The NEURON Simulator

neuron.yale.edu

School of Brain Cells and Circuits; Erice 2015

Supported by NINDS

General

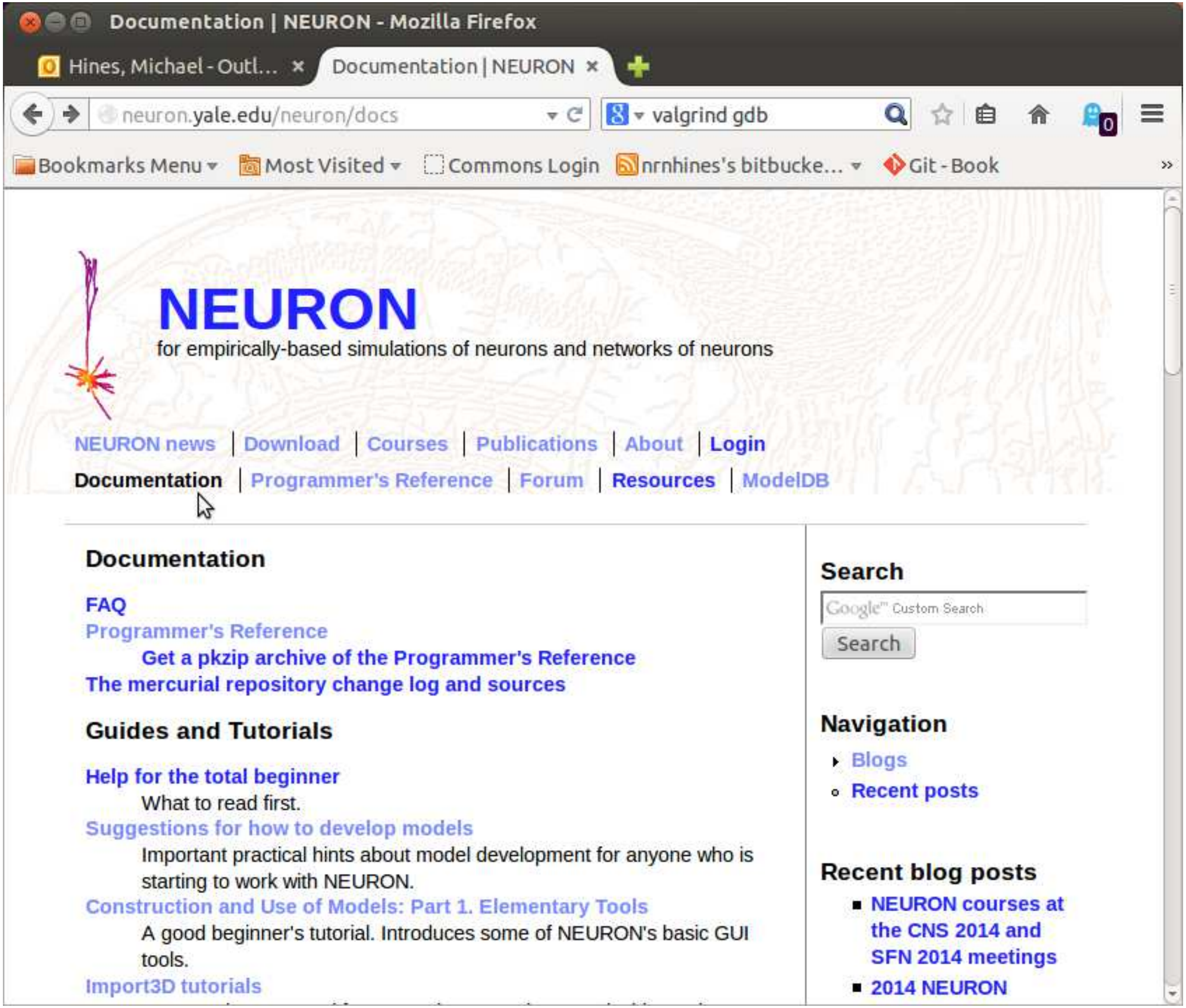
Slides available at:
http://neuron.yale.edu/ftp/neuron/neuron-erice2015.pdf

NEURON + Python

http://neuron.yale.edu/ftp/neuron/neuron-python-erice2015.pdf

NEURON + Python tutorial

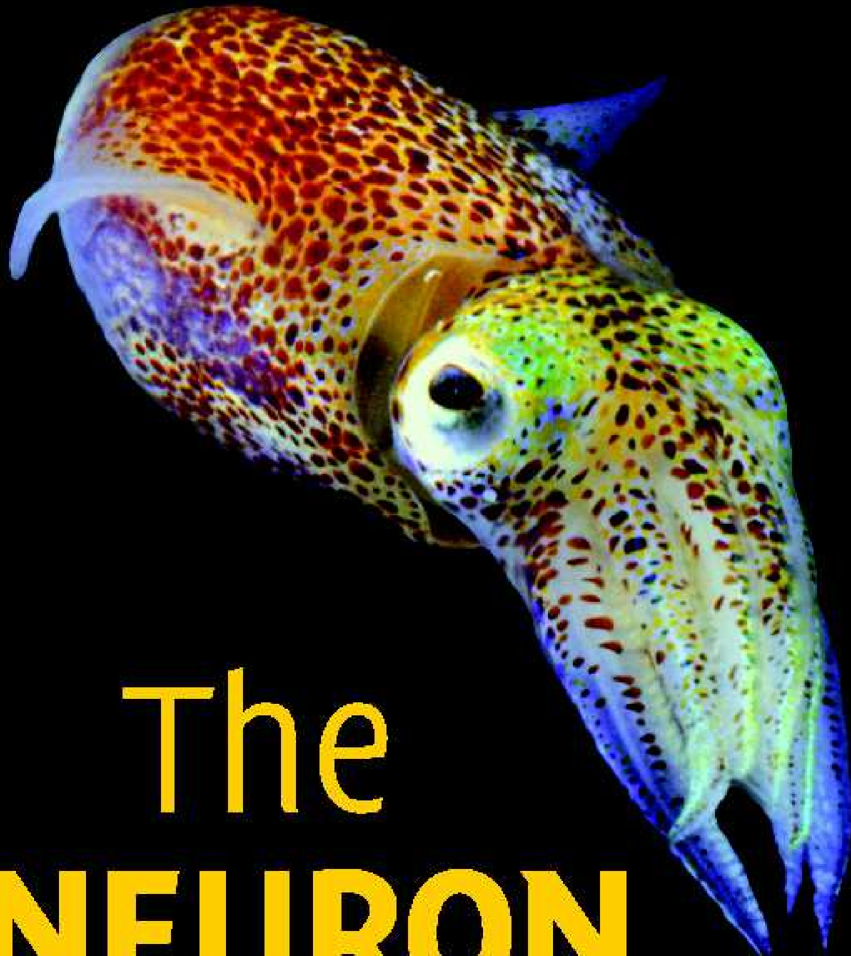
http://neuron.yale.edu/neuron/static/docs/neuronpython/index.html


Documentation | NEURON - Mozilla Firefox
Hines, Michael - Outl...
Documentation | NEURON
neuron.yale.edu/neuron/docs
valgrind gdb
Bookmarks Menu
Most Visited
Commons Login
nrnhines's bitbucke...
Git - Book
NEURON
for empirically-based simulations of neurons and networks of neurons
NEURON news | Download | Courses | Publications | About | Login
Documentation | Programmer's Reference | Forum | Resources | ModelDB
Documentation
FAQ
Programmer's Reference
Get a pkzip archive of the Programmer's Reference
The mercurial repository change log and sources
Guides and Tutorials
Help for the total beginner
What to read first.
Suggestions for how to develop models
Important practical hints about model development for anyone who is starting to work with NEURON.
Construction and Use of Models: Part 1. Elementary Tools
A good beginner's tutorial. Introduces some of NEURON's basic GUI tools.
Import3D tutorials
Search
Google™ Custom Search
Search
Navigation
Blogs
Recent posts
Recent blog posts
NEURON courses at the CNS 2014 and SFN 2014 meetings
2014 NEURON

NEURON Hands-on Course

EXERCISES

Table of Contents



The NEURON Book

Nicholas T. Carnevale
and Michael L. Hines

CAMBRIDGE

Where does it fit in the overall modeling process?

What kind of models?

How is the program organized?

Methods.

Using:

- **Building**
- **Running**
- **Analysing**

Networks (Inhibitory Synchronization).

Published models: ModelDB.

Where does it fit in the overall modeling process?

Physical System

↓

Model

↓

Representation in NEURON

Create Representation

Investigate/Explore/Control/Use Representation

Physical System

Squid axon

Model

Hodgkin−Huxley cable equations

$$\frac{D}{4R_a} \cdot \frac{\partial^2 V}{\partial x^2} = C_m \frac{\partial V}{\partial t} + \overline{g}_{na} m^3 h \cdot (V - E_{na}) + \overline{g}_k n^4 \cdot (V - E_k) + g_l \cdot (V - E_l)$$

$$\frac{dm}{dt} = -\alpha_m m + \beta_m \cdot (1-m) \quad \alpha_m = \frac{.1(V+40)}{1-e^{-.1(V+40)}} \quad \beta_m = 4e^{-(V+65)/18}$$

$$\frac{dh}{dt} = -\alpha_h h + \beta_h \cdot (1-h) \quad \alpha_h = .07e^{-.05(V+65)} \quad \beta_h = \frac{1}{1+e^{-.1(V+35)}}$$

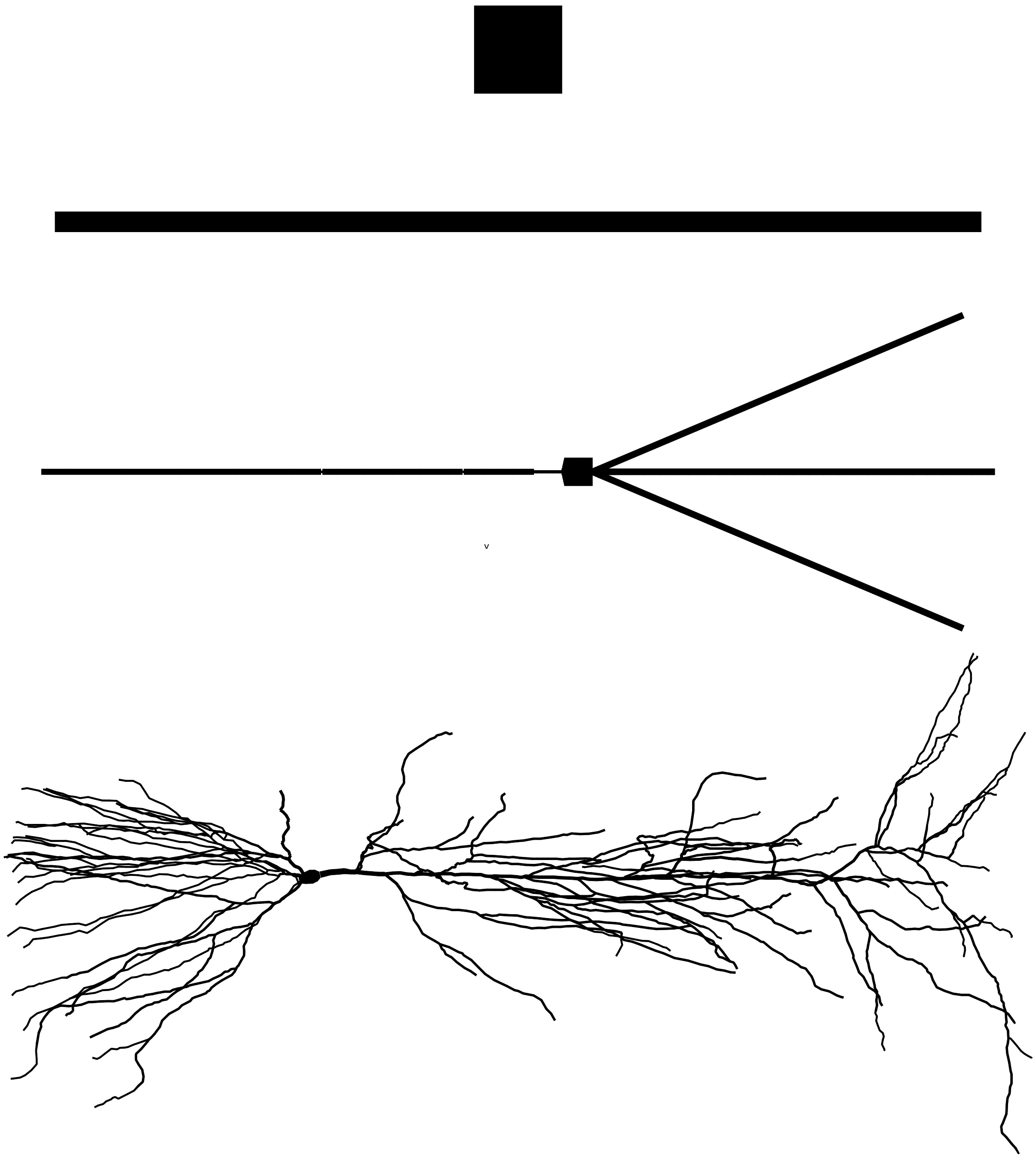
$$\frac{dn}{dt} = -\alpha_n n + \beta_n \cdot (1-n) \quad \alpha_n = \frac{.01(V+55)}{1-e^{-.1(V+55)}} \quad \beta_n = .125e^{-(V+65)/80}$$

