neuronvisio Documentation

Release 0.3.5

Michele Mattioni

November 29, 2009

CONTENTS

CHAPTER

ONE

WHAT IS IT

NeuronVisio is a GTK2 user interface for NEURON simulator enviroment. NeuronVisio connect with NEURON using the new python NEURON interface.

CHAPTER

TWO

FEATURES

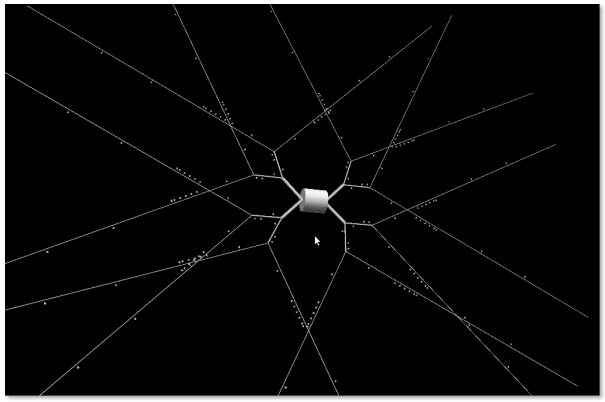
- 3D visualization of the model with the possibility to change it runtime
- Creation of vectors to record any variables present in the section
- Pylab integration to plot directly the result of the simulation
- Exploration of the timecourse of any variable among time using a color coded scale
- the GUI runs in its own thread so it's possible to use the console (strongly suggested ipython)

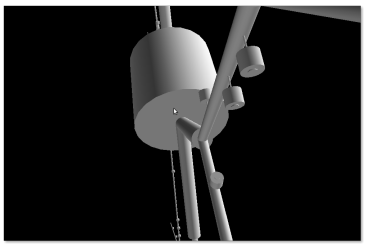
CHAPTER

THREE

QUICK OVERVIEW

Quick overview of the 3D capabilites. More [screenshots available](screenshots.html).





CHAPTER

FOUR

HELP AND DEVELOPMENT

4.1 Install

- To **install** Neuronvisio check the *Install*
- To **browse** the code online go to the github repo
- To **download and install** the code from github check the *Source Code* section
- To **submit a bug** use the tracker

4.2 MailingList

There is a google group to ask for help or send patches.

CHAPTER

FIVE

SITE MAP

5.1 Install

5.1.1 Requirements

To install NeuronVisio you need to satisfy the following dependencies

- pygtk: http://www.pygtk.org/
- visual: http://vpython.org/
- matplotlib: http://matplotlib.sourceforge.net/

and of course NEURON.

5.1.2 Ubuntu and friends

On Ubuntu you can easily install all the requirements using apt-get with:

```
sudo apt-get install python-numpy python-gtk2 python-visual python-matplotlib
```

and then add the Neuronvisio PPA on launchpad adding the repositories:

```
deb http://ppa.launchpad.net/mattions/neuronvisio/ubuntu karmic main
deb-src http://ppa.launchpad.net/mattions/neuronvisio/ubuntu karmic main
```

adding the key:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 4B2C6C7E
```

updating and installing:

```
sudo update
sudo install neuronvisio
```

If you are running a different flavour of GNU/Linux, like Fedora for example, just install the requirements with your package manager, then go to the *Package Install*.

5.1.3 Mac OS X

You need to install the requirements by yourself, because there is no package manager able to do it for you. I suggest you to get an Ubuntu. Anyway, for the brave, I'll give here some links to make this work easier for you:

- GTK for MAC : this is the GTK port for MAC
- Visual for MAC : there is a dedicated installer

- Matplotlib for MAC : Install the superpack and you will get Numpy, Scipy, and matplotlib.

If you have all this stuff installed then proceed to the Package Install.

5.1.4 Windows

Seriously? As for Mac, you need to install the requirements by yourself, because there is no package manager able to do it for you. It can be done but it's really painful. I suggest you to get an Ubuntu. Anyway, for the brave:

- PyGTK stack: To get this working you need to build the GTK, libglade, and PyGTK and install python
- Visual Python: You can install the visual package with the install
- Matplot and numpy: You need to compile everything.

If you have all this stuff installed then proceed to the Package Install.

5.1.5 Package Install

If you have pip installed and all the requirements are already met you can install neuronvisio and a really handy way:

```
pip install neuronvisio
```

Without pip, if you met all the requirements it's still pretty easy. Download the lates neuronvisio.tgz file from Neuronvisio's PyPI page, untar it and run:

```
python setup.py install
```

5.1.6 Legacy releases

You can find all the old Neuronvisio releases on github repo

5.1.7 Source Code

The source code is on github at this address and git is used as software management tool

To install from the git just clone the repo:

```
git clone git://github.com/mattions/neuronvisio.git
```

and then run:

```
python setup.py install
```

5.2 Getting Started

5.2.1 How does it work

You need to use NeuronVisio from an _ipython console started with the pylab switch:

```
ipython -pylab
```

To use the NeuronVisio module, after you have installed you should import with:

```
from neuronvisio.controls import Controls
controls = Controls()    # starting the GUI
```

The Control class run the main loop of the application with all the GUI activities in its own thread. The console is ready for input so you can enter your command to the prompt as you would do normally when using _NEURON.

5.2.2 How to integrate NeuronVisio with your code

The integration is rather simple and you can use either the python or the hoc scripts that you already have.

Python integration

If you have a model written in python, just import the module on top of your script. The simple example (in the example directory) give you an idea how to do it.

A classical template is:

