```
% assuming a xolotl object is set up
x.integrate;
x.snapshot('base');
x.set('*gbar') = 1e-3; % turn off all conductances
x.integrate;
% now go back to original state
x.reset('base')
```

5.13.2 See Also

- reset

5.13.3 Test coverage

`snapshot` is tested in:

5.14 setup

A static method that allows you to set up compilers on some operating systems. You need to run this only once. If xolotl works, there is no need to run this.

Usage

```
xolotl.setup
x.setup
```

5.14.1 Test coverage

`setup` is tested in:

5.15 plot

Makes a plot of voltage and calcium time series of all compartments. The default option is to color the voltage traces by the dominant current at that point using `contributingCurrents` and to also show the Calcium concentration on the same plot. Usage

```
x.plot()
```

If you want to turn off the colouring, or to hide the Calcium concentration, change your preference using

```
setpref('xolotl','plot_color',false)
setpref('xolotl','show_Ca',false)
```

5.15.1 See Also

- manipulate
- contributingCurrents

5.15.2 Test coverage

`plot` is tested in:

5.16 getGatingFunctions

static method of `xolotl` that returns function handles that represent the gating and activation functions of a particular conductance. Example use

```
[m_inf, h_inf, tau_m, tau_h] =  getGatingFunctions(conductance)
```

where `conductance` is a string that specifies a conductance C++ header file. The outputs are function handles that can be evaluated independently. This method is used internally in `xolotl.show()`

5.16.1 See Also

- show

5.16.2 Test coverage

`getGatingFunctions` is tested in:

5.17 cleanup

A static method that cleans up all transpiled `C++` and compiled binary files. Usage

```
xolotl.cleanup
x.cleanup
```

Use of this method will trigger a warning every time it is called. You do not need to use this in normal use, but can call this to force a recompile, or to delete old and unused binaries.

5.17.1 Test coverage

`cleanup` is tested in:

5.18 integrate

integrates a `xolotl` model. Usage

```
V = x.integrate;
I_clamp = x.integrate;
[V, Ca] = x.integrate;
[V, Ca, mech_state] = x.integrate;
[V, Ca, mech_state, I] = x.integrate;
[V, Ca, mech_state, I, syn_state] = x.integrate;
```

`integrate` will return different outputs as shown above. Unless you need every output, it is recommended to skip it, as it makes the integration faster (and reduces the memory footprint).

5.18.1 Explanation of outputs

- `V` Voltage trace of every compartment. A matrix of size (nsteps, n_comps)
- `I_clamp` also returned in the first argument, this is the clamping current when a compartment is being voltage clamped. This can be inter-leaved with the voltage of other, non-clamped compartments.

- `Ca` Calcium concentration in every cell and the corresponding `E_Ca` (reversal potential of Calcium). A matrix of size (nsteps, n_comps)
- `mech_state` a matrix representing every dimension of every mechanism in the tree. This matrix has size (nsteps, NC), where NC depends on the precise controllers used, and is automatically determined.
- `I` the currents of every ion channel type in the model. This is a matrix of size (nsteps, n_cond)

5.18.2 Test coverage

`integrate` is tested in:

5.19 transpile

Generate a C++ file that constructs the model, integrates it, and moves parameters and data from MATLAB to C++ and back. Usage

```
x.transpile;
```

Warning: `transpile` assumes that your `xolotl` object has a valid hash. Empty hashes will throw an error.

5.19.1 Example

```
% assuming a xolotl object is set up
x.transpile;

% now view the transpiled code
x.viewCode;
```

Warning: You should generally never use `transpile` since `xolotl` will automatically transpile and compile code for you. Manually transpiling will hinder performance.

5.19.2 See Also

- compile
- viewCode

5.19.3 Test coverage

`transpile` is tested in:

5.20 viewCode

view the C++ code generated by xolotl.transpile that constructs the model and integrates it

```
x.viewCode;
```

5.20.1 See Also:

- transpile

5.20.2 Test coverage

`viewCode` is tested in:

5.21 add

adds a `cpplab` object to a `xolotl` object.