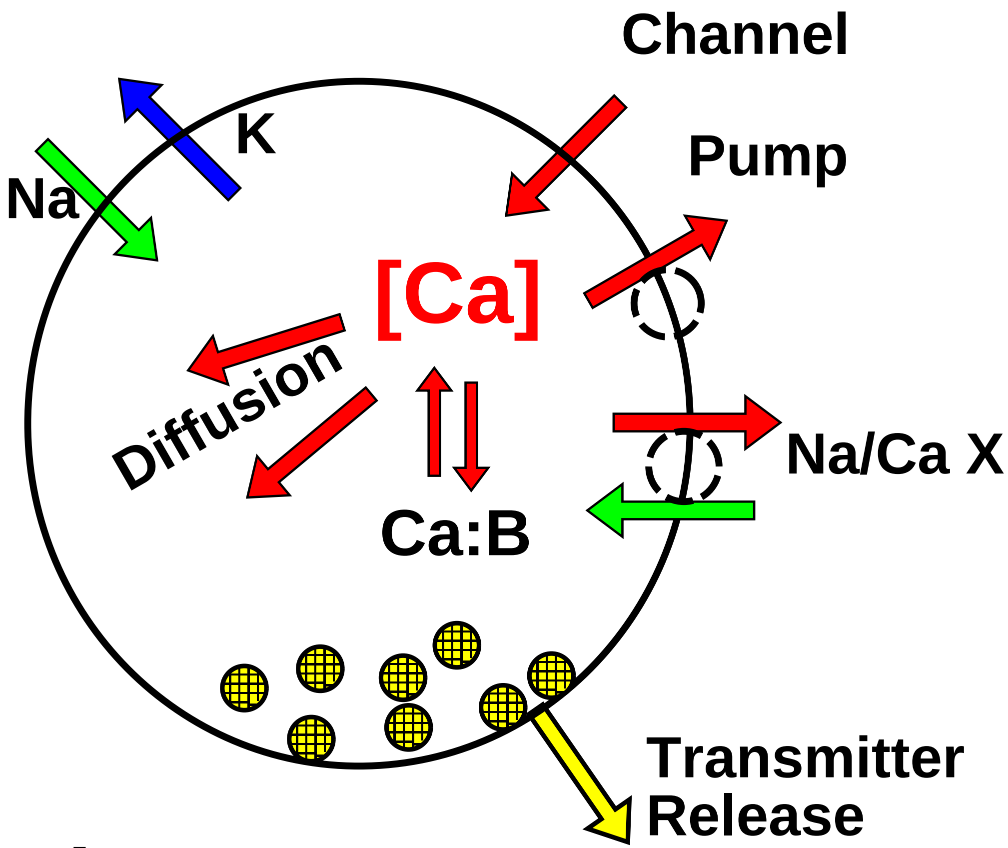
Simulation

NEURON representation

```
create axon
axon {
    nseg = 75
    diam = 100
    L = 20000
    insert hh
}
```

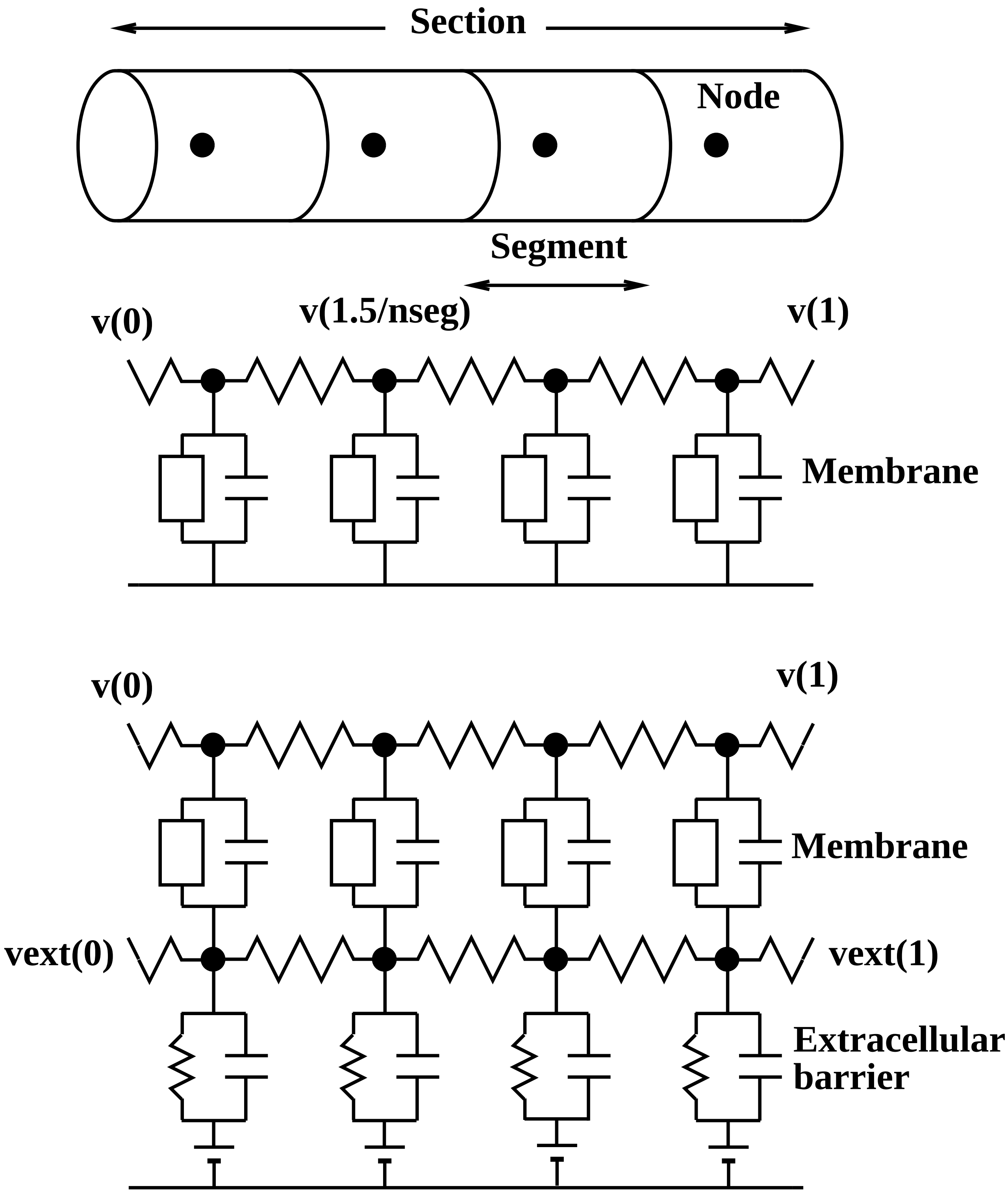
What kind of models?