```
from nrnvisio.controls import Controls
controls = Controls()    # starting the GUI
from neuron import h
# Your model here
```

Hoc Intergration

You have to load your hoc script using the python interface of _NEURON. The pyramidal example gives an idea how to integrate existent _NEURON model with it.

A classical template is:

```
import nrnvisio
from neuron import h
controls = nrnvisio.Controls()
h.load_file('path/to/my_model.hoc')
```

5.2.3 NeuronVisio features

Visualization

To visualize you model after you loaded you have to click the Draw button.

How to rotate

Hold the third button of your mouse (usually clicking the wheel) and move the mouse.

How to zoom

Hold the right button of your mouse and move the mouse.

How to move

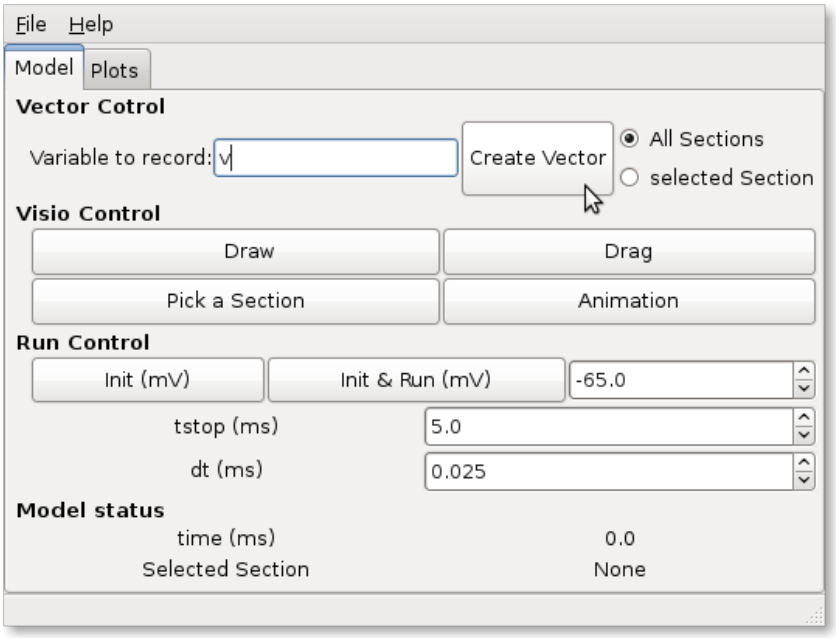
Click on *Drag* button and then pick a section of your model to move it.

5.2.4 Plotting the simulation results

Creating the vectors

To plot the simulation's results you first have to create a Vector (or more than one) to record the variable that you are interested in.

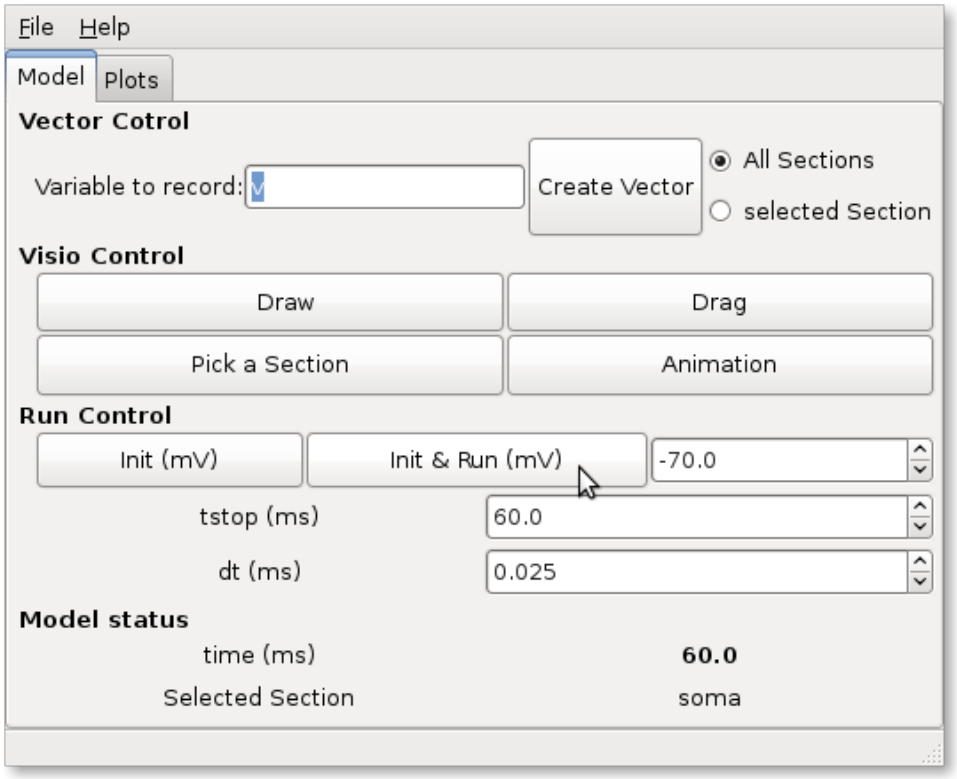
For example if you are interested in the voltage you have to insert *v* in the 'Variable to record and click record'.



If you want to create a vector for only one section just pick that section clicking on 'Pick a Section' and then select the section on the GUI.

Run the simulation

The simulation can be run clicking on the *Init & Run* button. It will run until the tstop.

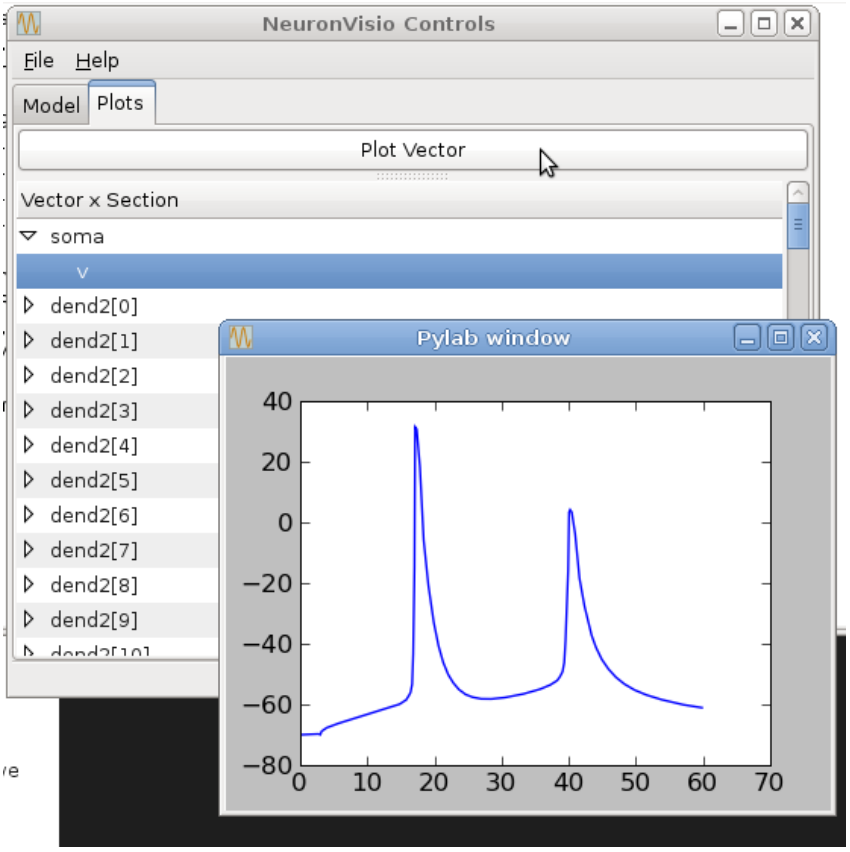


Plotting the simulation

To plot the results click on the tab 'Plots' and select the variable from the section you want to plot. Then click *Plot*.

If you want to plot more variables in one go hold *Ctrl* and select as many as you want, then click *Plot*

