Neuronvisio Documentation

Release 0.9.1

Michele Mattioni

January 19, 2017

Contents

CHAPTER 1

Install

1.1 Requirements

To install Neuronvisio you need to satisfy the following dependencies

- PyQt4: http://www.riverbankcomputing.co.uk/software/pyqt/download
- matplotlib: http://matplotlib.sourceforge.net/
- setuptools: http://pypi.python.org/pypi/setuptools
- pytables: http://www.pytables.org/
- ipython: http://ipython.org 0.12 (can be automatically installed, see *Package Install*)
- mayavi2: http://code.enthought.com/projects/mayavi/ 4.2.0 or better (can be automatically installed, see *Package Install*)

and of course NEURON>=7.2 compiled with python support

1.1.1 Easy Way: Use conda

You can install all the dependencies and everything you need (also NEURON) to run Neuronvisio with one command once you have conda installed. If you do not have, get it from here (http://conda.pydata.org/miniconda.html)

```
# Create a new environment and install neuronvisio there
conda create -n neuronvisio_env neuronvisio -c mattions
```

After the installation activate the environment and you good to go:

```
source activate neuronvisio_env
neuronvisio
```

If the HOC file does not get found, export NEURONHOME variable like

```
export NEURONHOME=$CONDA_ENV_PATH/share/nrn/
neuronvisio
```

The packages are made only for linux-x64, but we are looking into expanding them to all the supported platforms.

Otherwise you could install by hand following the instructions below.

1.1.2 Ubuntu and friends

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

```
sudo apt-get install python-qt4 python-matplotlib python-setuptools python-tables \
                     mayavi2
```

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

Next, see the instructions on installation of NEURON with Python available at http://andrewdavison.info/notes/installation-neuron-python/

Proceed to the *Package Install* .

1.1.3 Mac OS X

Install the Homebrew package manager:

```
ruby -e "$(curl -fsSL https://raw.github.com/Homebrew/homebrew/go/install)"
```

Ensure your homebrew installation is fully updated:

```
brew update
```

and "raring to brew" (configured to Homebrew's liking) by following the instructions given by:

```
brew doctor
```

For example, it is critical to ensure sure that you don't have another python distribution on your path (eg. Enthought), and it is also recommended that you place "/usr/local/bin" before "/usr/bin" in your PATH variable set in your "$HOME/.bash_profile" settings file.

Install all the homebrew packages you need, including the Homebrew version of python. NB: It is recommended by a few internet sources (eg. https://python-guide.readthedocs.org/en/latest/starting/install/osx/) to install python as a framework (the '–framework' option below) but it is not strictly necessary for the installation of Neuronvisio:

```
brew install python --framework
brew install qt
brew install homebrew/science/hdf5
```

At the time of writing, Mayavi (installed by the NeuronVisio package installer) doesn't work with the latest version of VTK (6.0.0) so you will need to install version 5.10.1 instead. Also, VTK has been moved to the homebrew-science "tap" so you will need to tap that first before installing it, i.e.:

```
brew tap homebrew/science
cd /usr/local/Library/Taps/homebrew-science
git checkout 27a4327 vtk.rb
brew install vtk --python --qt --pyqt
```

Install python packages using pip, ensuring that you are using the Homebrew version of pip, /usr/local/bin/pip, which you should be if you put /usr/local/bin before /usr/bin in your PATH variable (this can be checked using the "which pip" command). You may also run into problems if you have setuptools in your system python ('/Library/Python/2.7/site-packages'), in this case temporarily move the system setuptools to somewhere else for the rest of the installation procedure:

```
pip install numpy
pip install matplotlib
pip install numexpr
pip install cython
pip install ipython
```

After installing ipython you will probably want to put a link to it somewhere on your path, eg. /usr/local/bin:

```
ln -s /usr/local/share/ipython /usr/local/bin/ipython
```

Next, see the instructions on installation of NEURON with Python available at http://www.davison.webfactional.com/notes/installation-neuron-python/ (again ensuring you are using the Homebrew version of python '/usr/local/bin/python')

Try the *Package Install* but if that fails use the Source Code instructions (I needed to do this at least)

1.1.4 Windows

Under Windows it is recommended to get a prepackaged scientific python distribution which contains most of neuronvisio's dependencies, such as:

- Enthought Distribution: http://www.enthought.com/products/epd.php (free for an academic use)
- Python(x,y): http://www.pythonxy.com/foreword.php (free)

Alternatively, if you want to do it yourself, you will need to manually install and configure the dependencies listed in *Requirements*

A portable and already compiled version of NEURON for Python is available from https://bitbucket.org/uric/pyneuron/ or simply by using:

```
pip install PyNEURON
```

Proceed to the *Package Install* .

1.2 Package Install

To install Neuronvisio we suggest to create a virtualenv and install the packages there. Check out virtualenv and virtualenvwrapper

If you have pip installed and all the requirements are already met you can install neuronvisio from PyPi typing:

```
pip install -U neuronvisio
```

Note: Mayavi stack and ipython will be installed automatically as Neuronvisio requirements from PyPi.

1.2.1 Running the bleeding edge

If you want to run the latest code, directly from the repo, you can do it using pip:

```
pip install -e git+https://github.com/mattions/neuronvisio.git#egg=neuronvisio
```

then you need to add the directory (the absolute path) to your PYTHONPATH (in bash):

```
export PYTHONPATH=$PYTHONPATH:/path-to-neuronvisio-dir
```

1.3 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
```

CHAPTER 2

Getting Started

2.1 How does it work

To run Neuronvisio you can either use the provided *neuronvisio* executable, which will create an ipython session:

```
$ neuronvisio
```

Note: On windows, you can launch it with *neuronvisio.bat*

or you can start ipython and load neuronvisio from within your script, as explained in section *How to integrate Neuronvisio with your code*

To load Neuronvisio just paste this two lines in ipython:

```
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.

2.1.1 Loading a file

You can also load 3 different formats right now:

1. A NeuroML file (.xml)
2. A NEURON hoc file hoc file (.hoc)
3. A HDF file formatted according to Neuronvisio format, (.h5). Check *storage*.

to load any of them just pass it as an argument:

```
$ neuronvisio path/to/my/file.hoc (or .h5, or .xml)
```

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.

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

2.2.2 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.

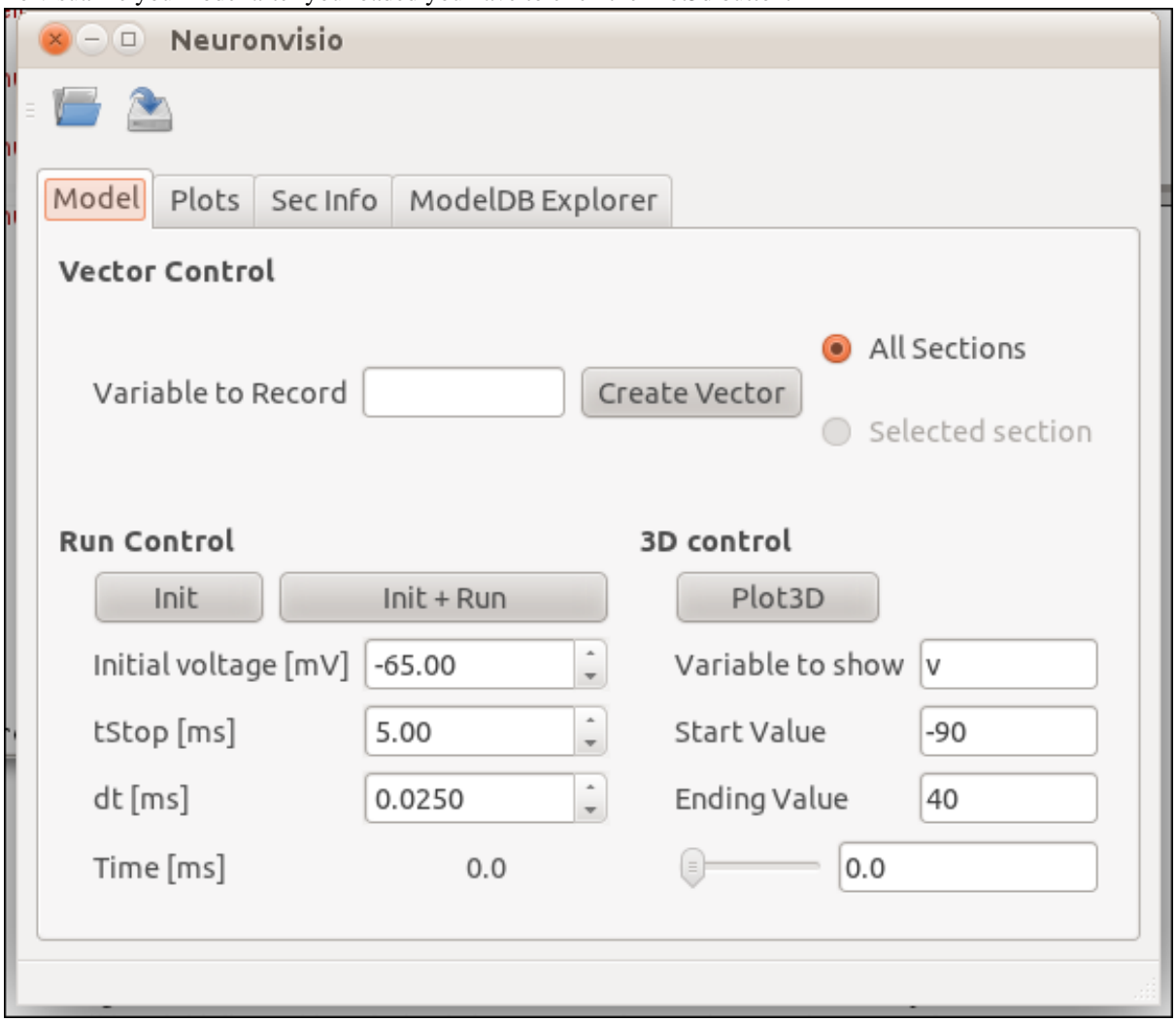
A classical template is:

```
from neuronvisio.controls import Controls
from neuron import h
controls = Controls()    # starting the GUI
h.load_file('path/to/my_model.hoc')
```