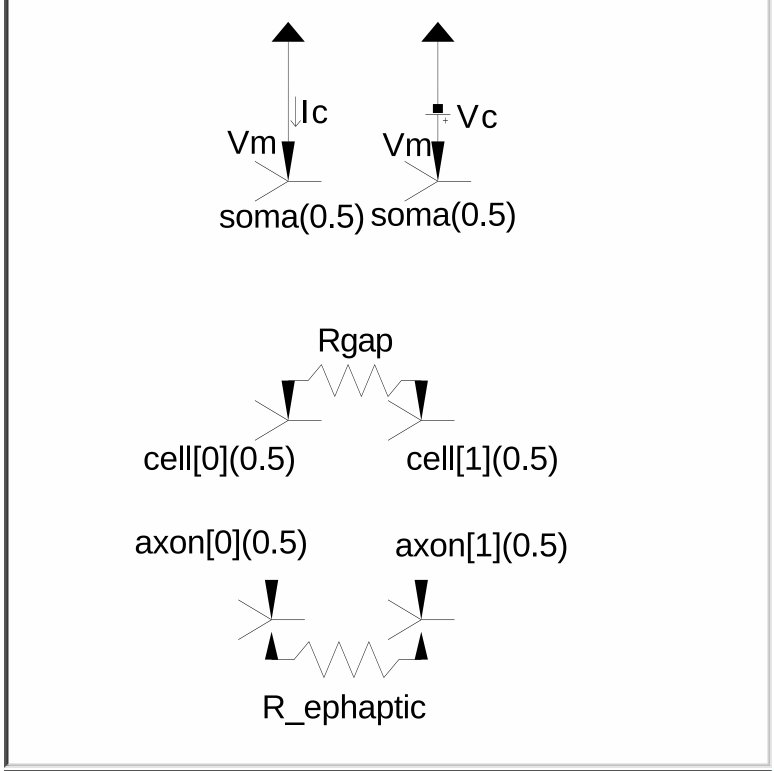


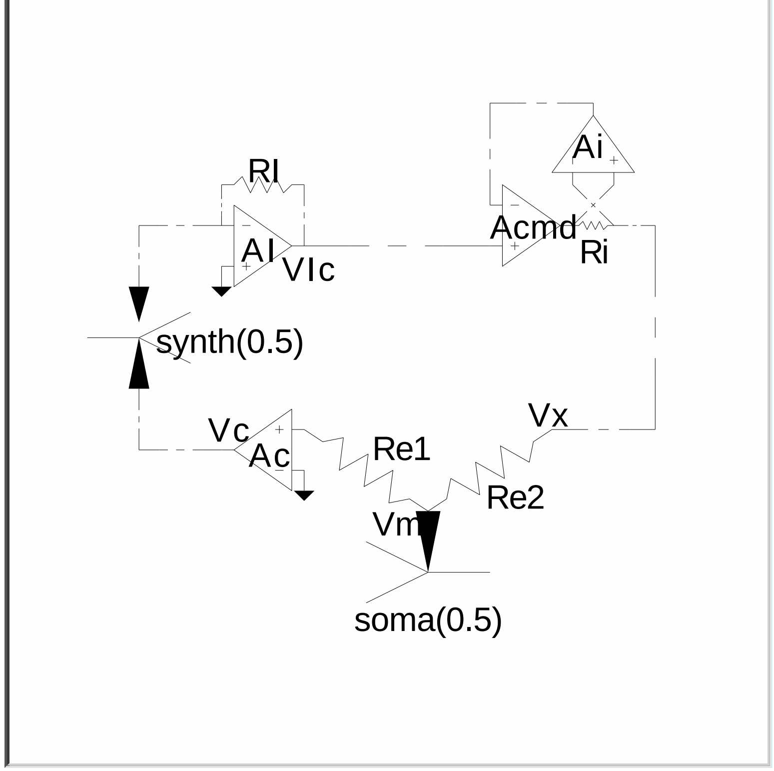


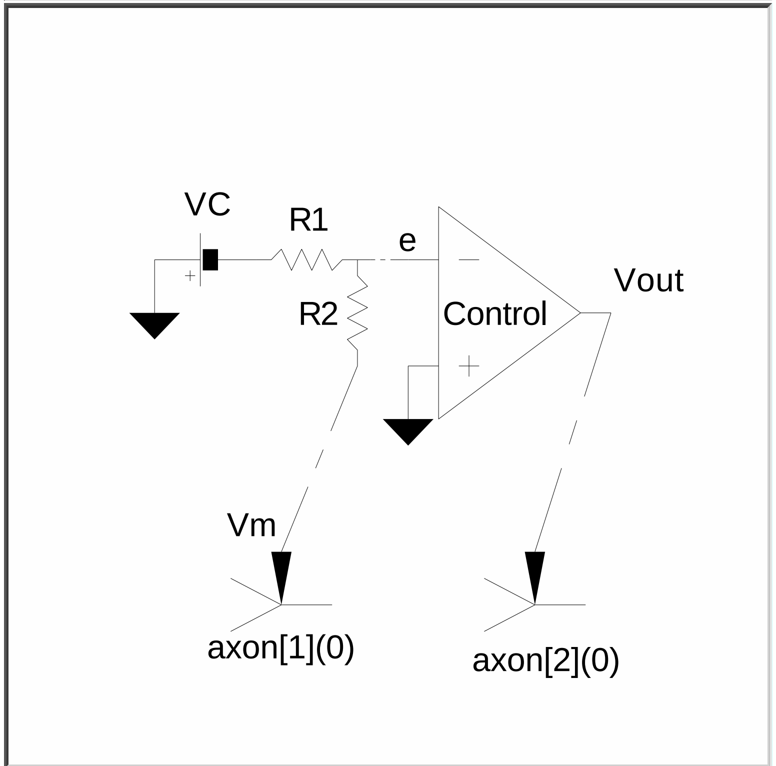
Single Channels
Extracellular fields
Linear circuits
Synapses
Networks

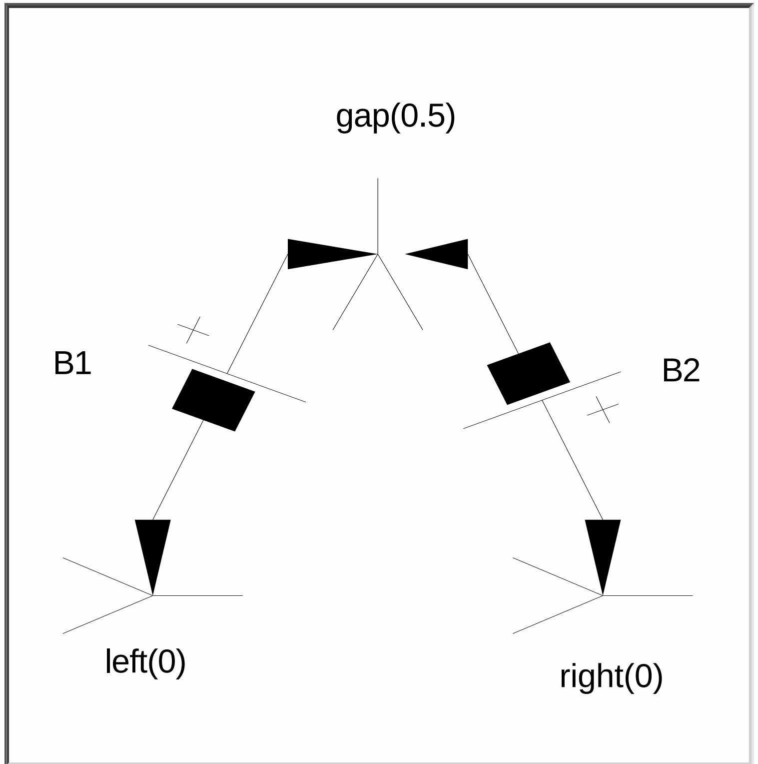
Discrete Event Simulation
Artificial Spiking Cells

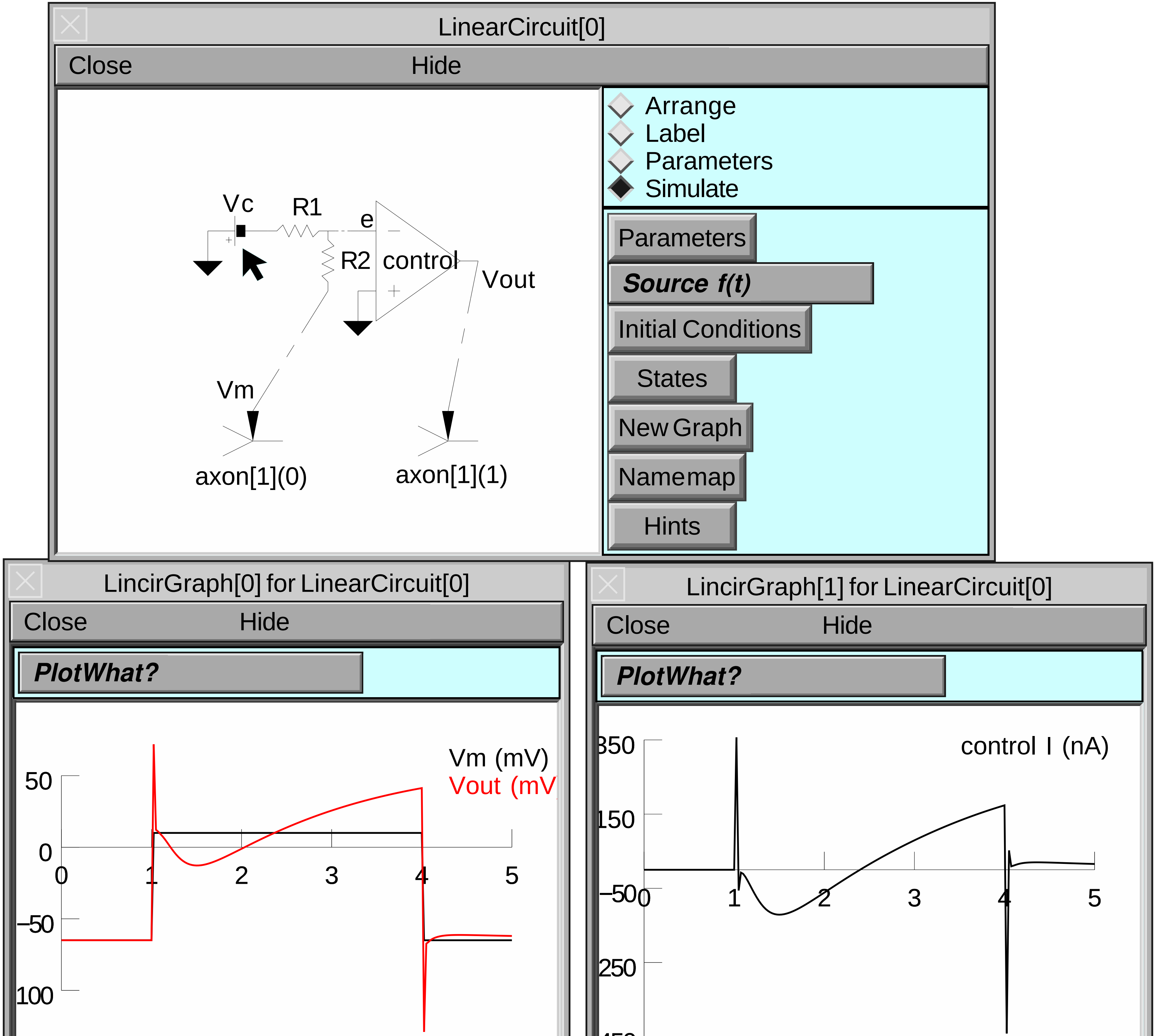

Section
Node
Segment
v(0)
v(1.5/nseg)
v(1)
Membrane
v(0)
v(1)
Membrane
vext(0)
vext(1)
Extracellular
barrier


Ic
Vc
Vm
Vm
soma(0.5)
soma(0.5)
Rgap
cell[0](0.5)
cell[1](0.5)
axon[0](0.5)
axon[1](0.5)
R_ephaptic

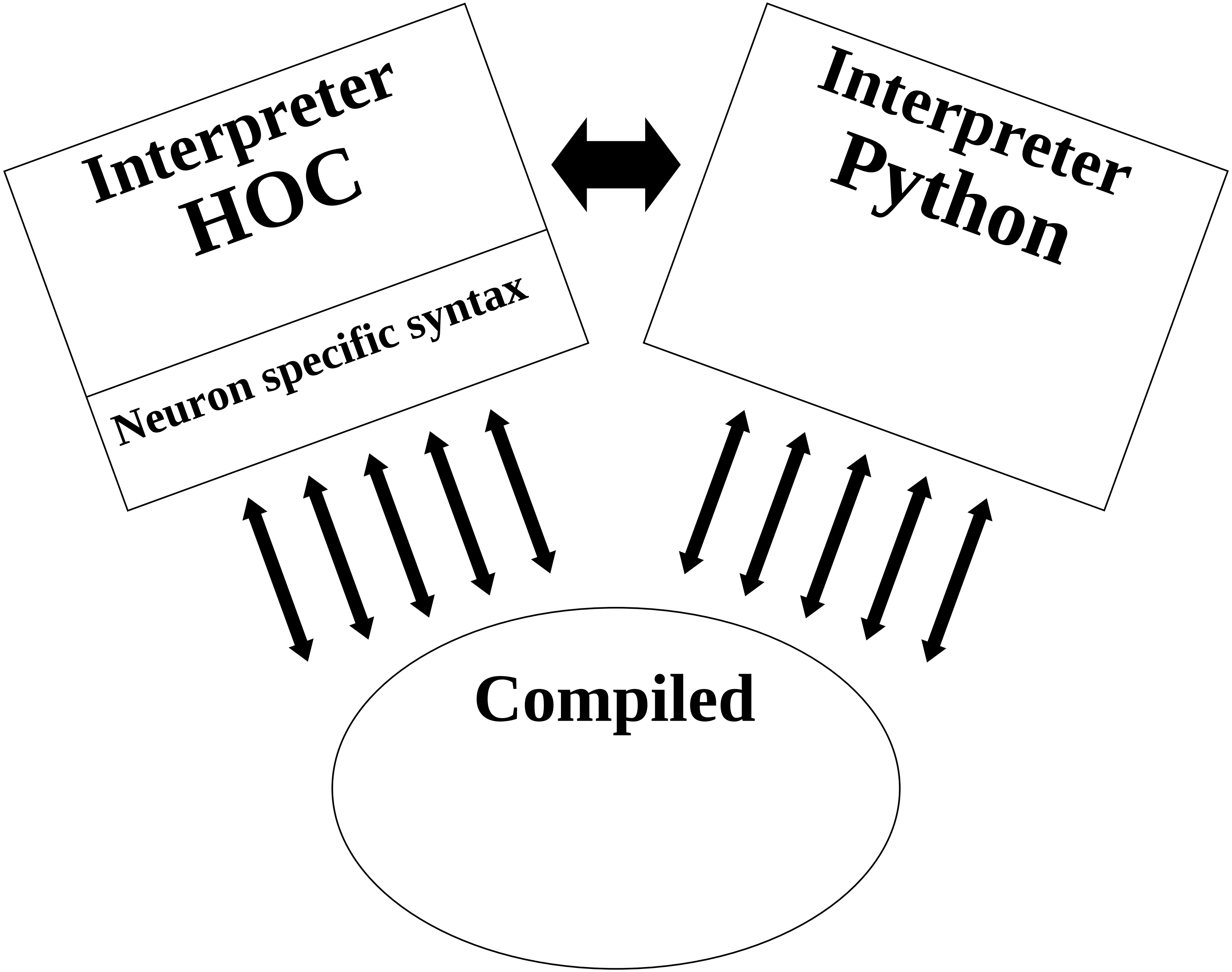

RI
Ai
AI
VIc
Acmd
Ri
synth(0.5)
Vc
Ac
Re1
Vx
Re2
Vm
soma(0.5)


