Soltesz Lab NEURON Model

Marianne Case

September 2, 2009

Abstract

This paper explains all aspects of the NEURON model created in Dr. Soltesz's lab. Specifically, it describes in detail how the code for our model works. Lists of input files needed by the code and result files produced by the code are included. It also documents one way to get NEURON running, in parallel, on a cluster computer.

Contents

1 Introduction

The model currently represents the dentate gyrus using 4 cell types. It is scaleable and runs in parallel. More detailed information follows.

2 Version History

Version	Release Date
model-2.3	2009-10-15
model-2.2	2009-09-28
model-2.1	2009-09-08
model-2.0	2009-09-02
50knet	2009-04-20

Table 1: Dates of major code changes

2.1 model-2.3

The option to print out a number of connections (summary) file was introduced here, and the options and parameters were made more straight forward. They are now set in a hoc file (allowing comments) rather than a dat file. It was also verified that the basket cell - basket cell threshold (which had been off from the others in the 50knet version) should in fact be uniform, so a comment warning that the connection's threshold was different, having been set to the same as the others, was removed. The runtimes for all processors are now written out to the runtime file. Some comments output to the command prompt now have the host number included. Now track the number of connections to each post cell type in the pre cell's category object. Option to print out whole connection file immediately after being created (2), after everything else has run or printed out (1) or not at all (0).

2.2 model-2.2

The option not to print out the connection and position files was introduced here, because the file was taking too long to print out with larger numbers of connections.

2.3 model-2.1

Time print outs and files are produced as soon as possible in model 2.1, where they were produced at the end of the program in the previous version. Now, the time taken for each section is printed out after each section finishes. The files are printed out after all information needed for the files has been produced by the program.

2.4 model-2.0

The major changes from the previous version of the model are that this model runs in parallel and that all hard-coded references to cell types have been removed from the main code to facilitate adding additional cell types. In detail, the following changes have been made in the upgrade from 50knet to model-2.0:

1. General Code Changes
   (a) Parallel (thanks to Michael Hines)
   (b) Hard-coded cell type references removed from code
   (c) All 'pre- and post- cell type dependent parameters' moved to pre.post cell connection info file
   (d) Cell templates redefined to store position information
      i. Position info generated algorithmically and stored in cell objects instead of in separate vectors using bins
   (e) General celltype class defined to store some info (axon length distributions, whether cell type is subject to sclerosis, number of cells) to streamline the code
   (f) Each cell now has its own random number generator
   (g) Random number stream seeds defined so that no part of stream is touched twice
   (h) Synapse types are stored in separate lists by pre-cell type
   (i) Thresholds now set in cell templates
2. Clean-up Changes
   (a) Cell Templates defined in separate files
   (b) Cell names spelled out for clarity
   (c) Removed obsolete references
   (d) Removed GUI references
   (e) Simplified many functions - main code reduced from greater than 1600 lines to fewer than 500 lines
3. NEURON Mechanism Changes
   (a) Exp2Syn mechanism altered (to Exp2Sid) to allow more robust tracking of connections between cells and synapse types involved in the connections [HC08]
4. In-code Documentation Changes
   (a) Added comments to each novel line of code - there are now over 200 comments
   (b) Separated code into logical sections with descriptive headings
5. Print Out Changes

(a) Added timers to each section of code, which print to command prompt and to file

(b) Introduced variable print flag to control amount of output to command prompt

(c) Added descriptive command prompt printouts to the code

6. File Output Additions

(a) Position of each cell and processor on which each cell resides

(b) Connection information for each cell, including type of synapse

(c) Name and index range of each cell type output

(d) Timing of each section of code, and total code

2.5 50knet

This version of the code was inherited from Rob Morgan in April 2009. It was a scaleable, functional network of the dentate gyrus. It had scaled up from 500 granule cells to 50,000 granule cells and included 4 cell types (model size is given in terms of the number of granule cells, assuming proportional numbers of all other cells). This version ran in serial only.

3 Outline of Code

An outline detailing the structure and flow of the current model version follows:

1. LOAD LIBRARIES & PARAMETERS

(a) Load the main nrngui.hoc

(b) Load the template for the parallelnetmanager class (used to parallelize the code)

(c) Load the template for the randomstream class (used to generate random number streams)

(d) Load the template for CellCategoryInfo class, which generates 1 object per cell type to store celltype specific data and enable the removal of all hard-coded cell type refs

(e) Load the hoc file containing defined and calculated parameters

2. SET MODEL SIZE, CELL DEFINITIONS

(a) Calculate numbers of cells of each type

(b) Load celltype specific data

i. For each cell type specified in the cells2include.hoc file

A. Read in the cell name, # cells, layer specifier, stim input specifier

B. Compute the start and end of the gid # range

C. Create an object of the CellCategoryInfo class and store the data from the above two points

D. Load the hoc file containing the celltype class template

ii. Load the hoc file containing the perforant path stimulation template

(c) Kill off the % specified by the sclerosis factor of cells that were specified as being in the hilar layer