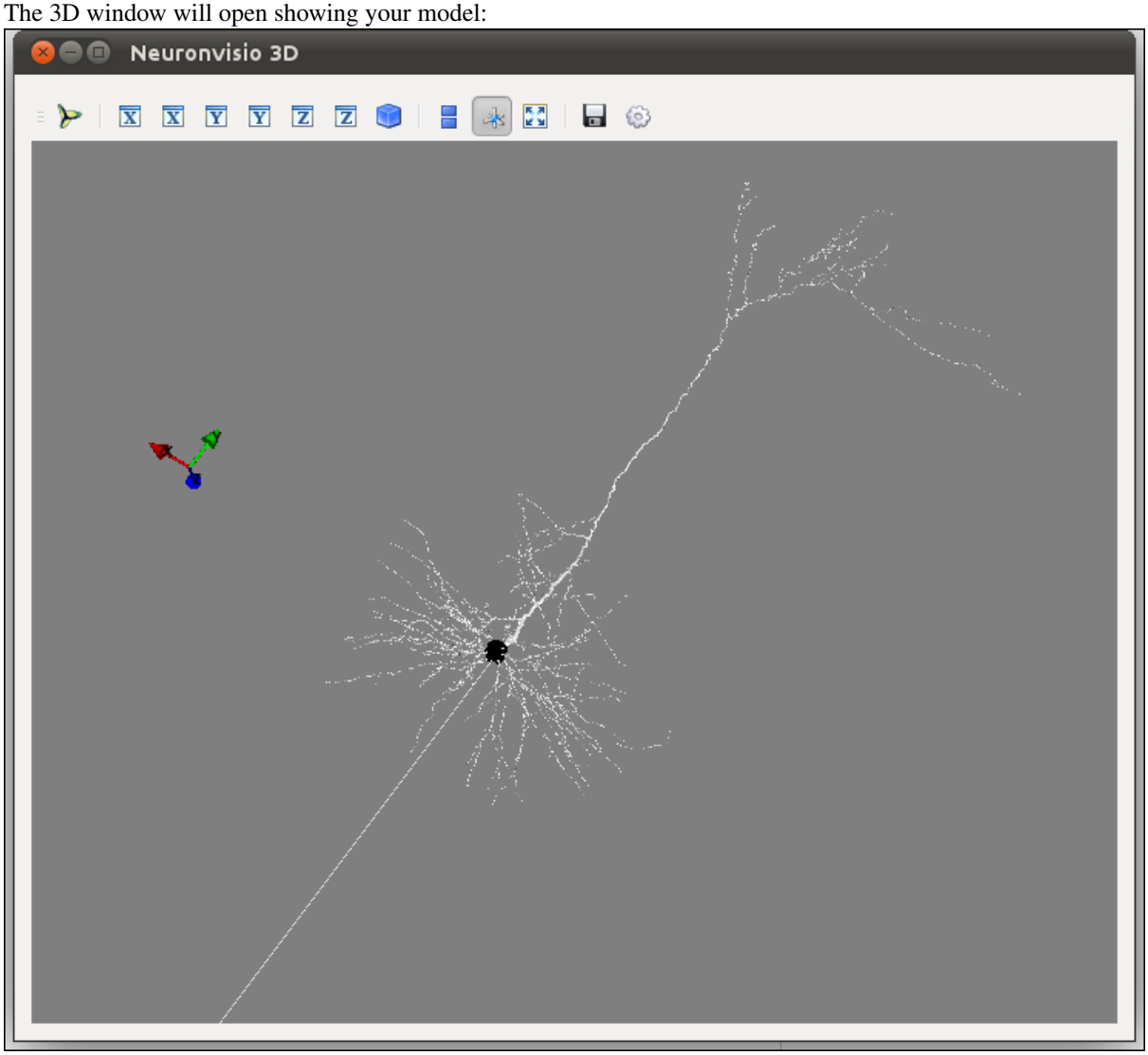
2.3 Neuronvisio features

2.3.1 Visualization

To visualize you model after you loaded you have to click the Plot3d button.



The 3D window will open showing your model:



2.3.2 How to rotate

Hold the left button and move the mouse.

2.3.3 How to zoom

Use your mouse's wheel or the right button of the mouse.

2.3.4 How to move

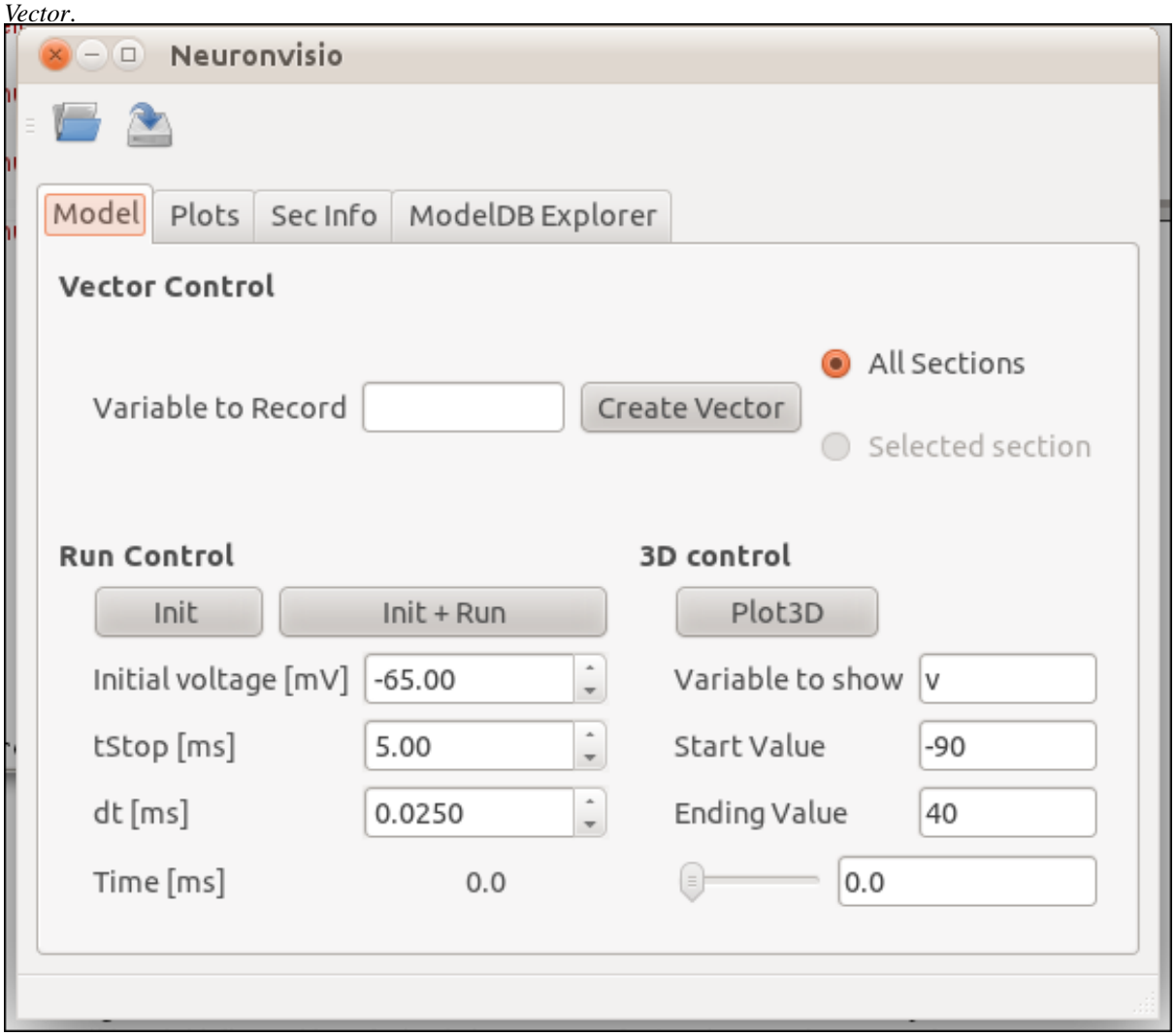
Hold the wheel down and move the mouse.