VC
R1
e
R2
Control
Vout
Vm
axon[1](0)
axon[2](0)

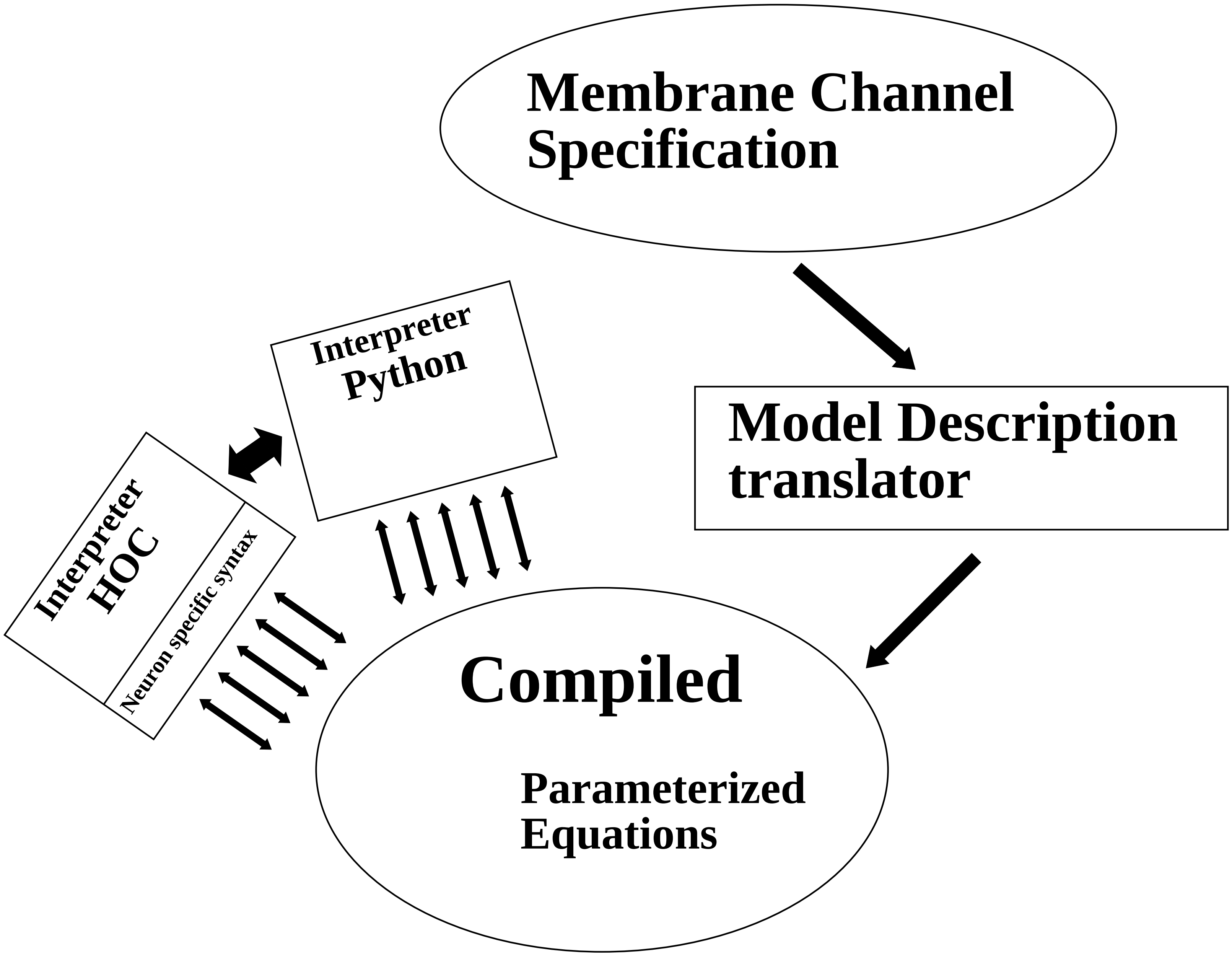

gap(0.5)
B1
B2
left(0)
right(0)

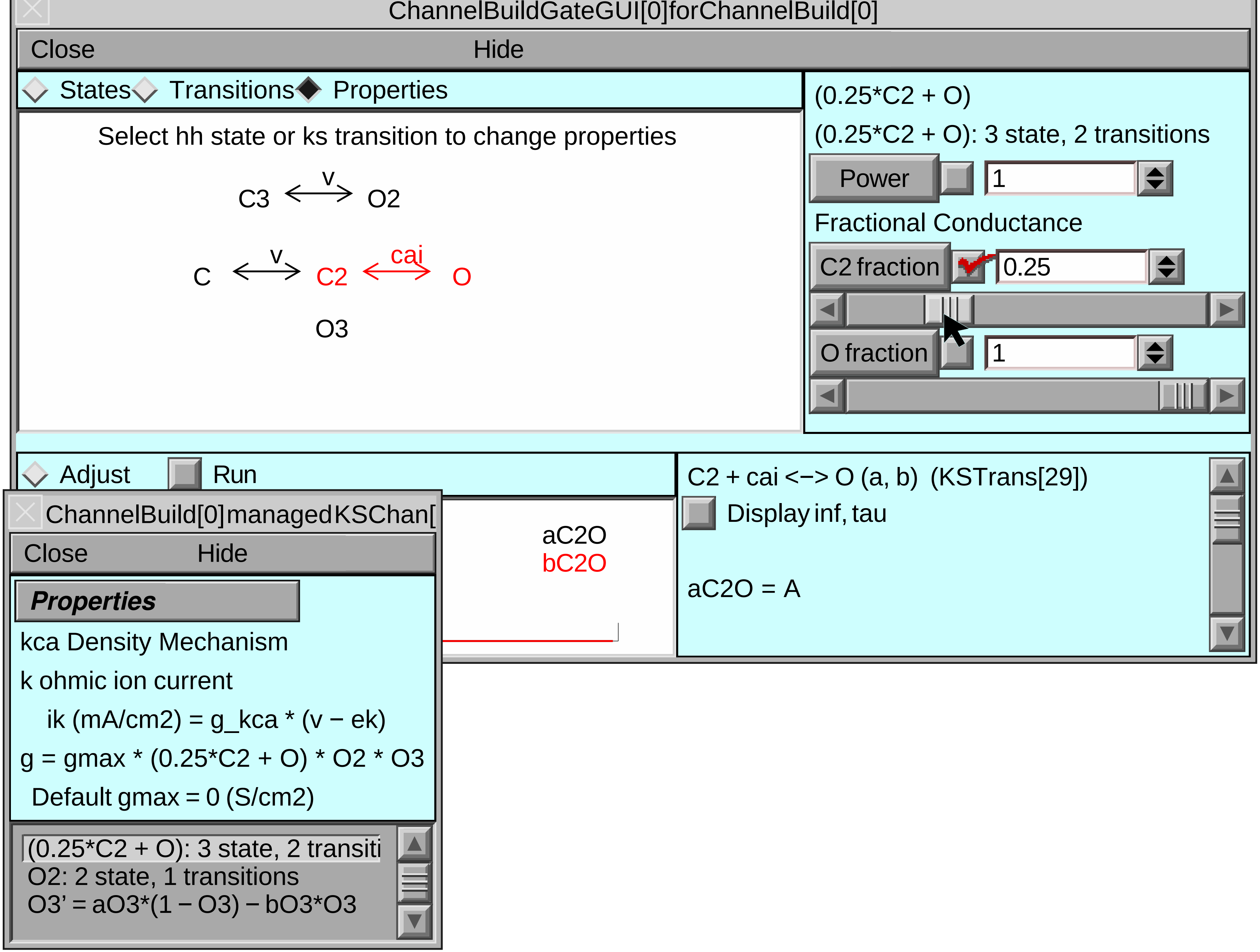

LinearCircuit[0]
Close
Hide
Vc
R1
e
R2
control
Vout
Vm
axon[1](0)
axon[1](1)
Arrange
Label
Parameters
Simulate
Parameters
Source f(t)
Initial Conditions
States
New Graph
Namemap
Hints
LincirGraph[0] for LinearCircuit[0]
Close
Hide
PlotWhat?
Vm (mV)
Vout (mV
50
0
–50
100
0
1
2
3
4
5
LincirGraph[1] for LinearCircuit[0]
Close
Hide
PlotWhat?
control I (nA)
350
150
–50
250
0
1
2
3
4
5

How is the program organized?