If you want to insert the legend just select the *legend box*



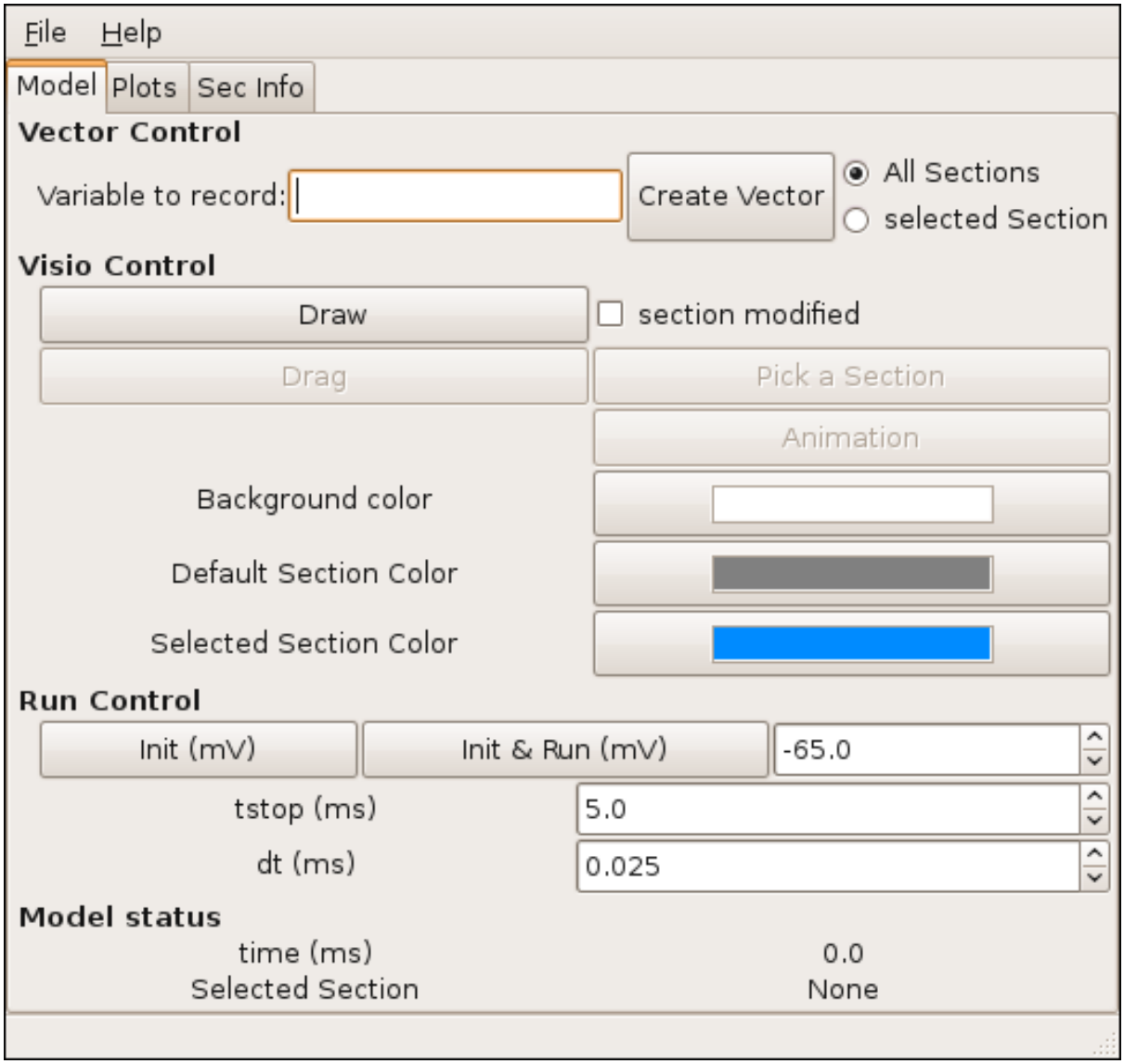
5.3 Screenshots

Everybody loves screenshots so here we go.

5.3.1 Gtk GUI control

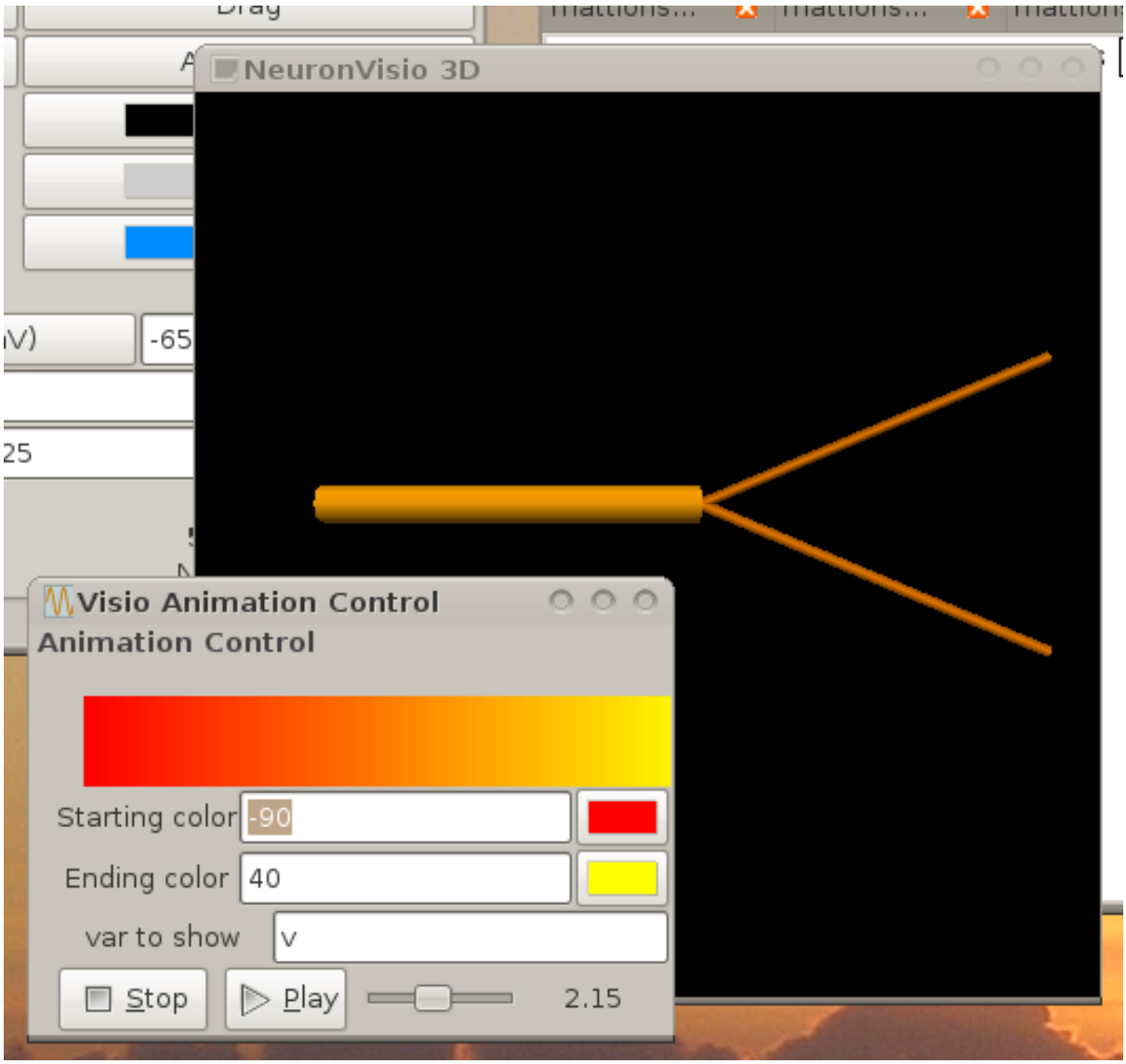
This is the main GTK window to cotrol Neuronvisio. You can create vectors and run the simulation. The time shows you the time of the NEURON simulator.

You can change the *tstop*, the *dt* and the initial voltage from the GUI or the console.

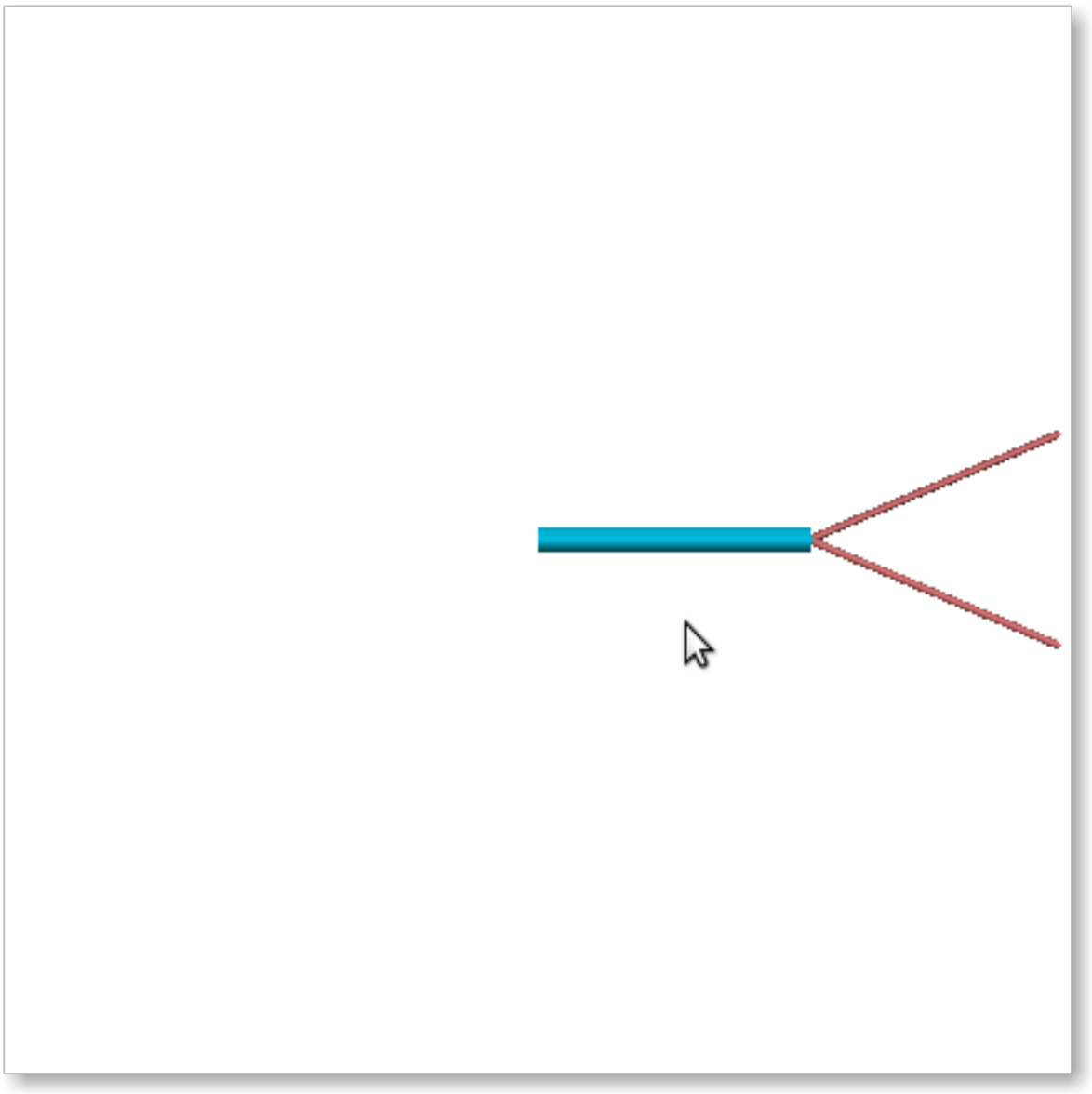


5.3.2 3-Dimensions with a simple model

Rendering of a simple model with 3 section.

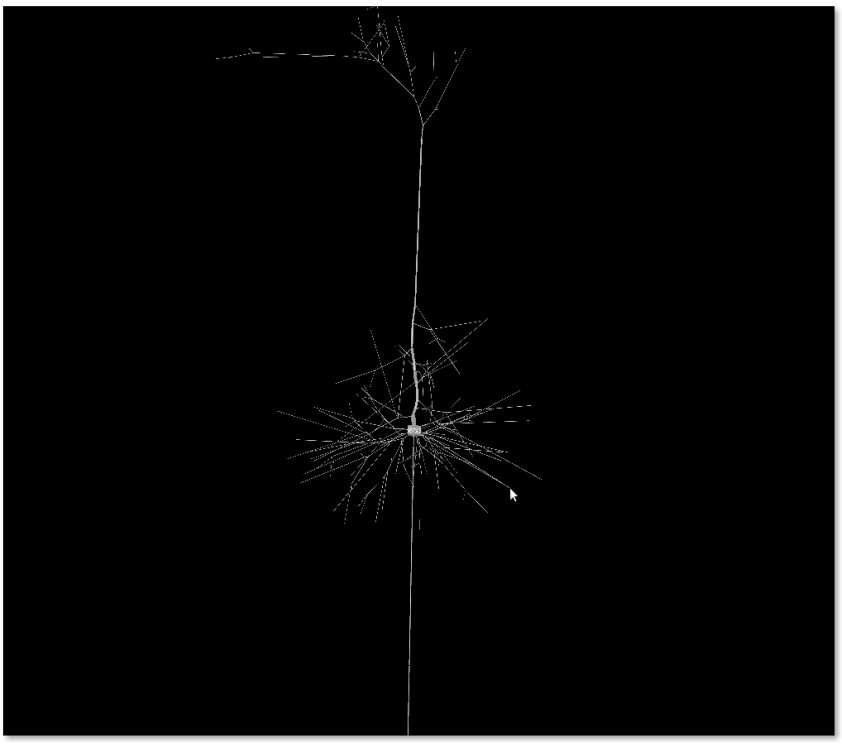


The colors can be changed by the user.



5.3.3 3-Dimensions with a complex model

Rendering of a more complex model, a pyramidal neuron.

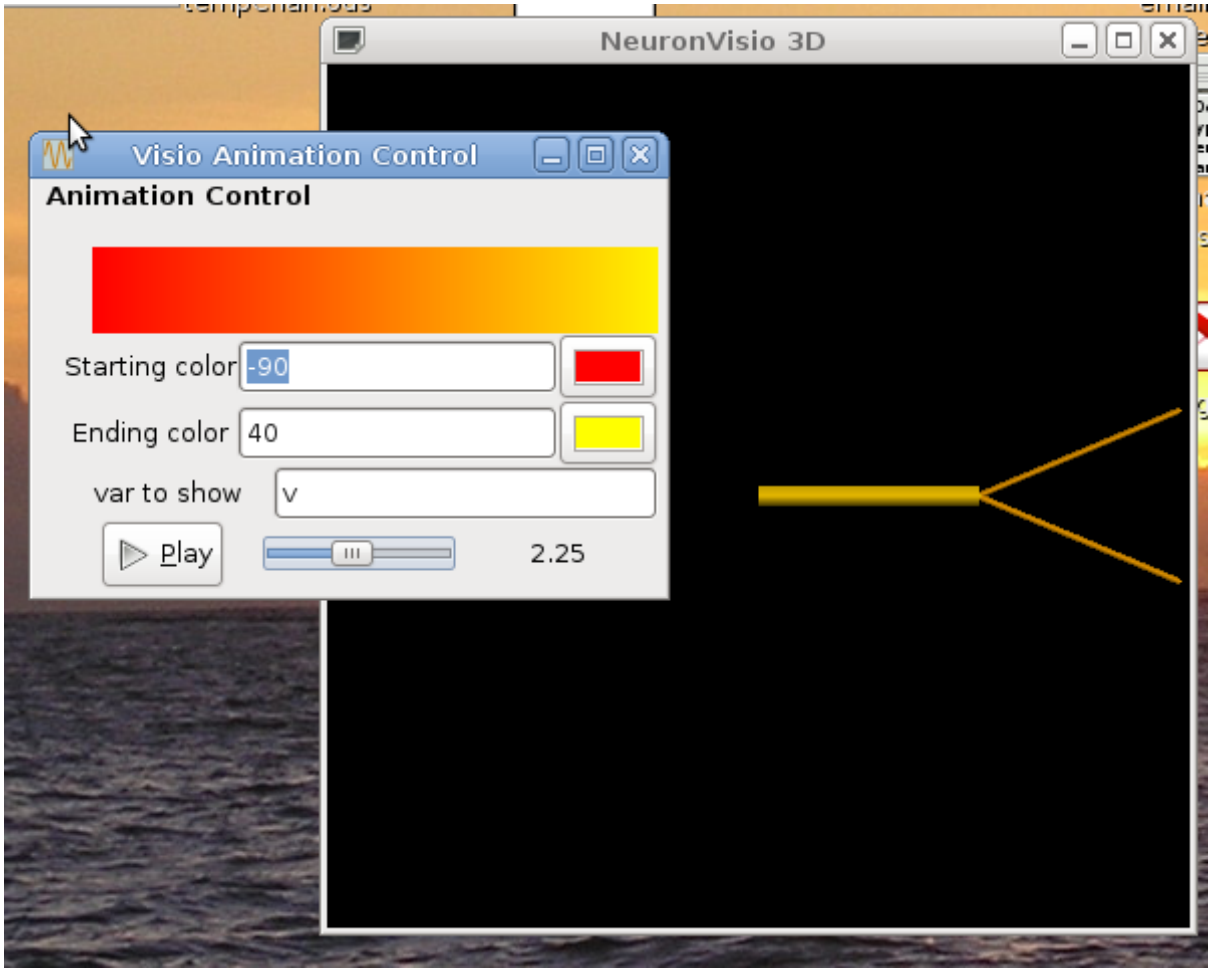


5.3.4 Animation window and pylab graph

The animation window and the pylab graph, showing the variation of the voltage in the soma and the behaviour of the same variable through the cell.

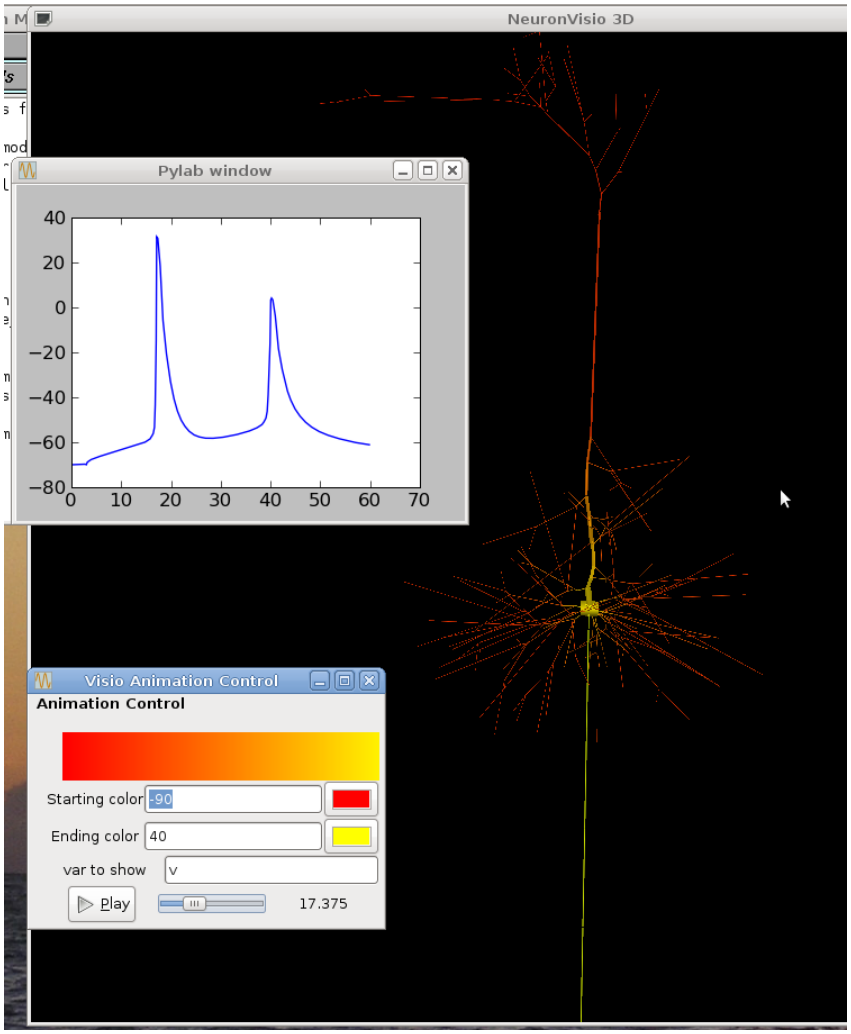
Simple model

A simple 3 sections model showing the different value of the voltage in the cell.



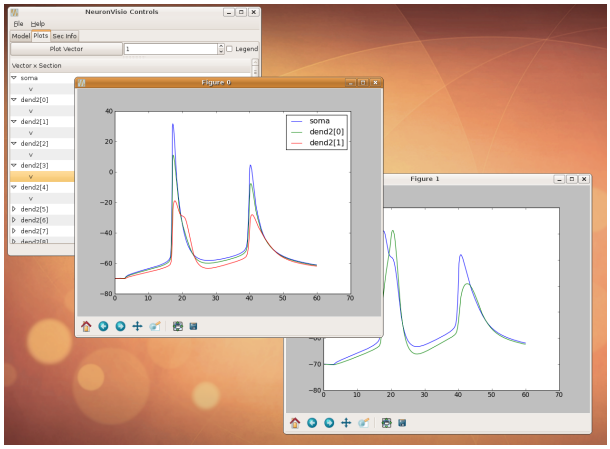
Pyramidal neuron

The propagation of the voltage among the neuron. The stimul was given in the soma.



Pylab integration

It is possible to use the standard pylab tool and to decide in which figure to plot the curve.



5.4 Reference

The API directly from the docstrings in the code.

5.4.1 `neuronvisio.manager` – Manage the map between vectors and sections

synopsis Manage the map between vectors and sections

`Manager`

class `Manager`()
The Manager class is used to manage all the vecRef, to create them and retrieve the information