The add method is the most important way you construct models. Usage

```
x.add(compartment,'comp_name')
x.add('compartment','comp_name')
x.add('compartment','comp_name',...)
```

There are two primary ways of using `add`. The first is to first construct a `cpplab` object (here called AB), and then add it to the `xolotl` object using `x.add(AB,'AB')`. `xolotl` requires that every compartment is named, and the name has to be specified as a string argument.

5.21.1 Test coverage

`add` is tested in:

5.22 findNSpikes

static method of `xolotl` that computes the number of spikes in a voltage trace. Example use

```
N = xolotl.findNSpikes(V);
N = xolotl.findNSpikes(V, on_off_thresh)
```

`V` is a vector of voltages, and `on_off_thresh` is an optional argument that determines the threshold above which a voltage fluctuation is considered a spikes. The default is 0 mV.

5.22.1 See Also

- findNSpikeTimes

5.22.2 Test coverage

`findNSpikes` is tested in:

5.23 manipulateEvaluate

This method is used to update the `xolotl` object every time a slider is moved in the manipulate window. This is used internally in `xolotl.manipulate`. You should not need to use this by itself.

5.23.1 See Also

- manipulate

5.23.2 Test coverage

`manipulateEvaluate` is tested in:

5.24 connect

Connects two compartments with a synapse. The basic syntax is

```
x.connect('Comp1', 'Comp2', 'SynapseType', ...)
```

The first two arguments are the presynaptic and postsynaptic compartment names. For example

```
% connects two different neurons with an electrical synapse
x.connect('AB', 'LP')
```

Axial synapses are a special type of electrical synapse that are created between spatially-discrete compartments in a morphological structure. Electrical and axial synapses differ in how they are integrated (see Dayan & Abbott 2001, Ch. 5-6).

`connect` defaults to an axial synapse when the type of synapse is not specified and either compartment has a defined `tree_idx` (which identifies the compartment as a part of a multi-compartment neuron model). Otherwise, the created synapse is electrical.

```
% create an (electrical or axial) synapse between AB and LP with gbar f NaN
x.connect('AB', 'LP')
% create an (electrical or axial) synapse between AB and LP with gbar f 10
x.connect('AB', 'LP', 10)
```

The most common way to produce a synapse is to pass the synapse type and hen any properties. This is used to create chemical synapses. For example, o add a glutamatergic synapse (from Prinz *et al.* 2004) between `AB` and `LP` with a maximal conductance of 100:

```
x.connect('AB', 'LP', 'prinz/Glut', 'gbar', 100)
```

Synapses can also be connected by passing a `cpplab` object to the `connect` method

```
% create a synapse using the cpplab object 'syn_cpplab'
x.connect('AB', 'LP', syn_cpplab)
```

The following properties can be specified

Name	PropertyName
Maximal conductance	`gbar`
Reversal potential	`E`
Activation variable	`s`

5.24.1 Test coverage

`connect` is tested in:

5.25 slice

`slice` partitions a cylindrical compartment into N slices. Usage

```
x.slice('comp_name',N)
```

The compartment to be sliced must explicitly be a cylindrical section, i.e., it must have a defined length and radius. `slice` cuts the cylinder along the axis, and connects each slice with `Axial` synapses. This object can then be treated as a multi-compartment model, and `xolotl` will integrate it using the Crank-Nicholson scheme reserved for multi-compartment models.

5.25.1 Example

```
% assuming there is a compartment called 'Dendrite'
xolotl.slice('Dendrite',10)
```

5.25.2 See Also

- connect

5.25.3 Test coverage

`slice` is tested in:

5.26 findNSpikeTimes

static method of `xolotl` that returns a vector of spike times from a voltage trace. Spikes are defined as voltage crossings across a threshold. Example use

```
spiketimes = xolotl.findNSpikeTimes(V,n_spikes,on_off_thresh);
```

`V` is a vector of voltages, and `on_off_thresh` is an optional argument that determines the threshold above which a voltage fluctuation is considered a spikes. The default is 0. `n_spikes` is the number of spikes it should look for, and `spiketimes` will always be a vector `n_spikes` elments long.