2.4 Plotting the simulation results

2.4.1 Creating the vectors

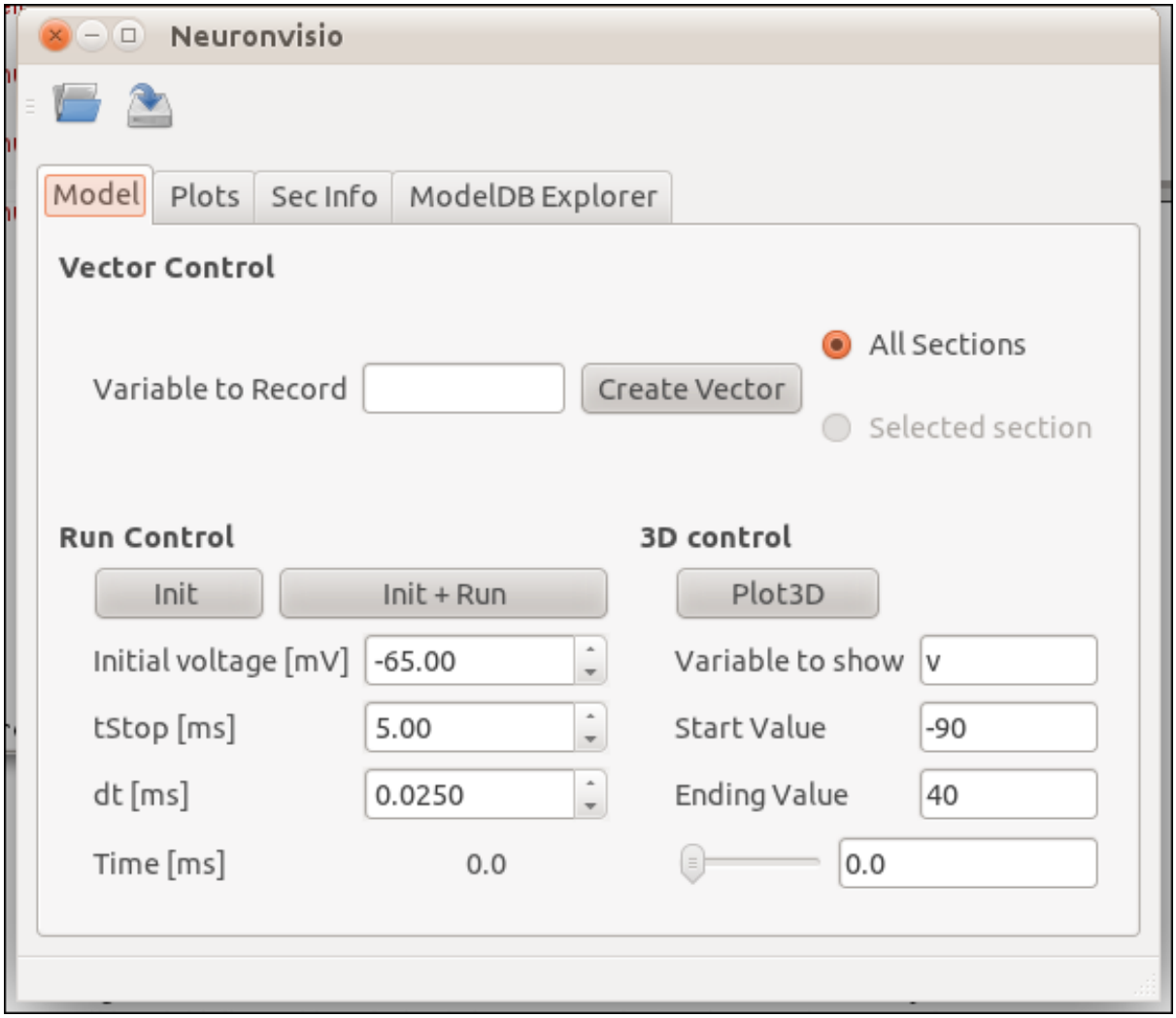
To plot the simulation's results you first have to create a Vector (or more than one) to record the variable that you are interested in.

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



2.4.2 Run the simulation

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

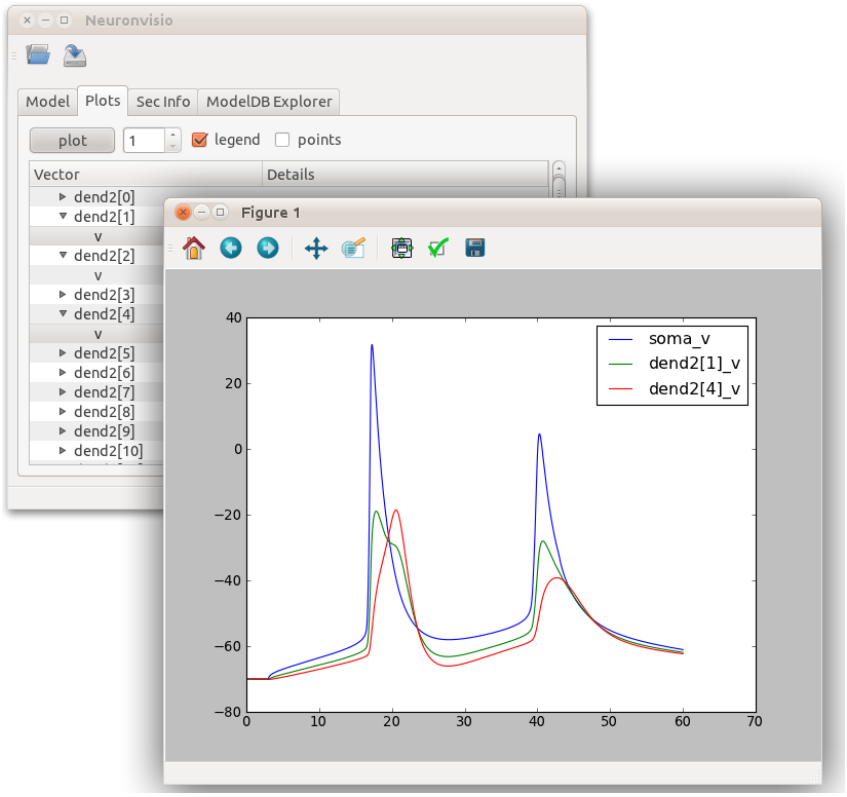


2.4.3 Plotting the simulation

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

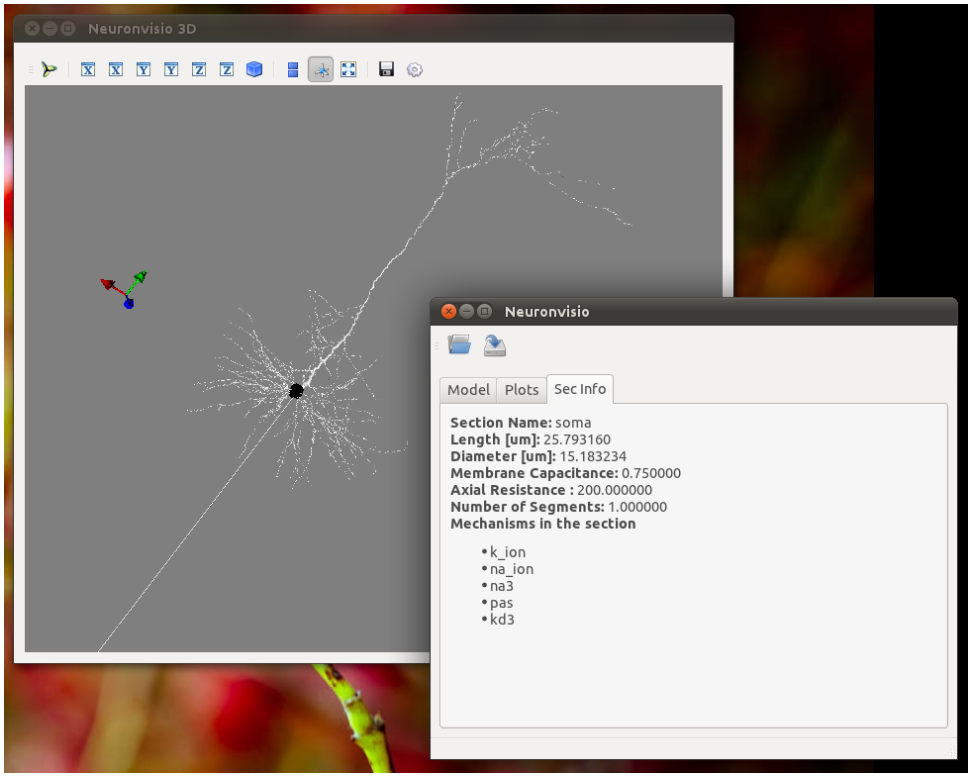
If you want to plot more variables in one go hold *Ctrl* and select as many as you want, then click *Plot*

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



2.5 Investigate the section parameters

Select a section (Just click over it) and the section info will be displayed in the Sec Info Tab.



