Backwards-compatibility in GENESIS 3.0 and beyond

Hugo Cornelis[1], Allan C Coop[2], Armando L Rodriguez[3], David Beeman[4], James M Bower[3]
[1] Department of Neurophysiology, Catholic University of Leuven, Leuven, 3000, Belgium
[2] Merindah Energy Ltd., Freemansreach, NSW, 2756, Australia
[3] Research Imaging Institute, University of Texas Health Science Center, San Antonio, TX 78229, USA
[4] Department of Electrical, Computer, and Energy Engineering, University of Colorado, Boulder, CO 80309, USA
E-mail: hugo.cornelis@gmail.com

UT Health Science Center San Antonio

Introduction

This poster describes from a users perspective, how the GENESIS 3 (G-3) CBI architecture discussed in Poster #15 allows G-3 users to maintain backwards compatibility with GENESIS 2 (G-2), while exploiting the multiscale modeling and interoperablity features of G-3. G-3 is in a phase of rapid development, making the transition from a tool that can be used by advanced users and modeler/developers to (soon, we hope) a user-friendly tool for novices. The examples given here are based on the current July 2011 GENESIS 3 Developers Release, available from http://genesis-sim.org. More detailed tutorials based on these examples can be found on the genesis-sim.org web site.

The previous poster explains the G-3 user workflow and separation of the monolithic architecture of G-2 into independent software components. The most relevant components for the examples given here are:

- Neurospaces Model Container (NMC) separates the biological model description from the details of implementation
- Multiple solvers perform numerical calculations and allow highly efficient solvers to be implemented for particular model objects. Heccer is the default solver for compartmental models, which transparently incorporates the hsolve object of G2.
- A scheduler (SSP or SSPy) to run the simulation
- The G-shell (or the new Python shell) to give interactive commands.

Conversion of G-2 cell models to G-3

As G-2 models are converted to the NDF representation used in G-3, they are added to the G-3 model library. After an installation of G-3, they are available in subdirectories of /usr/local/neurospaces/models/library. The list of model categories may be seen with the G-shell command 'library_show'. The cells and channels are listed with the commands 'library_show ndf cells' and 'library_show ndf channels'. Some of the cell models available in the cells/ subdirectory are:

- The basic two-compartment 'simplecell' model that is used in many G-2 tutorials.

traub91-nolib.ndf - The 'traub91' model is a burst-firing CA3 region hippocampal pyramidal cell, using a linear arrangement of 19 compartments containing active conductances in all compartments.

traub94-nolib.ndf - A burst-firing hippocampal pyramidal cell using 64 asymmetric compartments in a branched geometry, containing active conductances in all compartments.

traub95-nolib.ndf - A fast spiking hippocampal interneuron, using 51 branched asymmetric compartments containing active conductances in all compartments.

RScell-nolib.ndf - RScell is a single compartment regular spiking cell used in the RSnet simulation from the GENESIS Modeling Tutorial section "Creating large networks with GENESIS".

BDK5cell2-nolib.ndf - The 'BDK5cell2' model is a branched layer 5 cortical pyramidal cell with 9 compartments and 9 voltage or calcium activated channels in the soma.

purkinje/edsjb1994.ndf - This file and other variations represent the De Schutter and Bower (1994) large Purkinje cell model.

Example: Conversion of a fast spiking striatal interneuron model to G-3

This example of a version of the 127-compartment spiking striatal interneuron model by Kotaleski, Plenz and Blackell (2006) does not yet exist in the library. To create the necessary NDF format file, one uses the ns-sli backwards comptibility module, invoked in the G-shell to load the G-2 SLI script and save it in NDF format:

[redacted]

To be sure that you remember the name that was assigned to your cell model (fs in this case) give the command:

[redacted]

and continue with:

[redacted]

Running the model in the G-shell
The G-shell commands (described in the G-3 tutorials) to run a current injection experiment on this model and output the soma membrane potential to a file fsi_Vm.out are:

[redacted]

Using SSPy to run models in a shell or with a Python script

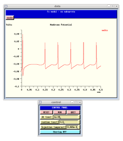
The new Python interface offers a shell with equivalent functionality to the G-shell. Most importantly, it provides an Applications Programming Interface (API) for interfacing with scripts written in Python. This means that a Python script that loads and runs a G-3 simulation, can also make use of the many GUI toolkits (e.g. wxPython), analysis and visualization tools (e.g. Matplotlib, which provides Python objects to replicate much of the functionality of Matlab), and Python modules for scientific computing such as scipy and numpy.

Example Python script to run the **striatal_fsi_KPB** simulation
The following commands are described in the tutorial "Creating GENESIS 3 Simulations with Python":

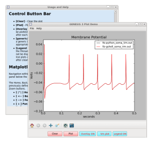
[redacted]

[redacted]

The plots below show the plotted results of running the simulations described above.