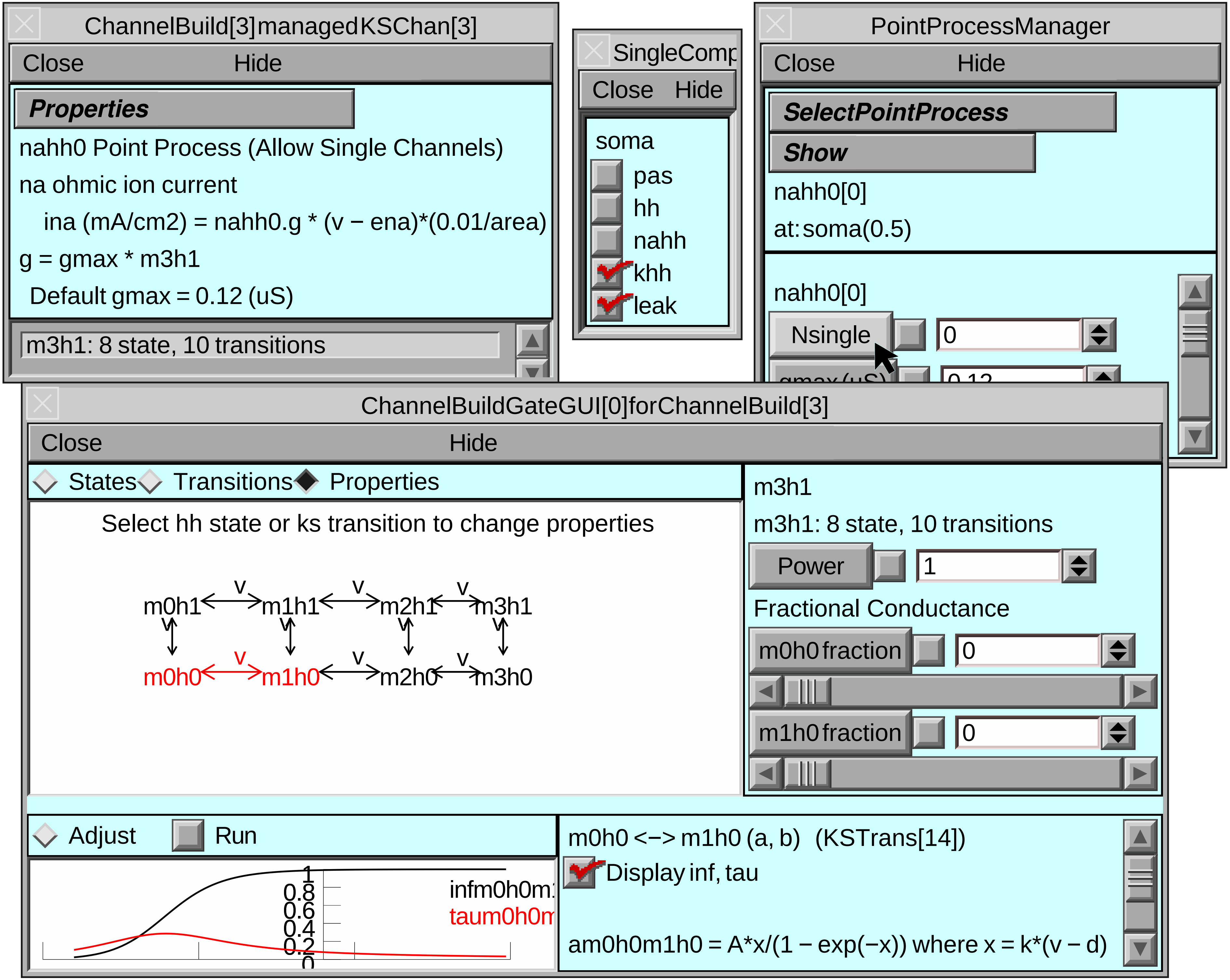


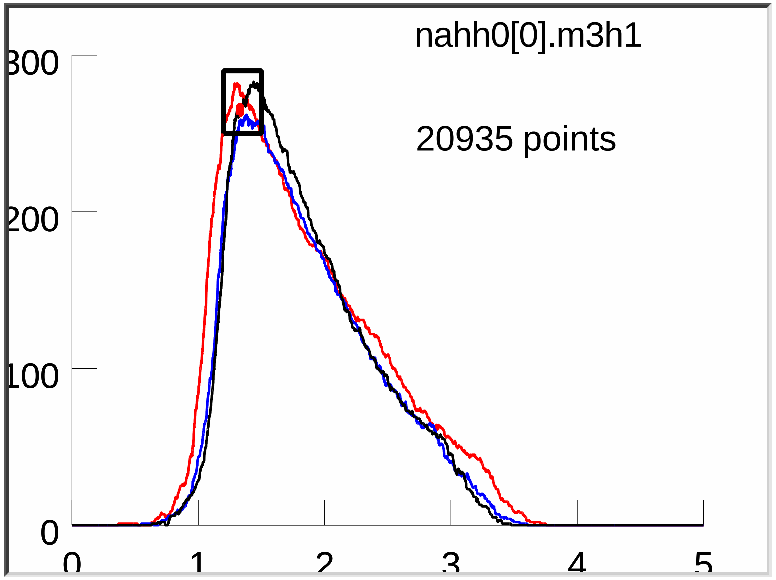
Interpreter
HOC
Neuron specific syntax
Interpreter
Python
Compiled

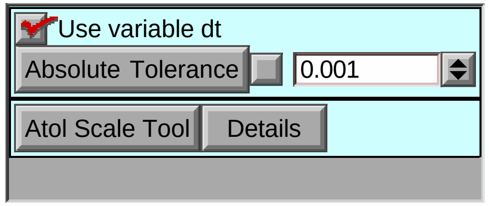

Membrane Channel
Specification
Model Description
translator
Interpreter
Python
Interpreter
HOC
Neuron specific syntax
Compiled
Parameterized
Equations

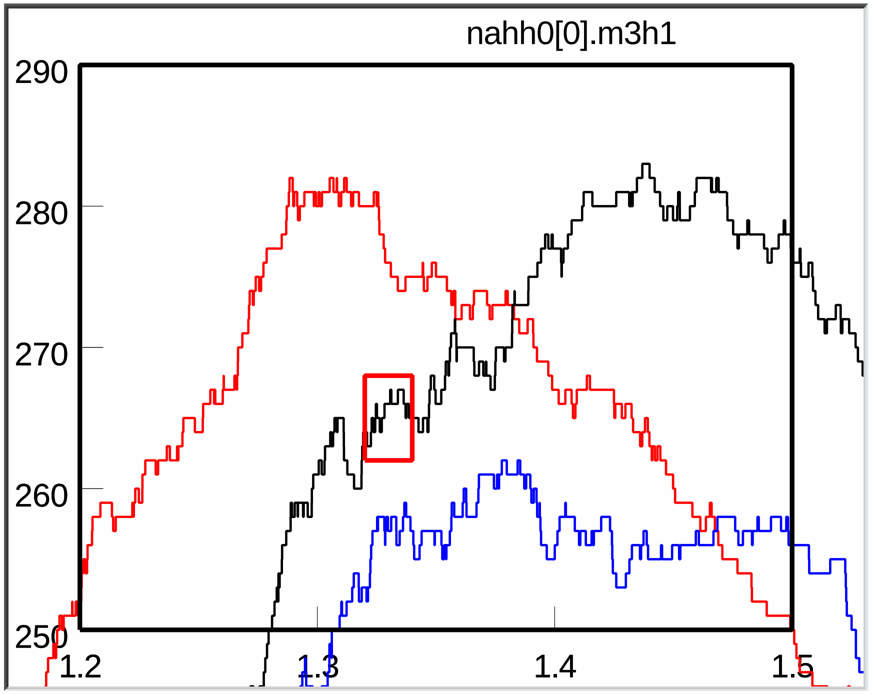


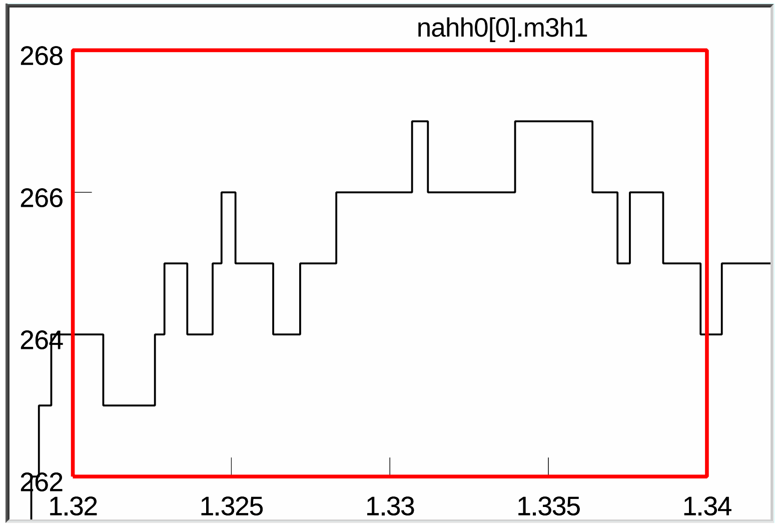
ChannelBuildGateGUI[0]forChannelBuild[0]
Close
Hide
States
Transitions
Properties
Select hh state or ks transition to change properties
C3
v
O2
C
v
C2
cai
O
O3
(0.25*C2 + O)
(0.25*C2 + O): 3 state, 2 transitions
Power
1
Fractional Conductance
C2 fraction
0.25
O fraction
1
Adjust
Run
aC2O
bC2O
C2 + cai <-> O (a, b) (KSTrans[29])
Display inf, tau
aC2O = A
ChannelBuild[0]managedKSChan[
Close
Hide
Properties
kca Density Mechanism
k ohmic ion current
ik (mA/cm2) = g_kca * (v − ek)
g = gmax * (0.25*C2 + O) * O2 * O3
Default gmax = 0 (S/cm2)
(0.25*C2 + O): 3 state, 2 transiti
O2: 2 state, 1 transitions
O3' = aO3*(1 − O3) − bO3*O3


ChannelBuild[3]managedKSChan[3]
Close
Hide
Properties
nahh0 Point Process (Allow Single Channels)
na ohmic ion current
ina (mA/cm2) = nahh0.g * (v − ena)*(0.01/area)
g = gmax * m3h1
Default gmax = 0.12 (uS)
m3h1: 8 state, 10 transitions
SingleComp
Close
Hide
soma
pas
hh
nahh
khh
leak
PointProcessManager
Close
Hide
SelectPointProcess
Show
nahh0[0]
at:soma(0.5)
nahh0[0]
Nsingle
0
ChannelBuildGateGUI[0]forChannelBuild[3]
Close
Hide
States
Transitions
Properties
Select hh state or ks transition to change properties
m0h1
m1h1
m2h1
m3h1
m0h0
m1h0
m2h0
m3h0
m3h1
m3h1: 8 state, 10 transitions
Power
1
Fractional Conductance
m0h0 fraction
0
m1h0 fraction
0
Adjust
Run
m0h0 <−> m1h0 (a, b) (KSTrans[14])
Display inf, tau
am0h0m1h0 = A*x/(1 − exp(−x)) where x = k*(v − d)