2.6 ModelDB Integration

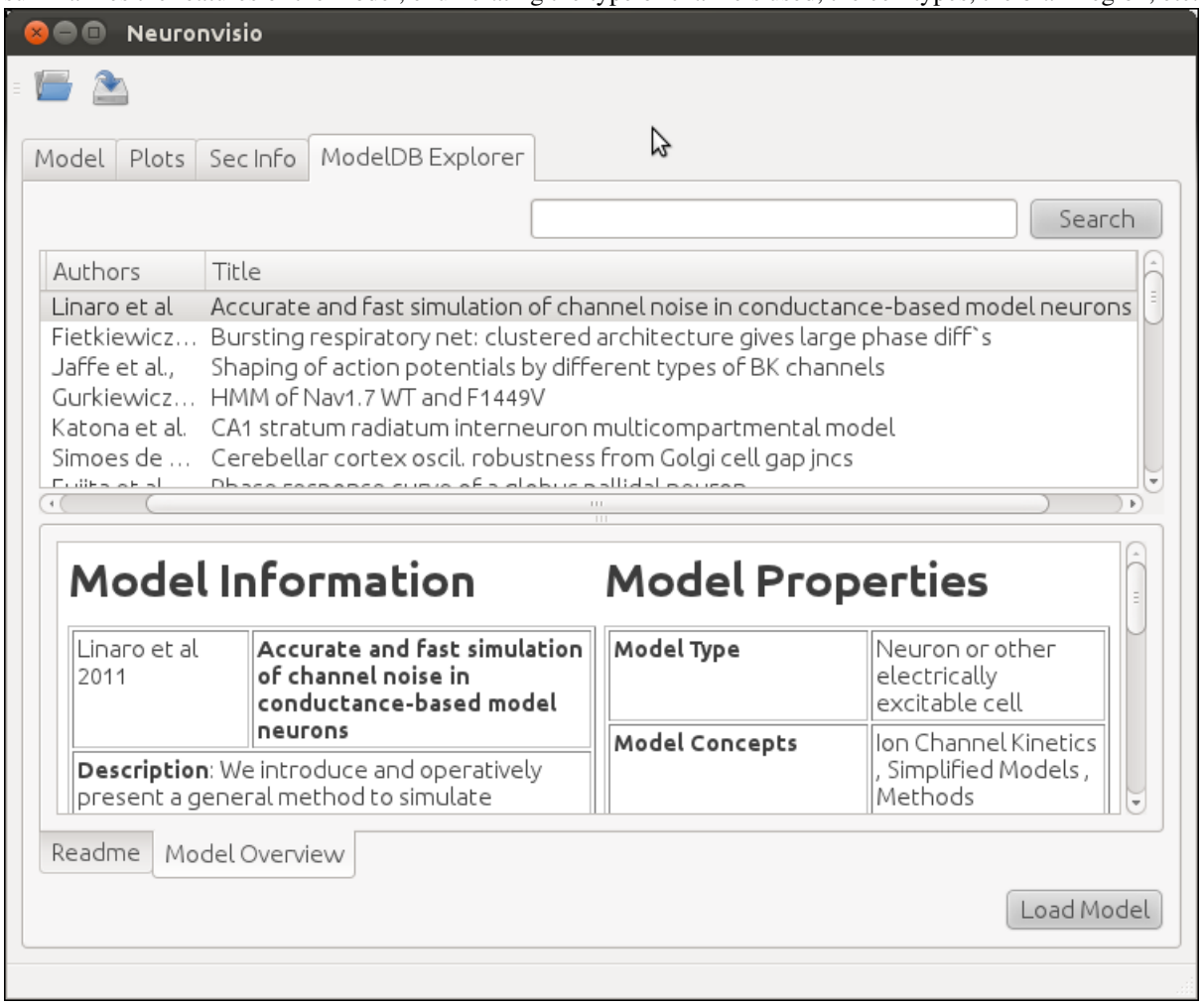
This section describe how to load a model from ModelDb in Neuronvisio, and how to get the latest model from the ModelDB, if they are not already present in the ModelDB XML list.

2.6.1 Browsing the NEURON models from ModelDB

ModelDB database is a lightly curated repository of computational models, published in litterature http://senselab.med.yale.edu/ModelDB/. While ModelDB accepts models in a variety of format, a large subset is formed by models stored in NEURON format. The ModelDB NEURON's model are stored in an XML file, which comes with Neuronvisio source code. The file is parsed at run time and the content is loaded in a Qt tree widget, available in the ModelDB explorer tab.

It is possible to browse among all the available models per year of publication, authors, title and unique id number. The columns can be ordered alphabetically, and it is possible to perform a simple search, writing the desired keyword in the search box.

If available, the README associated with the model is displayed, together with a custom model overview which summarizes the features of the model, enumerating the type of channels used, the cell types, the brain region, etc.



2.6.2 Loading a NEURON model from ModelDB

Any of the models available on the ModelDB explorer tab can downloaded and extracted. The models which have a *mosinit.hoc* will also be compiled and loaded in the current session, giving the user the possibility to explore and simulate the model.

If no *mosinit.hoc* is found, the software will open the folder where the model has been downloaded, warning the user of the impossibility to load it automatically.

Note: if the model does not have a *mosinit.hoc*, Neuronvisio cannot load it, because does not know what to load as master file. You can always load the relevant file following the section *Hoc Intergration* or *Python integration*.

2.6.3 Updating information from ModelDB

The content of the XML file which is included with each version of Neuronvisio is usually up-to-date with the content of ModelDB at the time of the release. Updating this file from the online DB can be done by manually, if required, by running from any shell the script *neuronvisio-modeldb-updater*. Neuronvisio updater will find where the current *ModelDB.xml* is stored, and update with the latest items, if any.

Note: You need to be able to write on the *ModelDB.xml* location. If you have installed the package as root, you may need to run it as root.

It should be noted that the model extraction from ModelDB is slowed down to 1/sec in order to avoid loading the site. Also this process only update the file with models which do not exist in the local XML file and does not currently refresh the content of existing ones.

2.7 Troubleshooting

If you start ipython with the *–pylab=qt* switch and get and error like:

```
    /usr/local/lib/python2.7/dist-packages/pyface/qt/__init__.py in prepare_pyqt4()
     15     # Set PySide compatible APIs.

     16     import sip
---> 17     sip.setapi('QString', 2)
     18     sip.setapi('QVariant', 2)
     19

    ValueError: API 'QString' has already been set to version 1
```

It means ipython has loaded the PyQt4 using the Version 1 of the API, which is default on Python 2.7 (Version 2 is default on Python 3.)

To solve this you can either define the variable *QT_API=pyqt*, e.g. in bash:

```
$ export QT_API=pyqt
$ ipython --pylab=qt
```

Or you can launch ipython without pylab support, load Neuronvisio and then load pylab with the magic *%pylab*.

2.7.1 Picking the right backend

If when you try to plot matplotlib segfault, you may have the wrong backend selected. Neuronvisio try to set the right backed, but if you start ipython with –pylab, automatically a bakcend is loaded and we can't change it due to matplotlib technical limitation.

To solve this, just customize your matplotlibrc downloading the matplotlibrc, saving the file in *.matplotlib/matplotlibrc* and changing the backend to *Qt4Agg*, from:

```
backend      : GTKAgg
```

to:

```
backend      : Qt4Agg
```

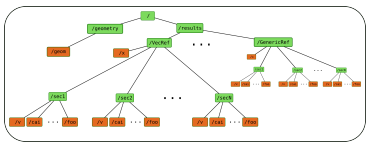
CHAPTER 3

Saving and loading simulations' reults

3.1 HDF structure

Neuronvisio stores simulation's results using the hdf standard, using PyTables This is very handy when you simulation takes a long time to run and you want to inspect again the results, without re-run it.

The file has a structure shown in the following _static



The Refs is the data structure used by Neuronvisio. The *VecRef* is the specialized one. It is possible to add more Ref subclassing the `manager.BaseRef`.

3.1.1 Using the manager object to store the results of your simulation

This is a quick example how to save the simulation in neuronvisio:

```
# Model geometry already instantiated.
#
from neuronvisio.manager import Manager
manager = Manager()
manager.add_all_vecRef('v') # Adding vector for the variable v

# Perform your simulation
# ...
#

# file where to save the results
filename = 'storage.h5'
# Saving the vectors
manager.save_to_hdf(filename)
```

If you run a lot of simulations you want maybe to run the same script but without rewriting the same results. Manager has a nice method to help you called create_new_dir:

```
saving_dir = manager.create_new_dir() # Create a new dir per Simulation, ordered by Day.
hdf_name = 'storage.h5'
filename = os.path.join(saving_dir, hdf_name)
# Saving the vectors
manager.save_to_hdf(filename)
```

3.1.2 Loading a previous simulation

To load the results of a simulation you can start neuronvisio giving the path_to_the_hdf_file:

```
$ neuronvisio path/to/storage.h5
```

or you can just start neuronvisio and use the Load button:

```
$ neuronvisio
```

3.2 Saving your variables in storage.h5 and use Neuronvisio to plot them

The *BaseRef* can be used to store computational results which are not in NEURON vectors format. Every BaseRef object is contained in a group, which is specified by the *group_id* attribute. The group_id is than used by Neuronvisio to pair the saved vectors with the time vectors which is required to plot.