(d) Recalculate the gid ranges for each cell type, now that the number of cells for some types has decreased

(e) Calculate the total number of cells including and excluding the perforant path stimulation cell(s)

3. SET UP PARALLEL CAPABILITY

(a) Set up a ParallelNetManager object

(b) Create a ParallelContext

(c) Call the round robin command, which distributes all cells among all processors in a round

(d) Define an iterator that can iterate over all the cells in a given range that are owned by the host that called the iterator

4. CREATE, UNIQUELY ID, AND POSITION CELLS

(a) For each cell type defined in cells2include.hoc:

i. For each host in the cluster:

A. Check that the gid is owned by the host (and it should be, because we are using the iterator)

B. Create a cell of that cell type and reference it in a list contained by that cellType object

C. Add the cell to the 'cells' list (this is something proposed by NEURON developers, but we don't use explicitly use it in the code)

D. Create an empty connection for the cell to use as a spike detector

E. Associate the cell with its gid and with the spike generation location (empty connection)

F. Calculate and store the cell's position using an algorithm based on gid, cell type, # of cells of that type, # available 'bins'

5. CONNECT THE CELLS AND CONNECT THE PERFORANT PATH TO SOME CELLS

(a) For each cell type x cell type combination, load in the connection properties (weight, delay, probability)

i. If the probability of connection is not 0, for a given cell type x cell type combination:

ii. Iterate through each (potential) post-synaptic cell of that type that exists on the host executing this code

A. Iterate through each (potential) pre-synaptic cell of that type (regardless of where it lives)
B. Algorithmically obtain the positions of the pre- and post- cells
C. Calculate the distance between the cells and obtain a probability factor based on the distance and distribution of axon length for the pre-synaptic cell
D. Multiply the distribution number by the probability of those two types of cells connecting and by a connection factor that was artificially added to ensure the proper amount of connectivity between cells in a network of a given size
E. Pick a random number between 0 and 1 and check if it is less than this product
F. If the above statement is true, make a connection between the cells with the given weight and delay and add the connection to a list 'nclist' (this is something proposed by NEURON developers)

(b) For each perforant path stimulator cell:

i. Connect the cell with the middle 10% of granule cells in the network
ii. Connect the cell with the middle 10% of basket cells in the network
iii. Make 10 connections to the middle 10% of mossy cells in the network (note that with small network sizes, a given cell may receive more than 1 connection)

6. INITIALIZE AND RUN NETWORK, OUTPUT RESULT FILES

(a) Initialize the network using parameters specified above

(b) Run a low resolution 'pre-simulation' to allow cells to 'settle' and all components to reach steady state

(c) Set the program to records all spikes of all cells on the host executing this code

(d) Run the simulation for the time specified in part I, at the resolution specified in part I

(e) Optionally output various result files:

i. A spike raster file giving spike times and gids of spiking cells
ii. A connection file that gives pre- and post- synaptic cell gids and synapse types
iii. A position file that gives the gid and x, y, and z coordinates of each cell
iv. A cell type file that gives cell name and gid range for each cell type
v. A runtimes file that gives the real time taken by each code section in seconds

4 Model Input Files

Library Files

nrngui.hoc The main library file included in the NEURON installation

netparmpi.hoc The main parallel library file included in the NEURON installation

ranstream.hoc A homemade class template that uses the Random class to generate random number streams

CellCategoryInfo.hoc A homemade class template that will be used to store cell type specific data (1 object generated per cell type)

Per Cell Type Files

class_celltemplatename.hoc A file that defines a class for each cell type, including celltype specific mechanisms and parameters

dist_celltemplatename.hoc A file that defines the axon length distribution for each cell type, currently formatted to provide 199 length probabilities, where length 100 is 0 away from the current location and lengths 1 and 199 are as far away as possible

Per Cell Type x Cell Type Connection Files

celltemplatename.celltemplatename A file that defines connection properties for the connection between the two cell types listed in the file name (first name is the presynaptic cell type). The properties are, in order: weight, delay, probability of connection

Parameter Files

parameters.hoc This commented file contains parameters needed by the model to control its behavior:

- **flagprint** Controls how many notes print to screen. 0=few, 1=some, 2=many
- **printNumCon** Controls whether to print the Number of COnnections (1=yes, 0=no)
- **printConMat** Controls whether to print the detailed COnnection list (2=yes immediately, 1=yes at end of program, 0=no)
- **printPosition** Controls whether to print the position list (1=yes, 0=no)
- **percentSclerosis** % of cell death, for cells flagged as susceptible to death
- **connectionFactor** factor to control # connections formed in the network (dependent on # of cells: 500 for 500, 5 for 50k, 1 for 1 million)
- **randnet** Controls whether to decrease granule cell sprouting by same percentage as sclerosis
- **secondorder** A global variable which specifies the time integration method
- **tstart** Start time of simulation
- **tstop** Stop time of simulation
- **dt** Integration interval for fadvance

v_init All voltages of all sections are initialized to this voltage (mV)

N Specifies step resolution (see NEURON reference for steps_per_ms)

steps_per_ms Controls the intervals at which points are plotted

dentateBins Equivalent subdivisions of the dentate

dentateLength Length of dentate in microns

dentateBinSize Length of each bin (along x dimension) in microns

noise_random_stream_offset_ Length of stream to reserve for each random number sequence

cells2include.hoc Lists the number of cell types to include followed by cell specific descriptors:

Cell Type Name Name of the cell type, must exactly match its class name

 # Cells Number of cells of this type in the model (prior to any sclerosis)

Layer Flag The layer of the dentate gyrus in which that cell type is found (hilar=1, granular=0), where hilar cells are subject to sclerosis

Note, the artificial stimulator cell must be listed as the first cell type. Other cell types can follow in no particular order.

5 Model Output Files

All the output files are written to the same directory as the one where the code was executed. All files except the spikeraster have headers giving the column name. All the files can be read by MatLab and are expected to be in the current format by the loadresults.m dashboard generation file. Note that the position, connection, and numcons files have options to be written, so they may not always be produced from the program.

spikeraster.dat Spike raster file giving spike times and gids of spiking cells

numcons.dat Summary connection file that gives pre- and post- synaptic cell types and # connections

connections.dat Connection file that gives pre- and post- synaptic cell gids and synapse types

position.dat Position file that gives the gid and x, y, and z coordinates of each cell, as well as the node on which the cell resides

celltype.dat File that gives cell name and gid range for each cell type

runtimes.dat Runtimes file that gives the real time taken by each code section in seconds

Appendices

A Installation, Set Up, Execution

This section explains what is required to run the model program on a computer. It includes a list of required software that must be installed prior to installing and running the program. It also details how to install the required software and how to install the program itself.

To run our model program in parallel on multiple nodes, the following hardware is required:

- **Multiple nodes:** We use Linux computers with Redhat. Memory requirements could be in the tens of GB and other requirements are unknown as of yet.

The following software is required:

- **Compiler:** preferably parallel (we are using Portland Group's PGCC compiler)
- **MPICh2:** to enable all the nodes on a cluster to run in parallel
- **NEURON:** to execute the model code

The following software is optional:

- **MatLab:** only necessary if you want to view the output files in the 'dashboard' format

Each of the software items will be covered in a separate subsection.

A.1 Compiler

Talk to Steve Chen, our administrator, for assistance with setting up a compiler on a cluster:
Phone: 824-2882
Email: steveywc@uci.edu

A.2 MPICh2

Follow the Quick Start instructions in this installation guide to install MPICh2 on a cluster:
http://www.mcs.anl.gov/research/projects/mpich2/documentation/files/mpich2-1.1-installguide.pdf

A.3 NEURON

Follow the installation instructions for NEURON available on the website. We install it on the cluster from source code:
http://www.neuron.yale.edu/neuron/install/getstd.html

A.4 Model Code

To run our model on the cluster, do the following:

A.4.1 Initial Set Up

1. Save the main code and the supporting files in your directory of choice (all in the same directory):
   (a) Main code (ex: Model-2.0.hoc)
   (b) Cell template files: [celltypename].hoc (ex: granulecell.hoc); save one file for each cell type. The file contains only a template definition of the [celltypename] class
   (c) Cell type x cell type connection files: [precelltypename].[postcelltypename] (ex: granulecell.mossycell); save one file for each combination of cell type x cell type, including same-cell type connections
   (d) Distribution files: dist_[celltypename].hoc (ex: dist_granulecell.hoc); save one file for each cell type that gives the distribution of axonal connections lengths
   (e) Mod files: Save all the mechanism files in this same directory
   (f) Parameter files: parameters3d.dat and cells2include.hoc must also be saved here

   Note: it is helpful to have an FTP software like WinSCP to transfer the files over. WinSCP is available online as a free download and can easily be found by a search engine.