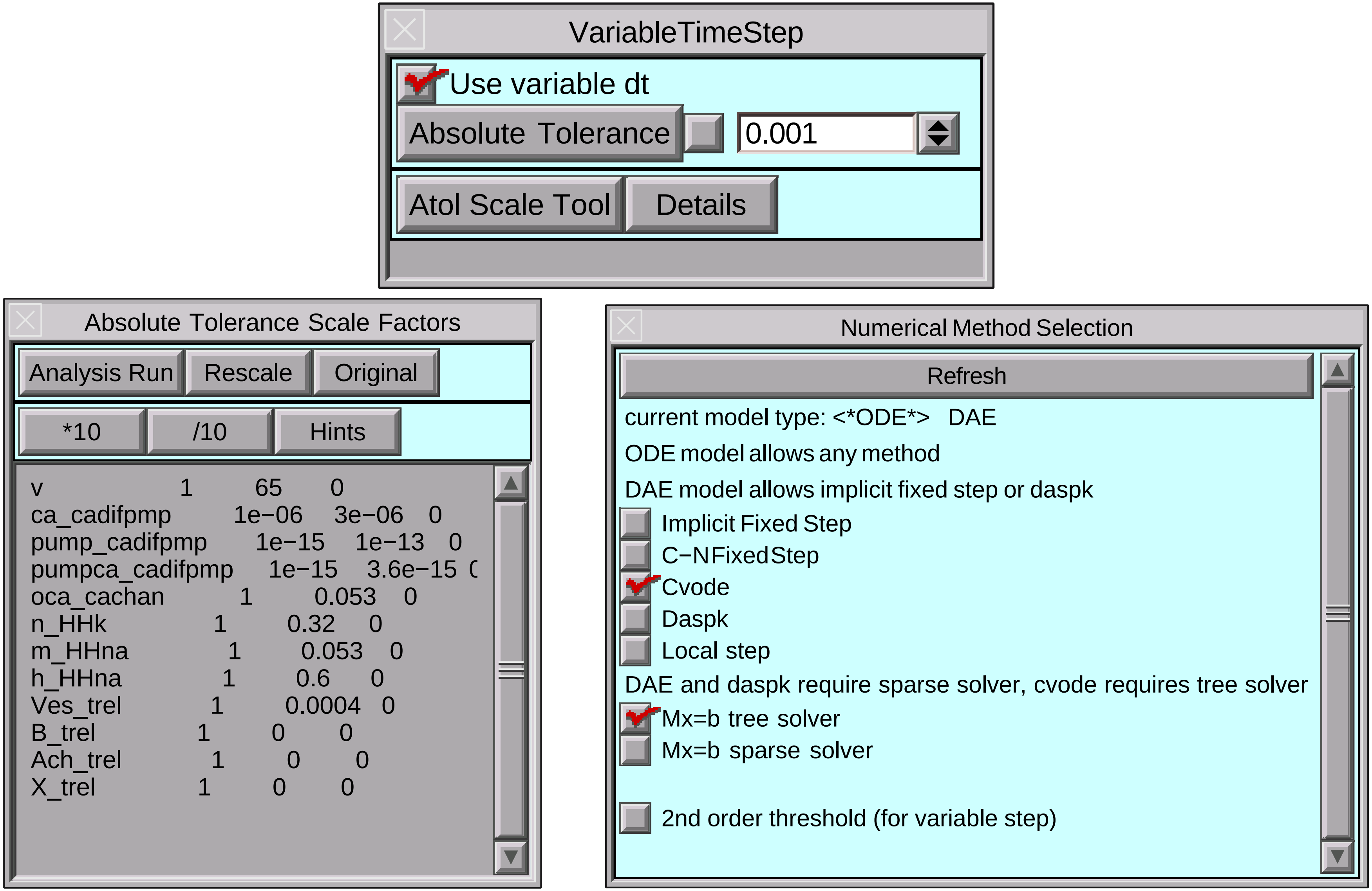

nahh0[0].m3h1
20935 points
300
200
100
0
0
1
2
3
4
5


Use variable dt
Absolute Tolerance
0.001
Atol Scale Tool
Details

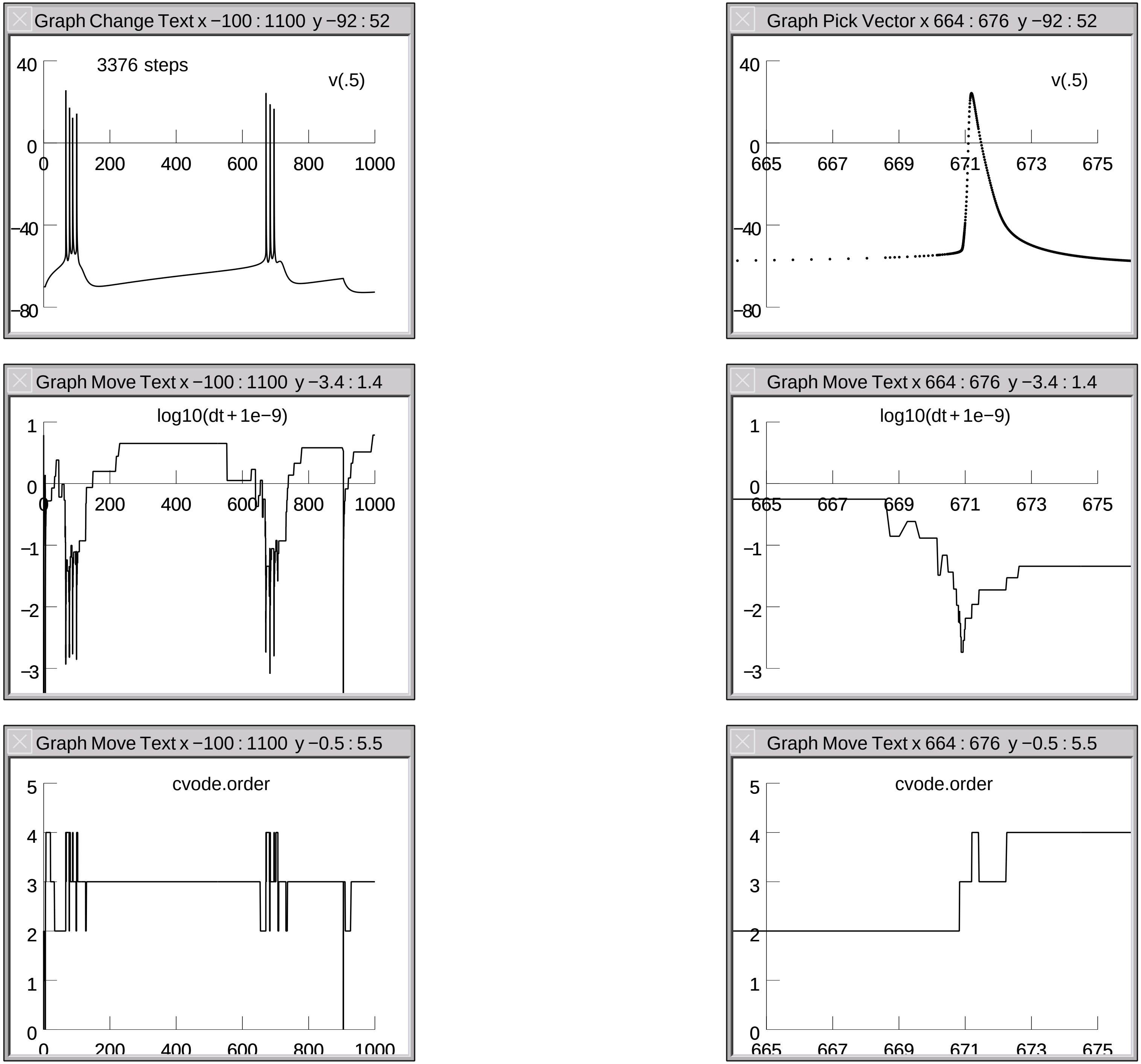

nahh0[0].m3h1
290
280
270
260
250
1.2
1.3
1.4
1.5

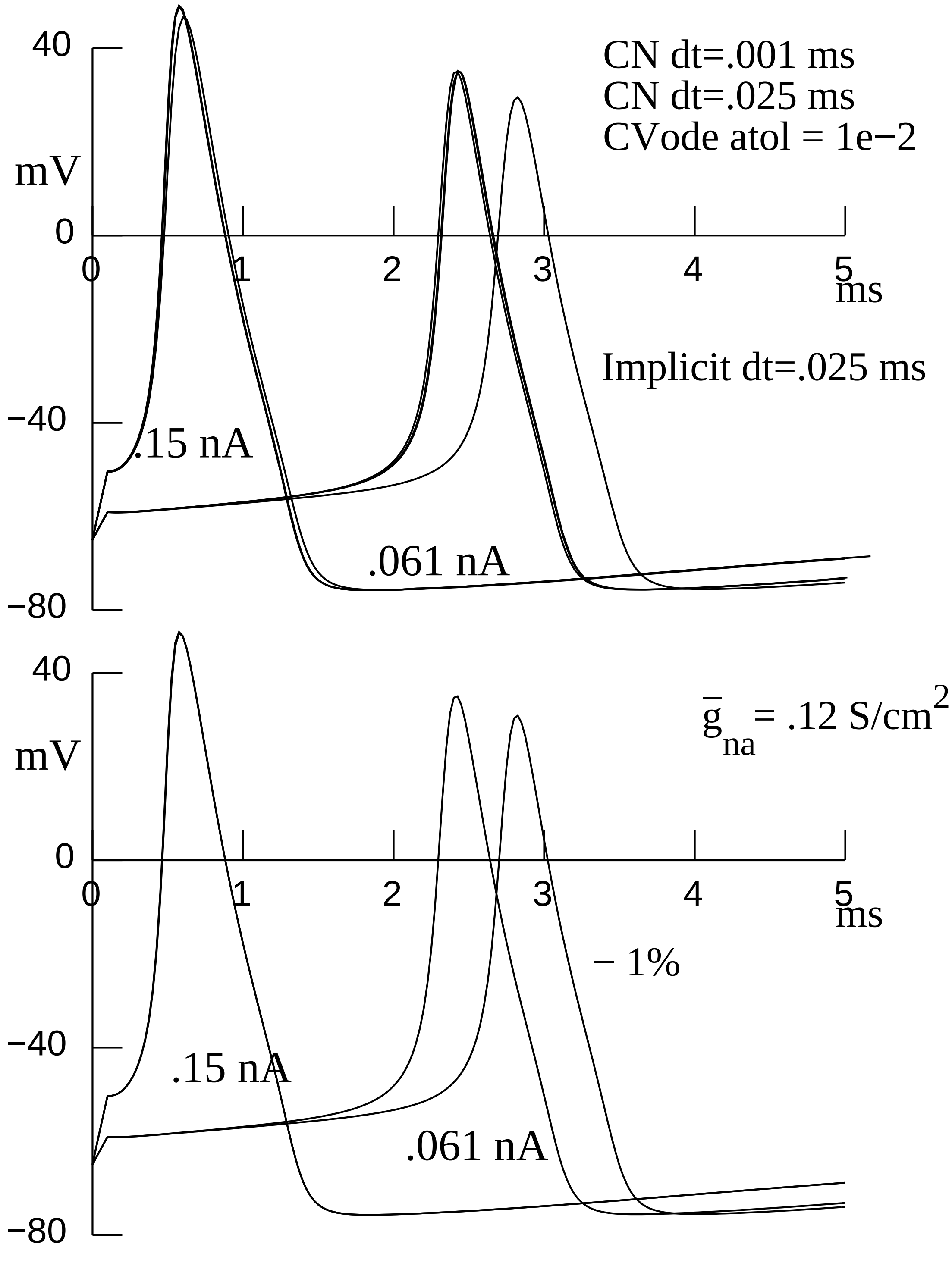

nahh0[0].m3h1
268
266
264
262
1.32
1.325
1.33
1.335
1.34

Methods.