To subclass the BaseRef just create a class:

```
from neuronvisio.manager import BaseRef

class MyRef(BaseRef):

    def __init__(self, sec_name=None, vecs=None, detail=None):

        BaseRef.__init__(self)
        self.group_id = 'MyGroup'
        self.sec_name = sec_name
        self.vecs = vecs
        self.detail = detail
```

Then you can create it:

```
myRef = MyRef(sec_name=sec_name,
              vecs=vecs,
              detail=detail)
```

sec_name should be the name of the section where the variable is been recorded, vecs is a dictionary with the variable name as key and the python_list with the computed value. A numpy array or an HocVector is also accepted.

After that you can add to the manager using the *manager.add_ref* which takes two arguments:

- the myRef object
- the x variable.

The x variable is the independent one, usually the time, which will be used to plot the from the Neuronvisio graphical interface. If don't need to supply your own time vector, because is the same of the main NEURON one, you can use the *manager.groups['t']* which will return the NEURON time array:

```
manager.add_ref(myRef, x)
```

All together is:

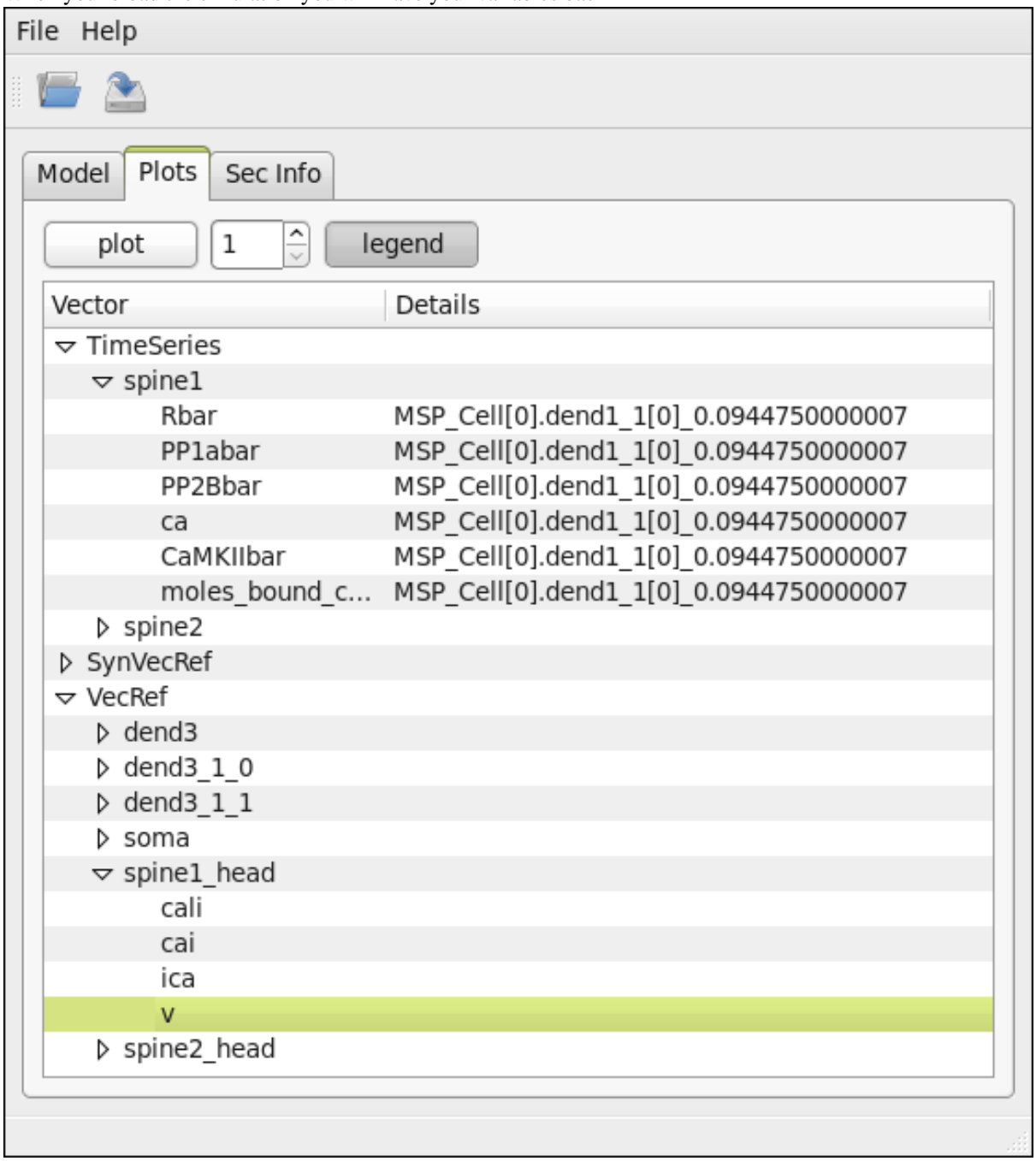
```
class MyRef(BaseRef):

    def __init__(self, sec_name=None, vecs=None, detail=None):

        BaseRef.__init__(self)
        self.sec_name = sec_name
        self.vecs = vecs
        self.detail = detail
```

```
myRef = MyRef(sec_name=sec_name,
          vecs=vecs,
          detail=detail)
manager.add_ref(myRef, x)
```

Then you just need to save the file where is more convenient for you:

```
filename = 'storage.h5'
# Saving the vectors
manager.save_to_hdf(filename)
```