G-2 XODUS-based GUI for running and displaying results



G3Plot.py standalone application for plotting G-3 simulation results

Interfacing simulation output with graphics using Python

In the line for 'CreateOutput' in the listing above. the output type is set to 'double_2_ascii', which produces file output similar to the G-2 'asc_file' object. If the type is set to 'line', the output will be sent one line at a time to stdout, normally the console. This is useful when piping the output to another program or Python object for analysis or plotting. When the output object type is 'live_output', the data is output to a list of lists.
The 'live_output' output object type can be used to make simulation output easily accessible for plotting within the Python simulation script. The script shown above can be modified to end with the statements:

[redacted]

This produces a plot similar to that produced by the G-3 standalone application plotVm.py, included with the current G-3 distribution.

Extending G-3 multiscale modeling with Chemesis

Chemesis is a G-2 'add-on' library of objects for modeling biochemical reactions, models of second messengers, and calcium dynamics, created by K. T. (Avrama) Blackwell (2000). As a test case for user-extensibility of G-3 we began implementing the Chemesis library as a G-3 software component. Our goal was to determine how difficult and time-consuming it would be to implement a new G-3 software component chemesis3 based on a G-2 implementation of a set of new objects. The original implementation relied on the generic G-2 numerical solver. Only limited time was available for the implementation of the specialized chemesis-3 solver. A time line reconstructed from the email conversations and from the version control system shows the course of development:

- exploratory email conversations May 30th and following two weeks.
- initial preparations and start of implementation on June 13th.
- core implementation on Sunday June 26th, including a fully working cal1 regression test case. This was a day of crazy coding as in the old days, total of 18 revisions with many enhancements.
- implementation of cal2 test case on June 29th and July 10th.
- initial scripting bindings were added starting at July 10th for perl and July 13 for Python.
- first successful integration with the SSP scheduler on July 12th.
- model-container bindings started on July 13 and finished on July 17th. Removed model-related functions such as compartment volume computation that are already available in the model-container (and now shared with other solvers).
- G-shell integration on July 17th and July 18th.

The objects currently implemented in the G-3 chemesis3 component are:

rxnpool: a concentration pool that interacts with reactions and diffuses to other pools. (Maps to NDF token POOL.)

conservepool: mass conservation based pool, computes the difference between the total of all molecules (a model parameter, rest state), and diffused molecules, divided by compartment volume. (Maps to NDF token POOL, with a different parameterization.)

reaction: standard forward / backward chemical reactions between pools of molecules. (Maps to NDF token REACTION.)

diffusion: Computes flux in molecules between two pools. (Maps to NDF segment parameter diffusion_constant. The appearance of this parameter in an NDF file requires running the model using a chemesis3 solver.)

Our goal was to implement and run two of the G-2 Chemesis tutorial examples.

The example script cal1.g creates a single compartment with interaction between calcium and a buffer. A second example cal2.g creates a two compartment model with a dendrite and soma. One additional diffusion object is required to allow for diffusion between compartments.

The two listings below show the G-2 SLI language script cal1.g and the G-3 NDF representation cal1.ndf, found with cal2.ndf in **'/usr/local/neurospaces/models/library/chemesis'**.

Cal1.g:

[redacted]

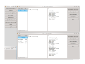
Cal1.ndf:

[redacted]

The commands below illustrate how the G-3 Studio (model explorer) is used to load cal1.ndf into the Model Container and explore the model:

[redacted]

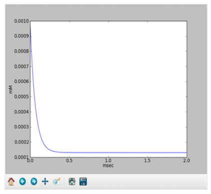
Exploring Cal1.ndf



Exploring Cal2.ndf

Comparison of G-2 and G-3 results

The plots below show results from the G-2 and G-3 versions of the cal1 simulation. Testing revealed that on average the G-3 version with its own solver ran about three times faster than the G-2 implementation.



Conclusion

Is GENESIS 3 ready for serious modeling use? Try the latest Developers release and see!

http://genesis-sim.org

References

Blackwell K.T. and Hellgren Kotaleski J. (2002) Modeling the dynamics of second messenger pathways, In: Neuroscience Databases: A Practical Guide, Ed. R. Kotter. Kluwer Academic Publishers, Norwell, MA

Blackwell, K. T. (2000) Evidence for a Distinct Light-Induced Calcium-Dependent Potassium Current in Hermissenda Crassicornis, J. Computational Neuroscience, 9: 149-170.

DeSchutter, E., and Bower, J.M. (1994) An Active Membrane Model of the Cerebellar Purkinje Cell: I simulation of current clamps in slice J. Neurophysiol. 71:375-400.

Kotaleski, J. H., Plenz, D. and Blackwell, K. T. (2006) Using potassium currents to solve signal-to-noise problems in inhibitory feedforward networks of the striatum. J. Neurophysiol. 95: 331-341.
.