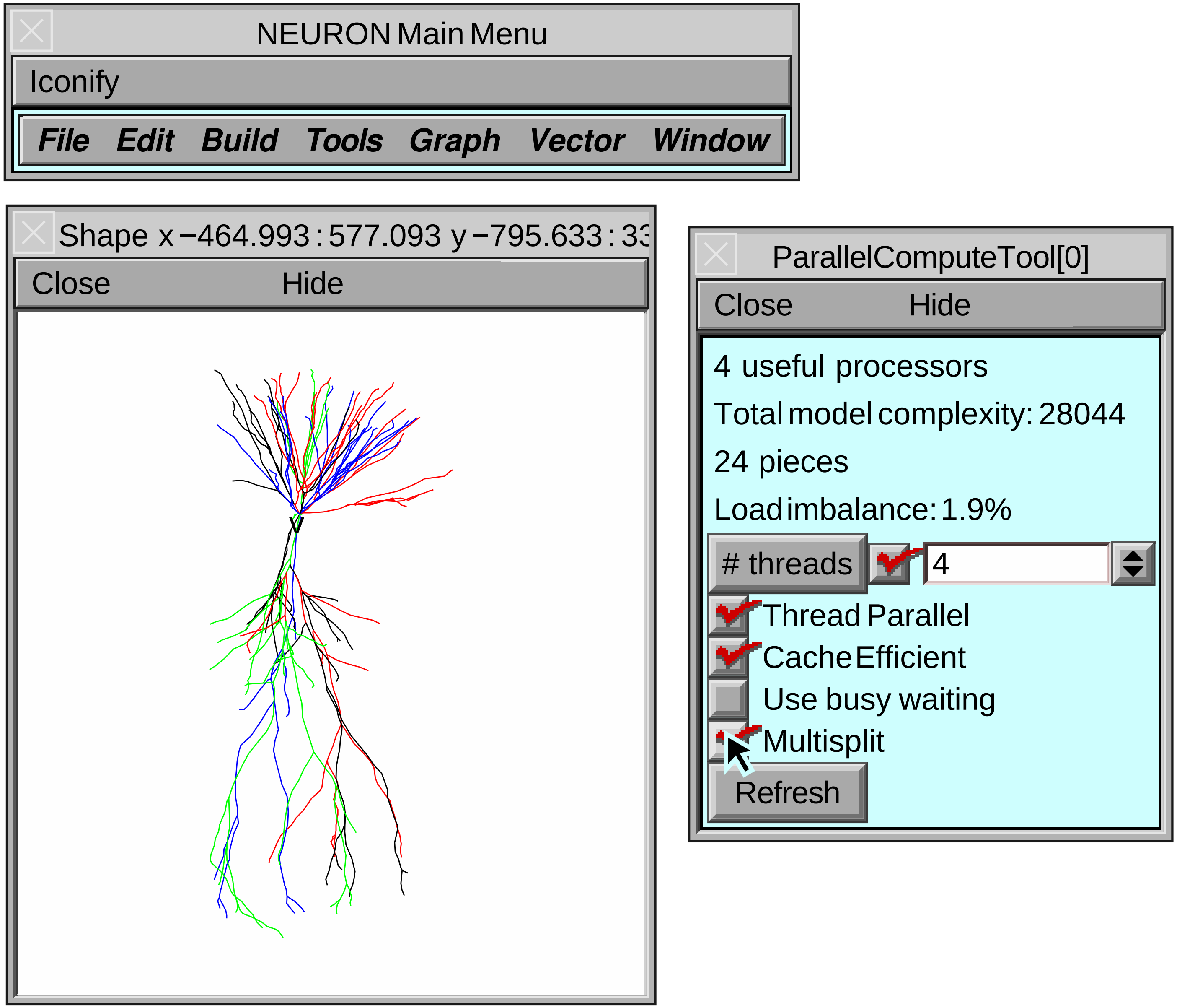


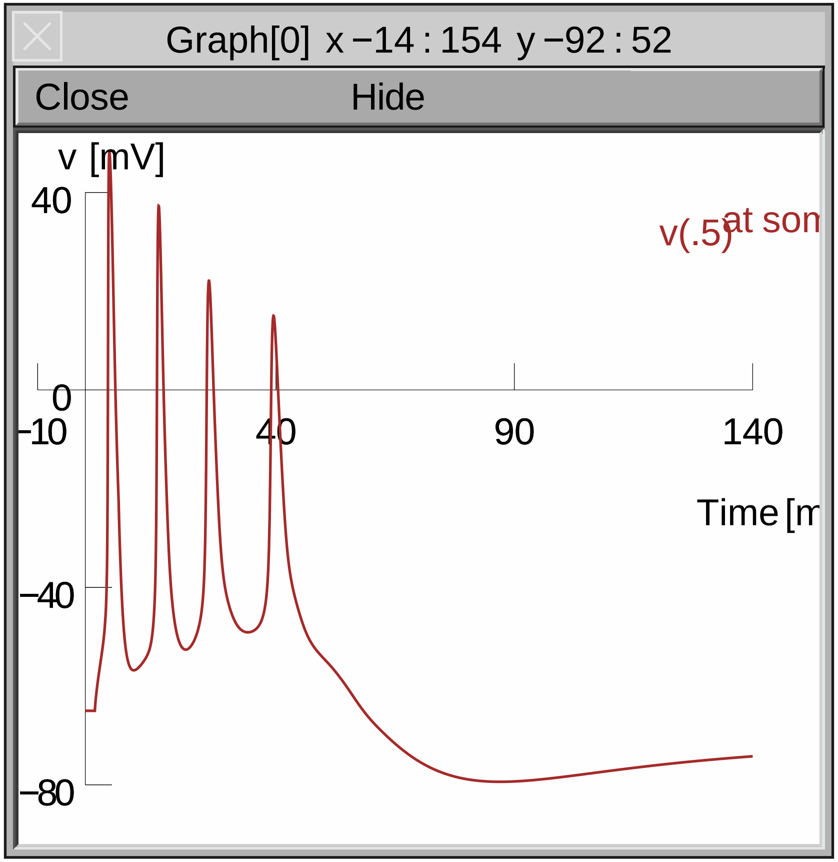
VariableTimeStep
Use variable dt
Absolute Tolerance
0.001
Atol Scale Tool
Details
Absolute Tolerance Scale Factors
Analysis Run
Rescale
Original
*10
/10
Hints
v 1 65 0
ca_cadifpmp 1e−06 3e−06 0
pump_cadifpmp 1e−15 1e−13 0
pumpca_cadifpmp 1e−15 3.6e−15
oca_cachan 1 0.053 0
n_HHk 1 0.32 0
m_HHna 1 0.053 0
h_HHna 1 0.6 0
Ves_trel 1 0.0004 0
B_trel 1 0 0
Ach_trel 1 0 0
X_trel 1 0 0
Numerical Method Selection
Refresh
current model type: <*ODE*> DAE
ODE model allows any method
DAE model allows implicit fixed step or daspk
Implicit Fixed Step
C−N Fixed Step
Cvode
Daspk
Local step
DAE and daspk require sparse solver, cvode requires tree solver
Mx=b tree solver
Mx=b sparse solver
2nd order threshold (for variable step)

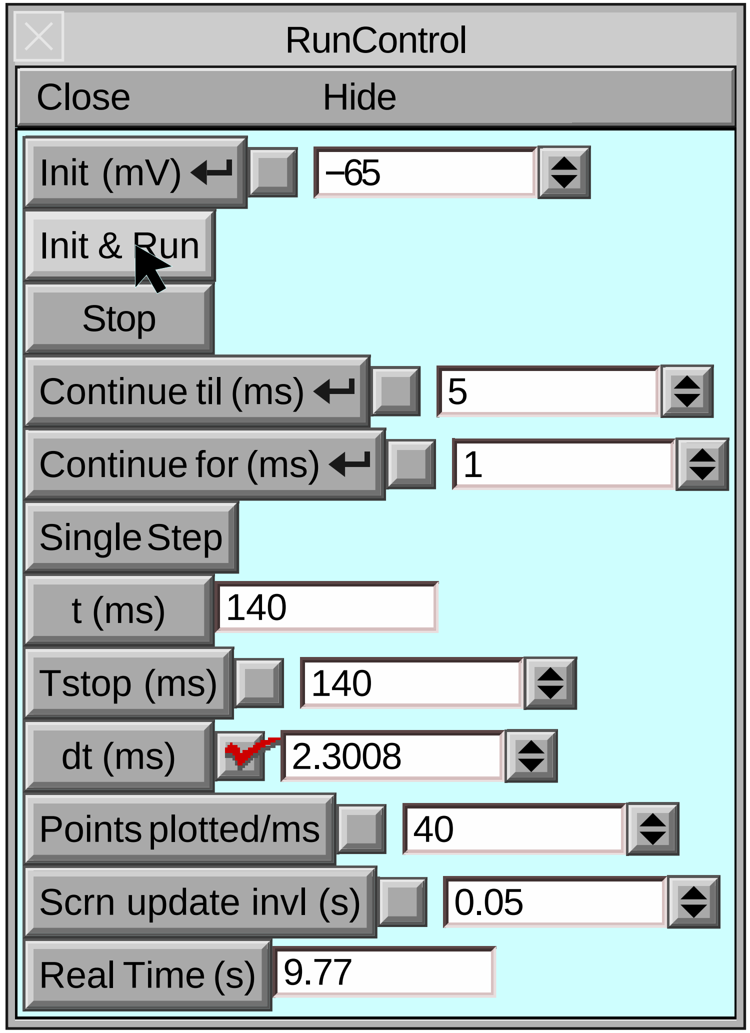

Graph Change Text x −100 : 1100 y −92 : 52
3376 steps
v(.5)
Graph Pick Vector x 664 : 676 y −92 : 52
v(.5)
Graph Move Text x −100 : 1100 y −3.4 : 1.4
log10(dt + 1e−9)
Graph Move Text x 664 : 676 y −3.4 : 1.4
log10(dt + 1e−9)
Graph Move Text x −100 : 1100 y −0.5 : 5.5
cvode.order
Graph Move Text x 664 : 676 y −0.5 : 5.5
cvode.order



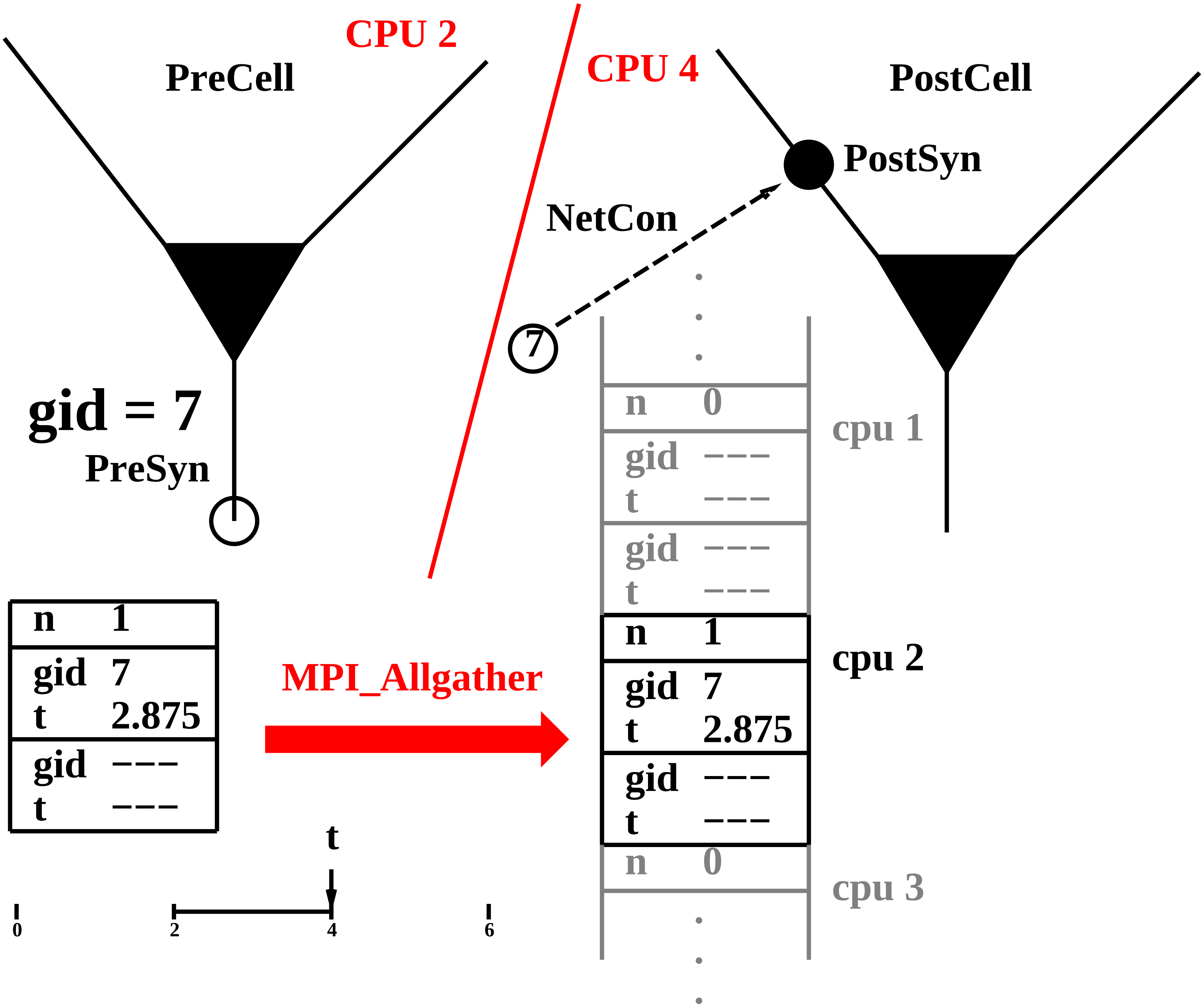
CN dt=.001 ms
CN dt=.025 ms
CVode atol = 1e−2
Implicit dt=.025 ms
mV
ms
.15 nA
.061 nA
40
0
−40
−80
0
1
2
3
4
5
$\bar{g}_{na}$ = .12 S/cm^2
− 1%


NEURON Main Menu
Iconify
File
Edit
Build
Tools
Graph
Vector
Window
Shape x −464.993 : 577.093 y −795.633 : 33
Close
Hide
ParallelComputeTool[0]
Close
Hide
4 useful processors
Total model complexity: 28044
24 pieces
Load imbalance: 1.9%
threads
4
Thread Parallel
Cache Efficient
Use busy waiting
Multisplit
Refresh