`add_all_vecRef`(*var*)
Create the vector for all the section present in the model with the given variable :param var: The variable to record

`add_synVecRef`(*synapse*)
Add the synVecRef object to the list

Parameter *synapse* – The synapse to record.

`add_vecRef`(*var, sec*)
Add the vecRef to the vec_res list. It takes care to create the vector and record the given variable.

Parameters • *var* – The variable to record

• *sec* – The section where to record

Returns True if the vector is created successfully.

`convert_syn_vec_refs`()
Convert the synVecRef into pickable changing the hocVector with a numpy array

`convert_vec_refs`()
Convert all the vecRefs into the pickable substistitute the hocVectors with a numpy array Set to None the ref for the section.

`get_tree`(*sec*)
Return the minimal tree of section Using the given section as the last leaf

Parameter *sec* – The section that will be used as the last leaf

Returns The section's tree in a list format

`get_vector`(*sec, var*)
Return the vec that record the var in a given section

Parameters • *sec* – Section of interest

• *var* – variable recorded by the vector.

Returns the vector that record the variable var

`get_vectors`(*section_list, var*)
Return a dictionary containing the vector which record the var. The section name is used as key.

Parameters • *section_list* – The list of the section which is interested

• *var* – The variable of interest

Returns The dictionary with section name as key and the vector as the value

Return type dictionary

`plotVecs`(*vecs_dic, legend=True, figure_num=None*)
Plot the vectors with plt

Parameters • *vecs_dic* – dictionary with section name as k and the vec obj as value

• *var* – Which variable we are plotting.

• *legend* – If True the legend is plotted