5.26.1 See Also

- findNSpikes

5.26.2 Test coverage

`findNSpikeTimes` is tested in:

5.27 manipulate

method that allows you to manipulate some or all parameters in a model hile visualizing its behaviour. Usage

```
x.manipulate();
x.manipulate('some*pattern')
x.manipulate({'parameter1','parameter2'})
```

The simplest way to use `manipulate` is to simply call it with no arguments. By default, all the parameters are linked to sliders that you can play with. In models with a large number of parameters, this can get messy. You can selectively only manipualte some parameters whose names match a pattern using `x.manipulate('some*pattern')`

5.27.1 Test coverage

`manipulate` is tested in:

5.28 rebase

Configures some internal house-keeping settings. This is called every time a new object is created. You probably don't ever have to use this, unless you copy `xolotl` objects across computers with different file systems or operating systems. Usage

```
x.rebase()
```

5.28.1 Test coverage

`rebase` is tested in:

CHAPTER 6

Compiler Support

`xolotl` simulates models by running compiled `C++` code in ``MATLAB``. This process requires a `C++` compiler that is linked to the `MATLAB` executable (`mex`) system.

6.1 Compiling on Microsoft Windows

For Windows systems, it is best to use the `MinGW-w64` compiler, which uses the GNU `gcc` toolchain. You should get this from the Add-Ons menu in `MATLAB` if possible, and the file exchange if not. More details can be found **here_**.

6.2 Compiling on MacOS & Linux

Compilers for which `xolotl` has been known to work (or not). Note that this has nothing to do with `xolotl`, but rather reflects whether `mex` works with these compilers on these OSes.

OS	Compiler	Version	Works
macOS 10.13.6	clang	Apple LLVM version 9.1.0	✓
macOS 10.12.6	clang	Apple LLVM version 9.0.0	✓
Ubuntu 18.04 LTS	g++	8.1	✓
Ubuntu 18.04 LTS	g++	7.3	✓
Ubuntu 17.10	g++	7.3	
Ubuntu 17.04	g++	7.0.1	
Ubuntu 17.10	g++	6.4	
Ubuntu 17.04	g++	6.3	
Ubuntu 17.04	g++	4.8.5	✓
Ubuntu 17.10	g++	4.8	✓
Ubuntu 16.04.3 LTS	g++	6.3	✓
Manjaro Linux	g++	8.1	✓
Manjaro Linux	g++	7.3	✓

6.3 Compiling on Linux

There are a couple of quirks specific to using the MEX compiler on Linux machines. First, make sure that you have a C++ compiler installed. We recommend g++, which is part of the GCC software project.

To get the requisite compiler on Debian/Ubuntu systems, do something similar to

```
sudo apt-get install g++
```

On Arch-based systems such as Manjaro, do something similar to

```
sudo pacman -Syu gcc6
```

Hint: MATLAB can be particular about which version of g++ it works with. For best results, use the compiler version recommended by MATLAB. In addition, it's best to point MATLAB towards the system compiler, rather than one installed through distributions like Anaconda.

On certain Linux distributions (Arch-based in particular), MEX cannot identify the installed g++ compiler, even when it exists on your path. The error looks something like this:

Even though the compiler exists on your path,

```
>> !which g++
/bin/g++
>> !g++ -dumpversion
8.1.0
```

MATLAB cannot find the intended C++ compiler.

```
>> mex -setup C++
Error using mex
No supported compiler was found.
```

MATLAB recommends changing your PATH so that you default to an older version of g++. This is not strictly necessary; MATLAB can still compile using MEX with newer versions of g++ in most cases. Generally, downgrading to an older version of g++ doesn't solve this problem.

There is a relatively simple fix however. Credit goes to github user bonanza123 for figuring it out.