When you reload the simulation you will have your variables back

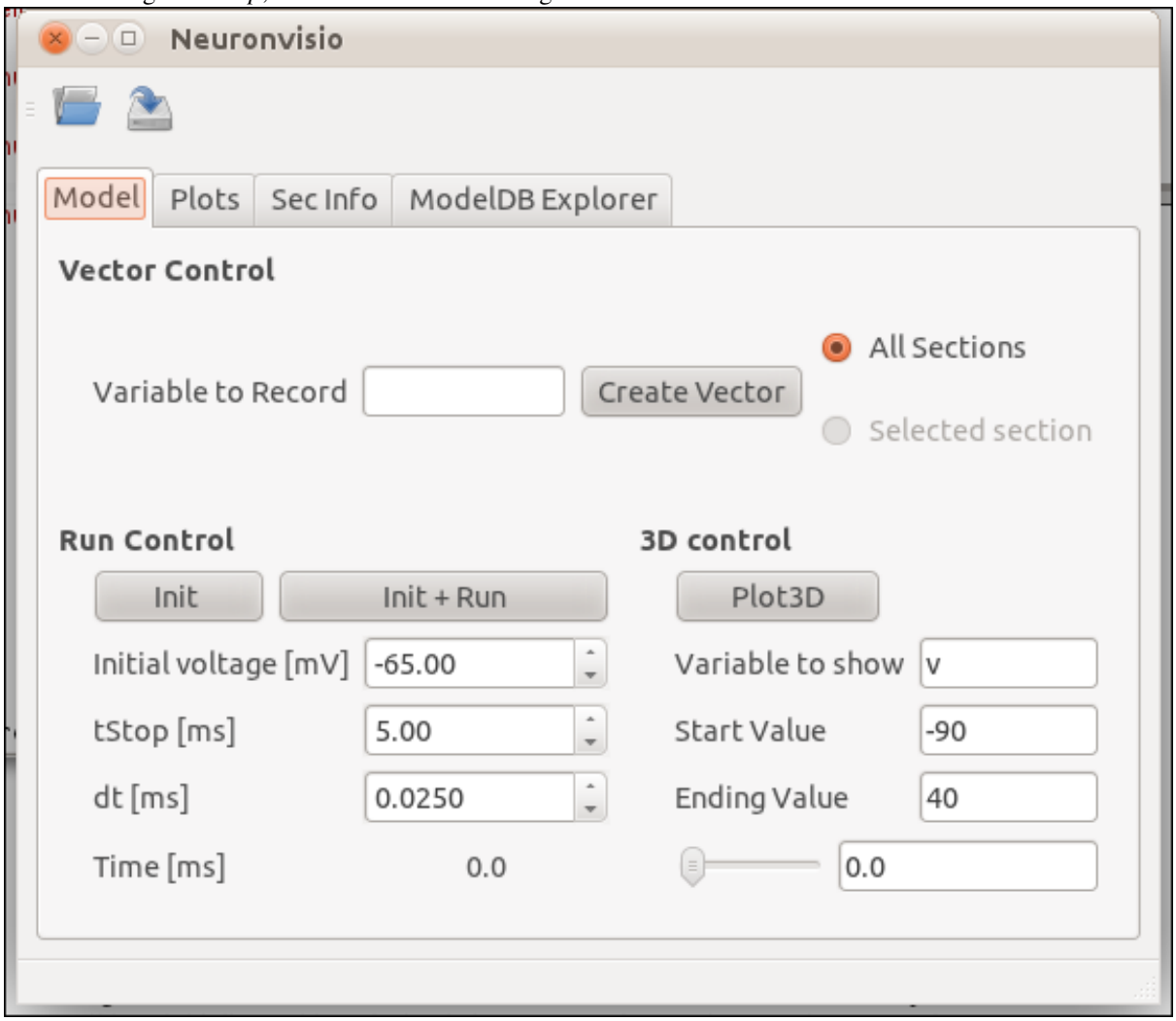


How to take screenshots and make movies

4.1 GUI control

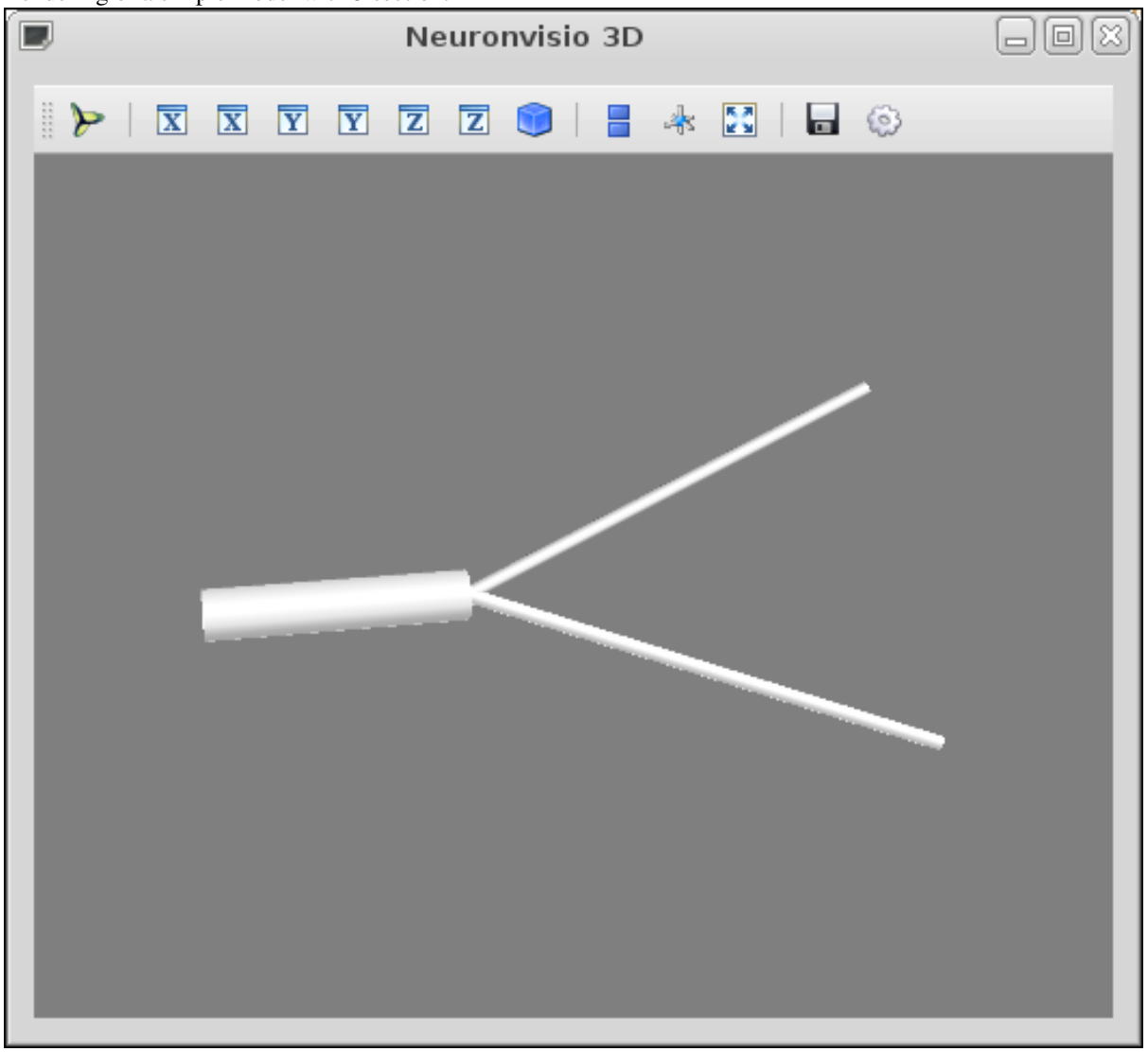
This is the Controls window. You can create vectors and run simulations. The time shows you the time of the NEURON simulator.

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

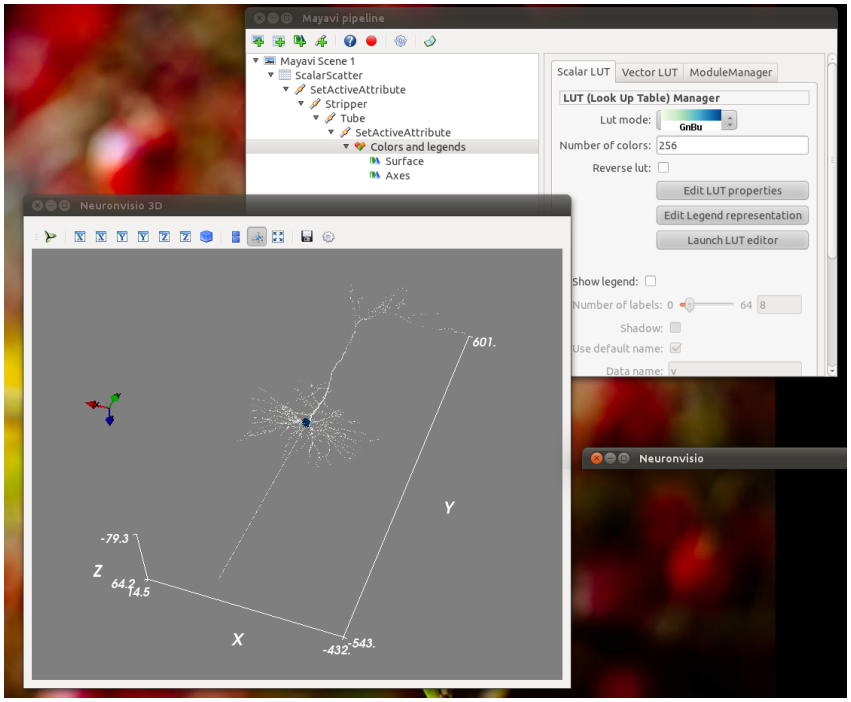


4.2 3-D rendering

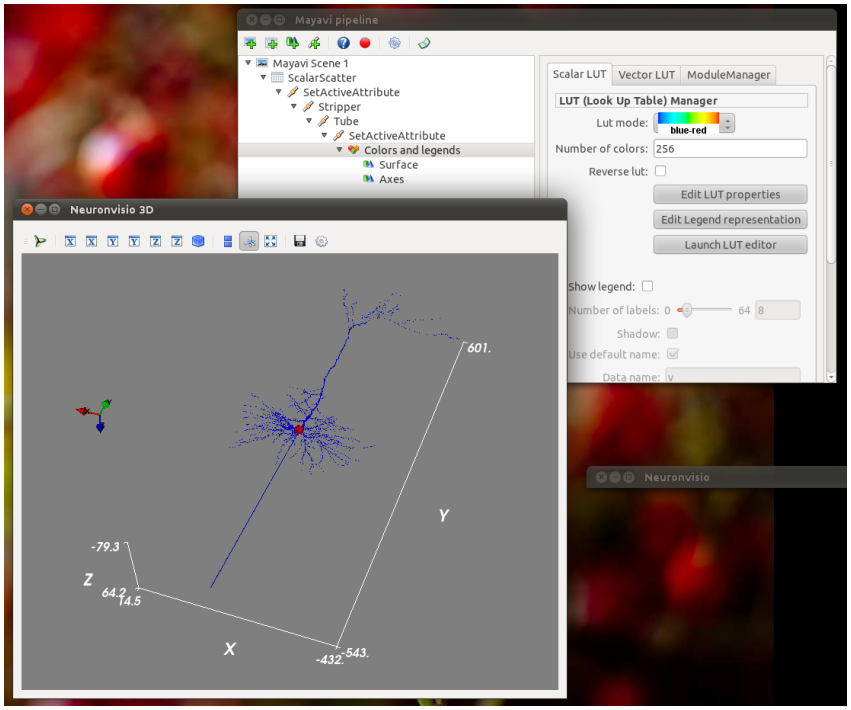
Rendering of a simple model with 3 section.



The neurons is rendered using the Mayavi pipeline, where is possibile to choose from different colormap and additional filters. Using the *GnBu* colormap with the Axes.