- *figure_num* – in which figure we want to plot the line

sum_vector(*vec1, vec2*)
Sums two vectors with the same length. The vector are converted in numpy array and then summed together

Parameters
- *vec1* – First addendum
- *vec2* – Second addendum

Returns The numpy array sum of the two.

Return type Numpy array

VecRef

class VecRef(*sec*)
Basic class to associate one or more vectors with a section

Create a vecRef object which map the section name and the recorded vectors.

Parameter *sec* – The section which all the vectors belongs

SynVecRef

class SynVecRef(*syn*)
Class to track all the synapse quantity of interest

Create a synVecRef object which map the synapse positiona and name and the recorded vectors in it.

Parameter *syn* – The synapse to map

5.4.2 neuronvisio.visio – 3D Visual operations

synopsis 3D Visual operations

Contain all the 3D operations.

Visio

class Visio()

calc_offset(*start_v, end_v, v*)
Calculate the offset for the cairo gradient according to the input variable

calculate_gradient(*var_value, start_value, start_col, end_value, end_col*)
Calculate the color in a gradient given the start and the end

params: var_value - The value read from the vector start_value - the initial value for the var end_value - the final value for the var start_col - the starting color for the linear gradient end_col - the final color for the linear gradient

drag_model()
Drag the model

draw_model(*controls*)
Draw the model. Params: controls - the main gui obj.

draw_section(*sec, color*)
Draw the section with the optional color and add it to the dictionary cyl2sec

Parameters
- *sec* – Section to draw

- *color* – tuple for the color in RGB value. i.e.: (0,0,1) blue

findSecs(*secList, secName*)