- First download the proper version of gcc/g++. If you use a package manager there are generally legacy versions under gcc-VERSION, where VERSION is the version number (e.g. 6). You can also find them **here_**.
- Second change the mex_LANG_glnxa64.xml specification file, where LANG is either C or C++. This is typically found at ~/.matlab/R2018a/mex_C_glnxa64.xml where R2018a is the version of MATLAB and C is the name of the language.
- Replace all references to $GCC with the path to the soft link to your gcc compiler (e.g. /usr/bin/gcc-6). If you don't have a soft link to your compiler set up (i.e. which gcc doesn't tell you the path to the link), then you have to set one up.
- Repeat this process for the mex_C++_glnxa64.xml file. It should be in the same location as the C-specific file.
- Sometimes MATLAB doesn't generate the C++ .xml file, causing a lot of errors. If it doesn't exist, copy the C version of the file, rename it to mex_C++_glnxa64.xml, and replace all references to gcc with g++, so that MATLAB knows to use the right compiler.

The problem is fixed if you see something like this in MATLAB

```
>> mex.getCompilerConfigurations('C++')
ans =

CompilerConfiguration with properties:

           Name: 'g++'
   Manufacturer: 'GNU'
       Language: 'C++'
        Version: ''
       Location: '/usr/bin/g++-6'
      ShortName: 'g++'
       Priority: 'A'
        Details: [1×1 mex.CompilerConfigurationDetails]
     LinkerName: ''
  LinkerVersion: ''
        MexOpt: '/home/alec/.matlab/R2018a/mex_C++_glnxa64.xml'
```

CHAPTER 7

Units

`xolotl` doesn't come with a automatic unit handling system like in `NEURON`, `BRIAN`, or in `Julia`, so you have to pay attention and make sure your parameters are in the right units. The following are default units for various parameters.

Variable	**Units**
Length of cell	mm
Surface area of cell	mm^2
Volume of cell	mm^3
Voltage	mV
Specific capacitance of membrane	nF/mm^2
Specific conductance of channel	μS/mm^2
Synaptic strength	nS
Current injected into cell	nA
Axial resistivity	MΩ mm
Calcium concentration	μM
temperature	K

CHAPTER 8

Contributing to xolotl

`xolotl` is far from feature complete, and your contributions are welcome.

8.1 Reporting Bugs

- Is it a bug? Are you sure the bug persists after you run `transpile` and `compile` and `xolotl.cleanup`?
- Describe what the expected behavior is, and what the actual behavior was

8.2 Requesting Features

- Describe what you want
- Describe why you want it
- List papers that describe this mechanism, or original research that describes the feature you want

8.3 Adding New Conductances/Synapses/Controllers

- Look at existing conductances/synapses/controllers and use them as a guideline
- If you're making a new conductance, put them in `c++/conductances/<first_author_name>`
- Make sure you add a reference to the paper you're getting the conductance details from in a comment at the top of the file
- Send us a pull request

CHAPTER 9

Troubleshooting

9.1 On macOS, I get an annoying warning saying "xcrun: error: SDK "macosx10.13.4" cannot be located"

Run the following in your shell (not the MATLAB prompt):

```
sudo xcode-select -s /Applications/Xcode.app
```

9.2 On macOS, I get a warning saying that "Warning: Xcode is installed, but its license has not been accepted."

First, make sure you have XCode installed (not just the Command Line Tools – the whole thing). You can get this from the Mac App Store. Then, open XCode and accept the license. You will have to do this only once.

9.3 I ran the quickstart, but I don't see anything

Are you using a tiny screen? Some UI elements may go out of the frame on very small screens. To fix this, acquire the handle to the figure and change the position property. For example

```
x.manipulate;
manip = gcf;
manip.Position = [100 100 34 56];
```

9.4 I get an error saying I don't have a compiler

You need a C/C++ compiler. You need to follow MATLAB's instructions on how to get one, how to install one, and how to configure one. It may be helpful to also see our advice on compilers.

CHAPTER 10

License

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change

the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free

programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does

not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange,

in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

CHAPTER 11

Indices and tables

- genindex
- modindex
- search