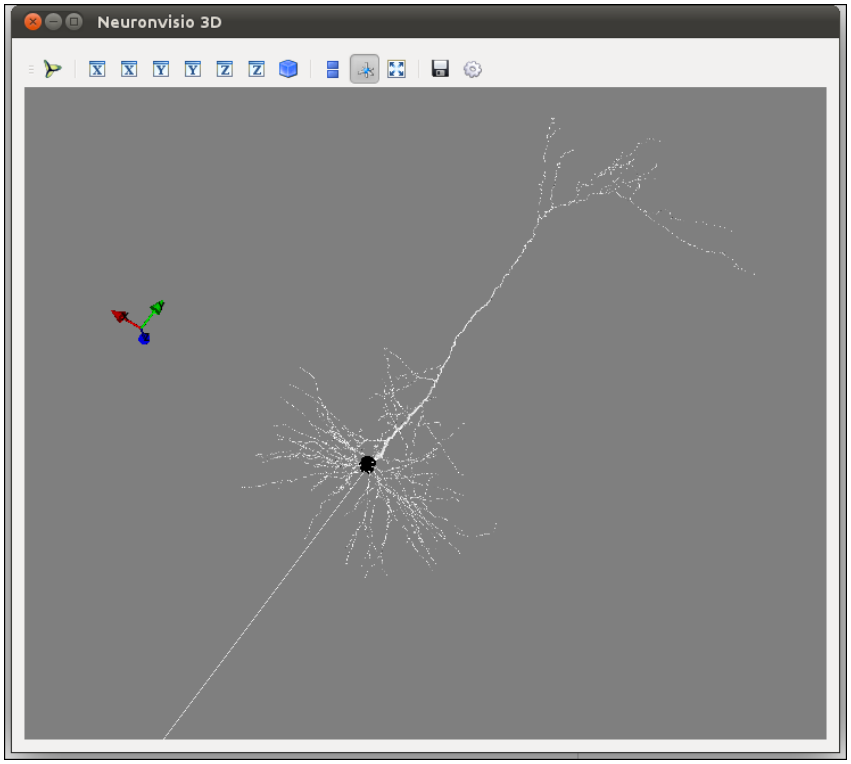


Using the *red-blue* colormap with the Axes.



4.2.1 3-rendering of a 3d-points reconstructed model

Rendering of a more complex model, a pyramidal neuron.



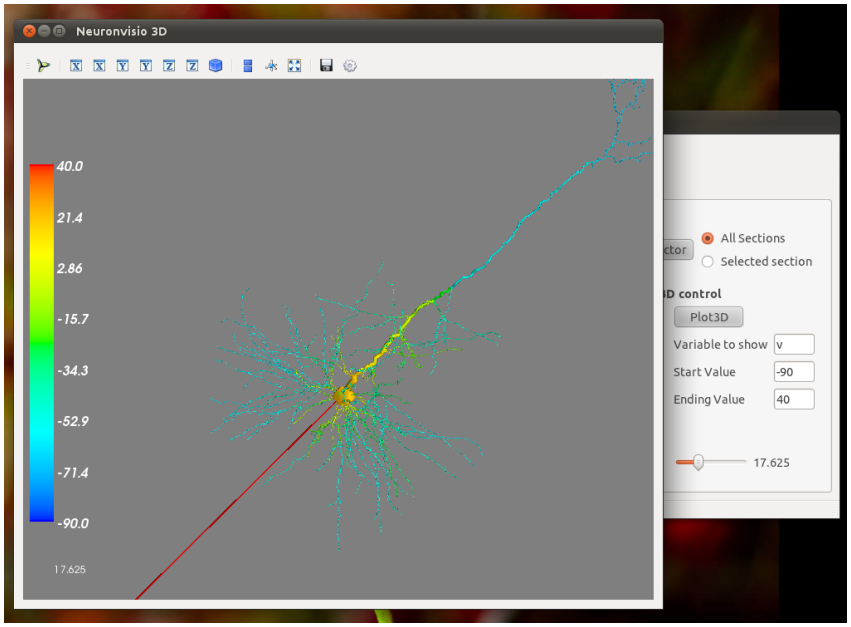
4.3 Timecourse movie and pylab graphs

It is possible to follow the timecourse of a variable in the whole neuron or networks using the bottom slider, after the simulation has been ran, or reloaded. If a particular point in time is of interest, insert in the animation Time line textfield and hit return.

The pylab graphs are integrated and can be easily generated with a click. For example here we shows hhow the voltage change in a simple model and in a pyramidal one.

4.3.1 Pyramidal neuron

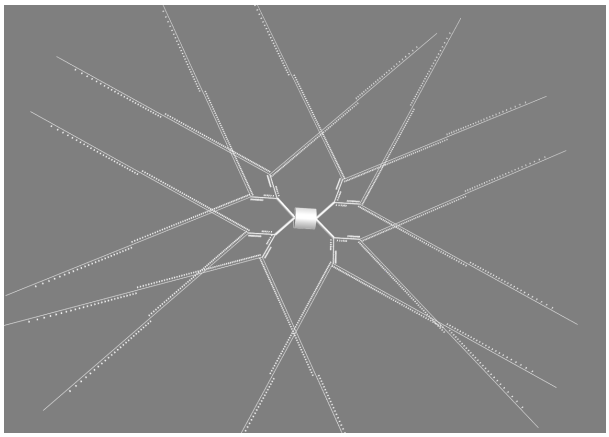
The propagation of the voltage among the neuron. The stimuli was given in the soma.



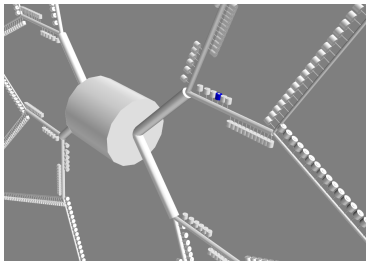
The pyramidal example is shipped with Neuronvisio.

4.3.2 Medium Spiny Neuron with explicit spines modelled

There are more than 4000 sections in this model, where each spine is individually modelled and distributed on the MSN body.

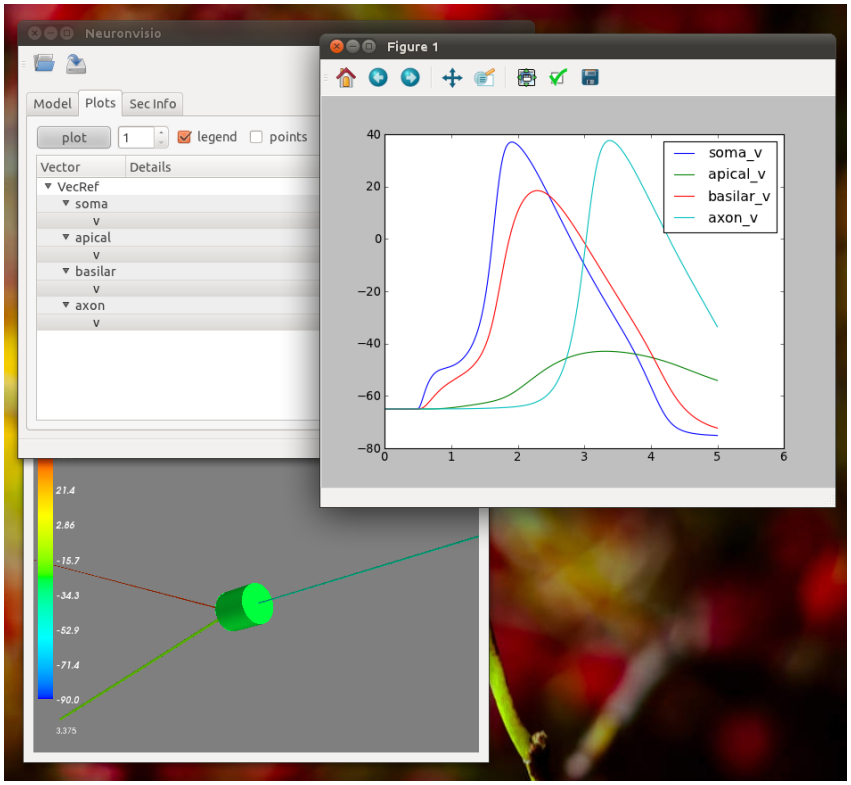


A zoomed version, with one spine's head selected.