Find a section with a given Name in a List of Section

pickSection()
Pick a section of the model

retrieve_coordinate(*sec*)
Retrieve the coordinates of the section

show_variable_timecourse(*var, time_point, start_value, start_col, end_value, end_col, vecRefs*)
Show an animation of all the section that have the recorded variable among time

Parameter *var* – the variable to show

5.4.3 neuronvisio.controls – Gtk UI module

synopsis Gtk UI module

GTK2 class and helpers' thread

Controls

class Controls()
Main GTK control window. create a control object and start with controls.start()

expose_gradient(*widget, event*)
Redraw the gradient everytime is shown. The colors value are taken by the tow gtkbuttoncolors

get_info(*section*)
Get the info of the given section

on_about_activate(*widget, data=None*)
About dialogue pop up

on_animation_clicked(*widget*)
Show the animation control

on_animation_control_delete_event(*widget, event*)
Hide the animation control instead of destroying

on_background_button_color_set(*widget*)
set the background color in the visio window

on_createVector_clicked(*widget, data=None*)
Create the vectors list

on_drag_clicked(*btn, data=None*)
To drag the model in the window

on_draw_clicked(*widget, data=None*)
Draw the whole model

on_dt_spin_value_changed(*widget*)
Update the dt value in the simulator

on_end_color_set(*widget*)
Set the end color when changed

on_init_clicked(*widget*)
Set the vm_init from the spin button and prepare the simulator

on_pick_clicked(*widget, data=None*)
Select a section from the 3D visio

on_play_clicked(*widget*)
Play the animation with the voltage color coded

on_plot_clicked(*widget, data=None*)
Create a plot of the selected vector

on_quit_activate(*widget, data=None*)
Destroy the window