2. Compile the mod files by entering the following command:
   `nrnivmodl`

A.4.2 Each Time

1. Edit the parameter input files (listed below) as necessary

2. Turn on mpd by entering the following command (see A.5 if you are confused about where to type this):
   `mpdboot`

3. Call the program and replace '2' with the number of available hosts:
   `mpiexec -n 2 nrngui -nobanner -nogui -mpi model-2.0.hoc`
   Or, to run the program in the background, use the following:
   `nohup mpiexec -n 2 nrniv -nobanner -nogui -mpi model-2.0.hoc > nohup.out < /dev/null &`

4. When finished using mpi, shut down mpd:
   `mpdallexit`

A.5 Accessing the Cluster

To access the server or cluster from your personal computer, two things are required:

1. An account on the server or cluster

2. Software to interact with the server or cluster remotely

To obtain an account on the

1. Lab Server
   (a) Have an existing lab member log in as root
   (b) In the system settings, go to user groups and add a user
   (c) Add the directories: /data/[newuser], /backup/[newuser], (/home/[newuser] may automatically be created)
   (d) Then, double-click the user, go to groups. Add the user to the jay group (and possibly other groups?)
2. Cluster
   (a) Email or call Steve Chen to request the new account

To interact remotely with the server or cluster: Cygwin is recommended by the NEURON developers. They post instructions on their website for installation and customization of the Cygwin bash shell. This software is installed on your personal computer, not on the server or cluster.
http://www.neuron.yale.edu/neuron/install/getcygwin.html

Once you have installed the software and obtained an account, you can access the server or cluster, by doing the following:

1. Start an XWin server session
2. Open an rxvt shell
3. Type the following at the command prompt to access:
   (a) Server (two processors)
   `ssh -X -Y username@solteszserver.anat.uci.edu`
      i. -X is to let the computer talk to you
      ii. -Y is to trust the account (?)
      iii. username is your own username, chosen when your account was set up on the server or cluster
      i. Cluster (8 nodes)
      `ssh -X -Y username@128.200.145.87`
   (b) Enter your password when prompted
   (c) You are now in your home directory on the server and can type commands into the command prompt to control the server

B Viewing the Matlab Dashboard

There is a standard MatLab program that can show a 'dashboard' of the model program's results. The benefit of using this dashboard is that it gives a standard, reproducible view. It provides several data types that are useful, and it doesn't have to be rewritten each time. The dashboard can be viewed on any computer that has MatLab, just by moving the result files generated from the model run to that computer, and the 'loadresults.m' file into MatLab's work directory on that computer. From the command prompt in MatLab, call the dashboard program with the path where the result files are located as an argument. A dashboard will appear as a new figure.

If additional MatLab analysis needs to be done, it may be faster to do the analysis on the cluster. Of course, the end product can be transfered to another machine for viewing. To use MatLab on the cluster, please talk to Steve Chen. The Parallel Toolbox for MatLab will also need to be installed to take advantage of the cluster.

B.1 Each Time

1. Enter the following at the command prompt before starting Matlab:
   `LD_ASSUME_KERNEL=2.4.19`
   Note: The server's version of Matlab with the Linux version of Matlab has a problem that requires you to use a different kernel

2. Switch the single matlab user license over to yourself if someone else used it last:

   (a) Open the following file: /usr/local/matlab/etc/MLM.opt

   (b) Replace the existing username with your username

   (c) Ensure the location of the license manager is in your $PATH variable: /usr/local/matlab/etc

   (d) Enter `lmstart` at the command line to rerun the license manager

3. Add the Matlab path to your $PATH variable: /usr/local/matlab/bin

   (a) Then export the path to keep from having to append that directory each time

4. Type in `matlab` at the command prompt

   (a) You may also type in `matlab -nodisplay -nojvm` to run matlab without the display and without java

B.2 License Notes

- The license # for Matlab 2006a linux version is: 257746
- The license file must contain the following text:

BEGIN——cut here——CUT HERE——BEGIN # MATLAB license passcode file for use with FLEXlm. # LicenseNo: 257746 HostID: ID=257746 SERVER solteszserver.anat.uci.edu ID=257746 27000 DAEMON MLM /usr/local/matlab/etc/lm_matlab options=/usr/local/matlab/etc/MLM.opt INCREMENT TMW_Archive MLM 14 01-jan-0000 0 0DC102C903CBC1131470 VENDOR_STRING=1 HOSTID=DEMO SN=257746 INCREMENT MATLAB MLM 14 01-jan-0000 1 5D718259B08BA40BEB2D USER_BASED DUP_GROUP=U SN=257746 # END——cut here——CUT HERE——END

References

[HC08] M. L. Hines and N. T. Carnevale. Translating network models to parallel hardware in neuron. *J Neurosci Methods*, 169(2):425–55, 2008. Hines, M L Carnevale, N T R01 NS011613-31/NS/NINDS NIH HHS/United States Netherlands Journal of neuroscience methods Nihms47936 J Neurosci Methods. 2008 Apr 30;169(2):425-55. Epub 2007 Sep 16.