4.3.3 Pylab integration

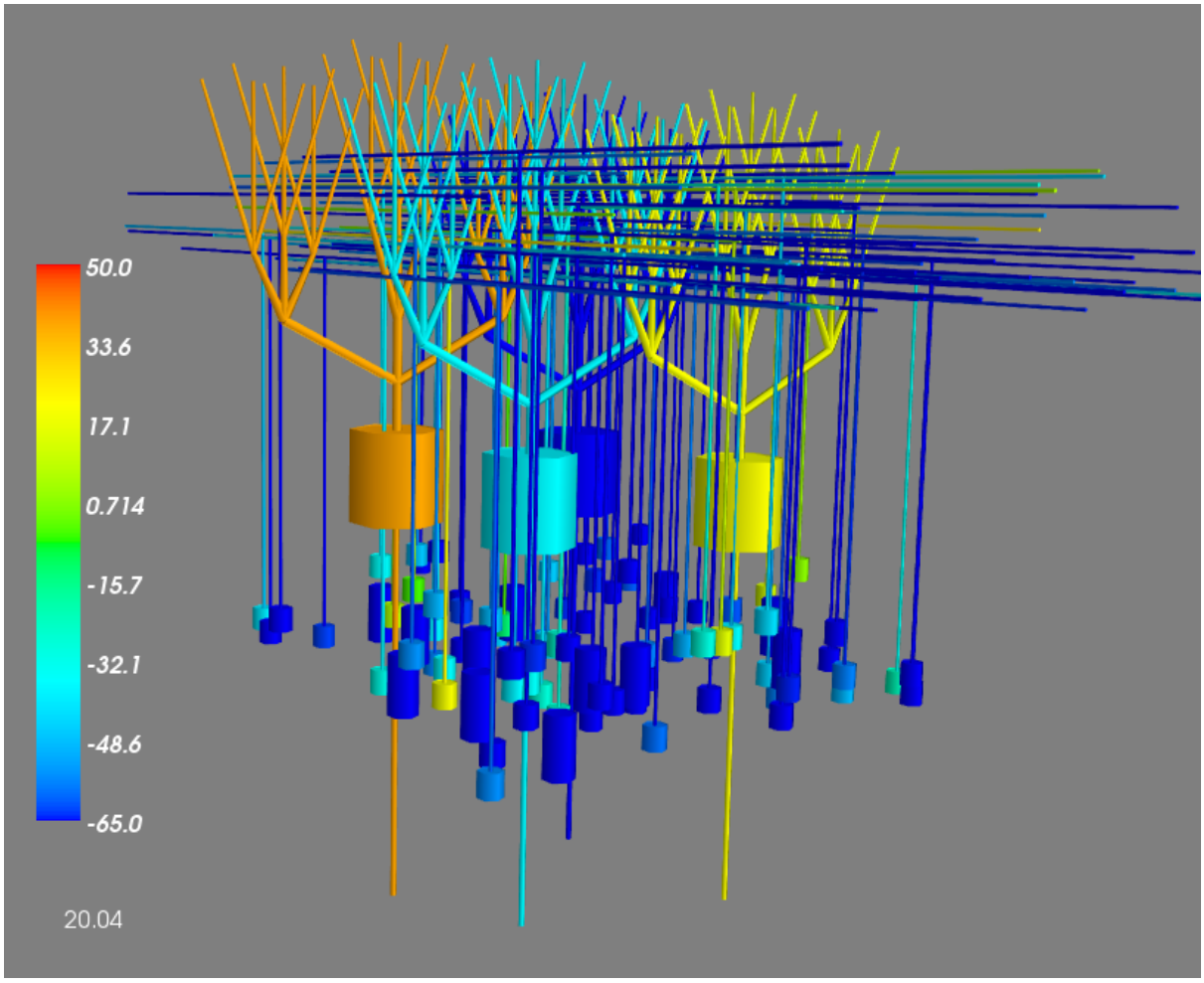
It is possible to use the standard pylab tool and to plot the timecourses in the current figure, or create a new one just selecting another output figure number.



4.4 Network example

Neuronvisio can handle the visualization of networks. For example check out the Cerebellum Networks example, ported to NEURON with neuroConstruct.

This Cerebellum Network example is shipped with Neuronvisio code.



4.5 Making a movie

To make a movie it is possible to call the function `neuronvisio.controls.Controls.make_animation_screenshot`

```
controls.make_animation_screenshots(time_start, time_stop=None,
                                    saving_dir='anim')
```

which will save all the screenshots in brand new directory (default is anim). To stack them in a movie, you can use ffmpeg with the following command:

```
ffmpeg -f image2 -r 10 -i %09d.png -sameq anim.mov -pass 2
```

One of the example video, using the pyrimidal neuron can be seen here: http://www.youtube.com/watch?v=LOuptLKZ5rU

CHAPTER 5

Reference

The API directly from the docstrings in the code.

5.1 neuronvisio.manager – Manage the map between vectors and sections

5.1.1 Manager

5.1.2 BaseRef

5.1.3 VecRef

5.1.4 SynVecRef

5.2 neuronvisio.visio – 3D Visual operations

5.2.1 Visio

5.3 neuronvisio.controls – User Interface module

5.3.1 Controls

5.3.2 Timeloop

CHAPTER 6

Changes in Neuronvisio

6.1 0.9.1 – 1 October 2015

- Fixed small formatting issue on the docs
- Added pytables to the run dependences

6.2 0.9.0 – 24 September 2015

- Created a conda package for Neuronvisio
- Dropped paver for normal install via PyPi (Closed #57)

6.3 0.8.6 – 8 March 2013

- we are using pip and requirements.txt to be able to install packages from DCVS
- MacOs is now supported when we open the model, if not mosinit is present #56
- Better explanation to the User if a model can't be loaded. #55
- Improved Install Docs for Mac. PR #53 and #54
- Added 24 models available from ModelDB

6.4 0.8.5 – 6 Jul 2012

- Closed #49
- Text in the Animation is always formatted with 3 digits.
- Added the ability to load a NeuroML (xml), a Hoc (.hoc) or a HDF file (.h5) with Neuronvisio format directly when the program is launched.

6.5 0.8.4 – 12 Jun 2012

- Added new model to the ModelDBlist
- updated the documentation
- Added 3 new models to ModelDB

6.6 0.8.3 – 18 May 2012

- Fixed paver minilib, missing from the tarball
- Added one new model to the ModelDB.xml
- Improved the documentation of several classes
- Inserted the domain in the about (neuronvisio.org)

6.7 0.8.2 – 23 Apr 2012

- Added two new model to ModelDB.xml
- Introduced a new method to assign a color to more than one sections from the console
- Added the apptools patched version as explicit dependency until it's not released officially.

6.8 0.8.1 – 19 March 2012

- Added new models on the ModelDB.xml
- Closed #45
- Easy function to export a video as a stack of screenshots
- Updated the docs to work witht the latest pyqt
- Geometry node in the HDF is directly a NeuroML string and not wrapped in a list

6.9 0.8.0 – 15 Feb 2012

- New repo layout
- Ported to Mayavi 4.1.1.dev
- Closed #31
- Reorganized controls method in alphabetical order.
- Animation set true when simulation is launched.
- Fixed a vectors handling if no values present.
- Updated the doc for Mayavi 4.0.0
- Load hoc or hdf with the same method
- Added new models from ModelDB
- User Interface improvement
- Closed #37
- Closed #32
- Closed #34
- Improved nrnvisio.py to integrate seamlessy with IPython session, or create a new one on the fly
- Automatic handling of Mayavi installation through pip.

6.10 0.7.3 – 24 Nov 2011

- Inserted how to configure ipython with the pylab and Q4Agg support in the doc
- Selected section is now stored and used if the vector are created only for one sec
- Inserted two new models on the ModelDb.xml. Fix #30.
- Included the ModelDB.xml in the package

6.11 0.7.2 – 8 Nov 2011

- Inserted modeldb in the pavement.

6.12 0.7.1 – 8 Nov 2011

- fixed the name of the exectable on the setup file

6.13 0.7.0 – 3 Nov 2011

- Added a new tab to have a ModelDb explorer inside Neuronvisio
- Integrate the ability to download and extract a NEURON model from ModelDb
- Ability to load the model directly from Neuronvisio
- Used the logging python system through the package
- Updated the docs to explain the ModelDb integration.

6.14 0.6.2 – 16 Jun 2011

- New API to select sections directly from the commandline (*controls.select_sections()*)
- Animation Timeliner set up only if simulation has ran or be reloaded.

6.15 0.6.1 – 9 Jun 2011

- Animation ported to the new infrastructure.
- Updated the requirements with setuptools (needed for paver)
- Animation timeline accept an arbitrary time and display it on the visio window.

6.16 0.6.0 – 10 May 2011

- Added the function to plot a 3D plot to the manager (manager.plot3D)
- Animation can will be enabled if any simulation is ran, either from the gui or the code.
- Completely rewrote the visualization method. Now completely integrates in the mayavi pipelines, therefore axes and other modules/filters can be added at will
- Now it's faster. A lot faster

- All the segments are plotted, not only the section. This is extremely helpful with geometrical reconstruct neurons and networks.
- Updated the docs and the screenshots.

6.17 0.5.2 – 26 Jan 2011

- Updated the version recall to GitPython 0.3.1 (used only if present!)
- HocVector into NumpyArray for saving with swap in place, to reduce overhead
- Restructured the package for an easier installation
- Added the possibility to build documentation offline

6.18 0.5.1 - 23 Nov 2010

- Fixed the picking of the cylinder. Possible to select a cylinder clicking anywhere.
- Possibility to plot points instead of a lines
- BaseREf class are discriminated through the group id and not any more on class base.

6.19 0.5.0 - 19 Jun 2010

- Closed #16
- Fixed some typos on the docs
- Mechanisms are shown on the info tab
- Refactored code for extensibility
- Storage moved to a hdf file.
- Extensibility to other kind of variables, not only vectors

6.20 0.4.4 - 1 Apr 2010

- Fixed the name on the README
- Treeview updated everytime a database is loaded.

6.21 0.4.3 - 2 Mar 2010

- Info sections updated
- Update the docs and website

6.22 0.4.2 - 18 Feb 2010

- Added simulation saving abilities.
- Updated the doc

6.23 0.4.1 - 28 Jan 2010

- Closed #13
- Introduced a tab to retrieve info on the section

6.24 0.4.0 - 19 Jan 2010

- Remplemented using Mayavi2 and Qt4 for better performance and better usability.
- Cleanup and refactoring of the code.
- Closed #11, #12, #15

6.25 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

6.26 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.

6.27 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.

6.28 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)

6.29 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.

6.30 0.3.2 - 20 Jul 2009

Bug Release. Closed Bug #9

6.31 0.3.1 - 18 Jul 2009

Bug Release.

6.32 0.3.0 - 14 Jul 2009

6.32.1 New Features

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

6.32.2 BUGFixes

- Closed bug #8
- Closed bug #3

6.33 0.2.0 - 6 Jul 2009

6.33.1 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.

6.33.2 BUGFixes

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

6.34 0.1.0 - 30 Jun 2009

Fist public release.

6.34.1 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.