on_run_sim_clicked(*widget*)
Run the simulator till tstop

on_section_button_color_set(*widget*)
Set the default color for the section

on_selected_section_button_color_set(*widget*)
Set the default color for the selected section

on_start_color_set(*widget*)
Set the start color when changed

on_stop_clicked(*widget*)
Stop the animation

on_timeline_value_changed(*widget*)
Draw the animation according to the value of the timeline

on_tstop_spin_value_changed(*widget*)
Update the tstop value in the simulator

on_voltage_spin_value_changed(*widget*)
Update the voltage value in the simulator

read_only(*storage*)
Function used to inspect the results of a simulation

run()
Running the gtk loop in our thread

set_colors()
Set the colors in the visio module

update()
Update the GUI spinbuttons only if the user is not using them with the value from the console.

update_timeline(*t_indx, time*)
update the timeline

update_visio_buttons()
Update the ui buttons connected with visio

TimelineHelper

class TimelineHelper(*controls, var, start_value, start_color, end_value, end_color, vecRefs*)
Thread to update the timeline when the play button is clicked

TimeLoop

class TimeLoop(*controls*)
Daemon Thread to connect the console with the GUI

run()
Update the GUI interface calling the update method

5.5 Changes in Neuronvisio

5.5.1 0.3.5+ dev

5.5.2 0.3.5 - 20 Nov 2009

- Using sphinx for the doc
- Using paver for deployment
- python egg and easy install support
- User manuel available in pdf format

5.5.3 0.3.4 - 15 Sep 2009

- Changed the way the module is imported to allow other program to use the manager as a storing objects for results.

5.5.4 0.3.3 - 3 Sep 2009

- Integrated the pylab interface using the GTK backend provided by pylab. It is possible to zoom and navigate the graph with the pylab tools.
- It is now possible to decide in which figure to plot, using the current figure selector.

5.5.5 0.3.22 - 31 Jul 2009

- Closed bug #10
- Changed the name of the module from nrnVisio to nrnvisio to be python standard compliant.
- Manager being transformed into a library (WIP)

5.5.6 0.3.21 - 20 Jul 2009

- Better handling of the pick section routine
- Changed the examples to use the create statement for hoc, to have a proper name of the section also in python.
- Modified the GUI to handle a runtime change of a section. The model is redrawn completely, the zoom is conserved.

5.5.7 0.3.2 - 20 Jul 2009

Bug Release. Closed Bug #9

5.5.8 0.3.1 - 18 Jul 2009

Bug Release.

5.5.9 0.3.0 - 14 Jul 2009

New Features

- Stop Button on the animation Control
- Better handling on the timeline updating routine.

BUGFixes

- Closed bug #8
- Closed bug #3

5.5.10 0.2.0 - 6 Jul 2009

New Features

Some new features has been introduced:

- User defined color. The user can now change the colors of the model for a better contrast.
- Info tab. Reports the properties of the selected section.

BUGFixes

- Closed bug #4
- Closed bug #5
- Closed bug #6

5.5.11 0.1.0 - 30 Jun 2009

Fist public release.

Features

- 3D visualization of the model with the possibility to change it runtime
- Creation of vectors to record any variable present in the section
- Pylab integration to plot directly the result of the simulation
- Explore of the timecourse of any variable among time using a color coded scale in the 3d representation
- the GUI runs in its own thread so it's possible to use the console to modify/interact with the model.

CHAPTER

SIX

INDICES AND TABLES

- *Index*
- *Module Index*
- *Search Page*