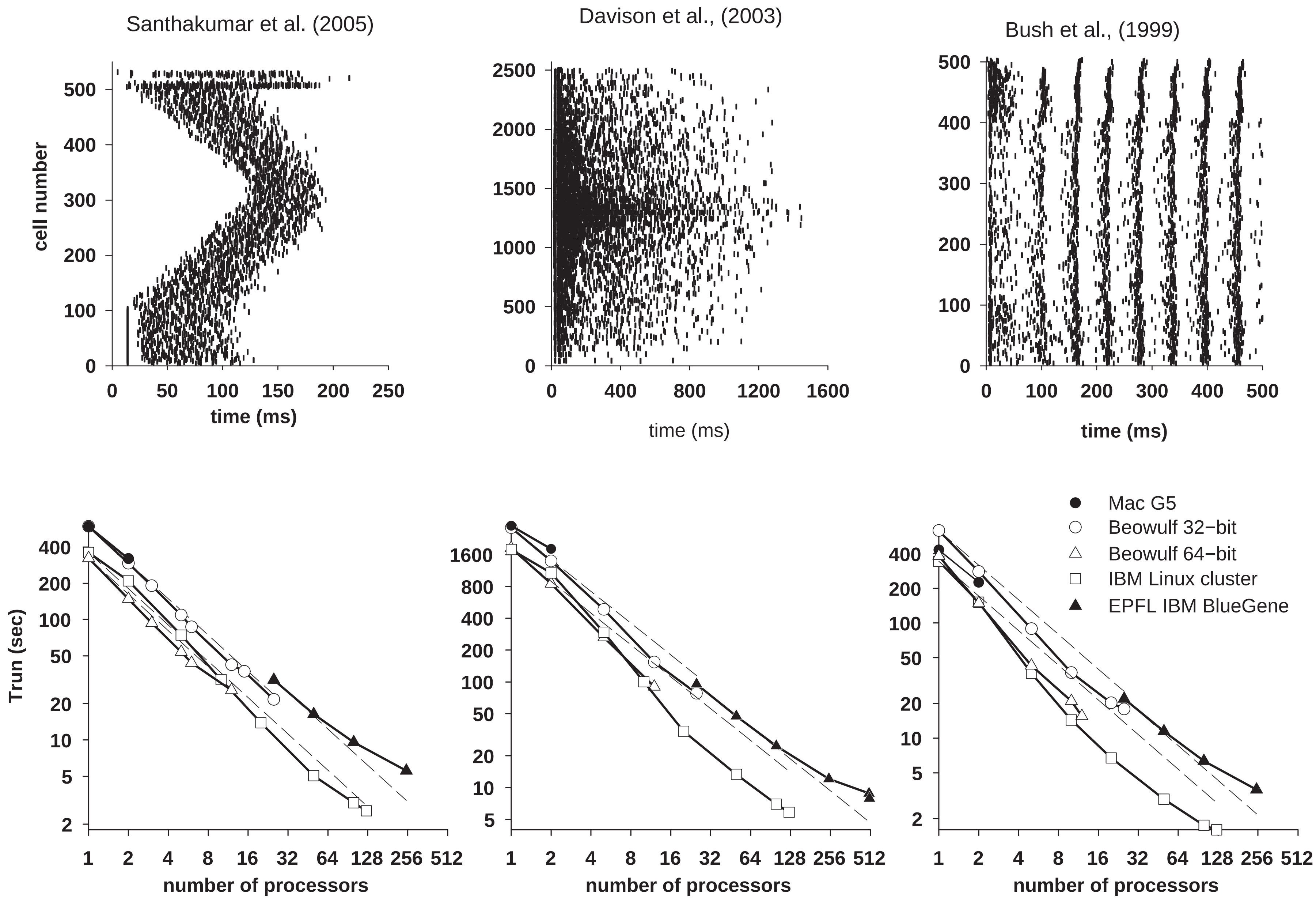


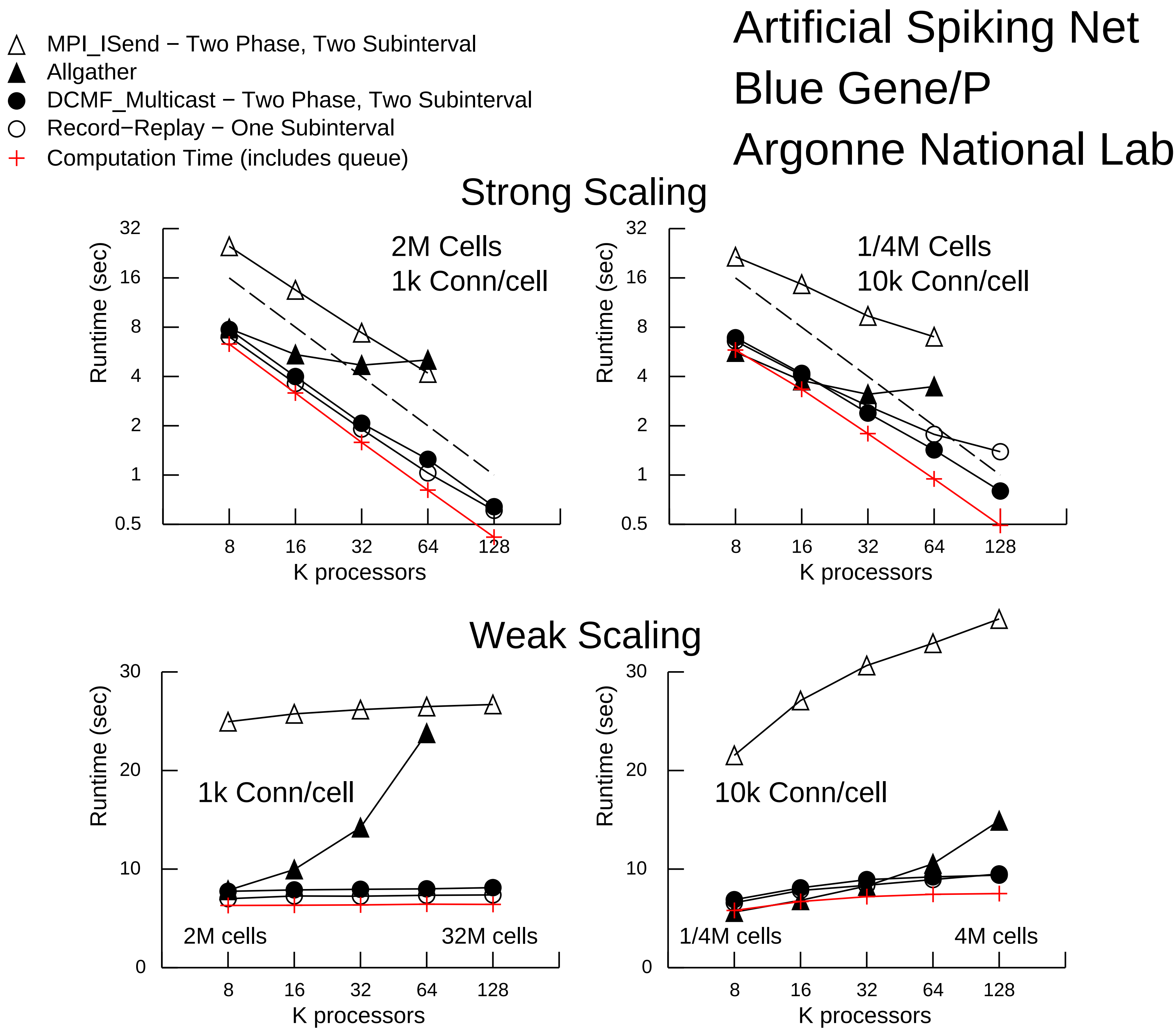
instead of 35.4s



CPU 2
PreCell
CPU 4
PostCell
PostSyn
NetCon
7
gid = 7
PreSyn
n 1
gid 7
t 2.875
gid ---
t ---
MPI_Allgather
n 0
cpu 1
gid ---
t ---
gid ---
t ---
n 1
cpu 2
gid 7
t 2.875
gid ---
t ---
n 0
cpu 3
t
0
2
4
6

Migliore et al (2006) J. Comput. Neurosci. 21(2):119




Artificial Spiking Net
Blue Gene/P
Argonne National Lab
MPI_ISend − Two Phase, Two Subinterval
Allgather
DCMF_Multicast − Two Phase, Two Subinterval
Record−Replay − One Subinterval
Computation Time (includes queue)
Strong Scaling
2M Cells
1k Conn/cell
1/4M Cells
10k Conn/cell
Runtime (sec)
32
16
8
4
2
1
0.5
8
16
32
64
128
K processors
Weak Scaling
1k Conn/cell
10k Conn/cell
30
20
10
0
2M cells
32M cells
1/4M cells
4M cells

Using:

- **Building**
- **Running**
- **Analysing**

Building a Cell

Shape

- Draw stylized
- Import 3−d reconstructions

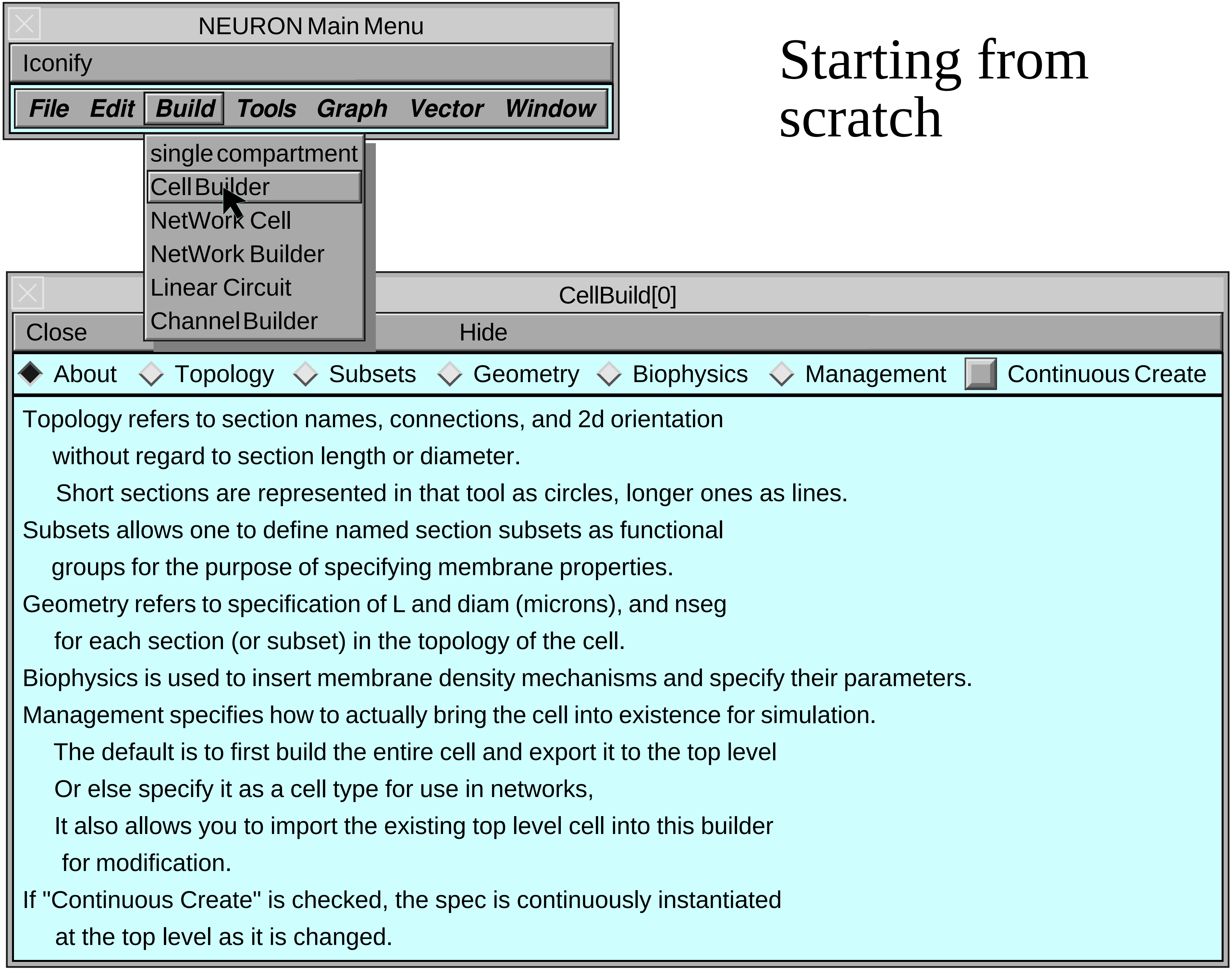
Channel Distribution

- Density
  - Homogeneous
  - Inhomogeneous
- ...over...
  - Whole cell
  - Regions
  - Individual sections

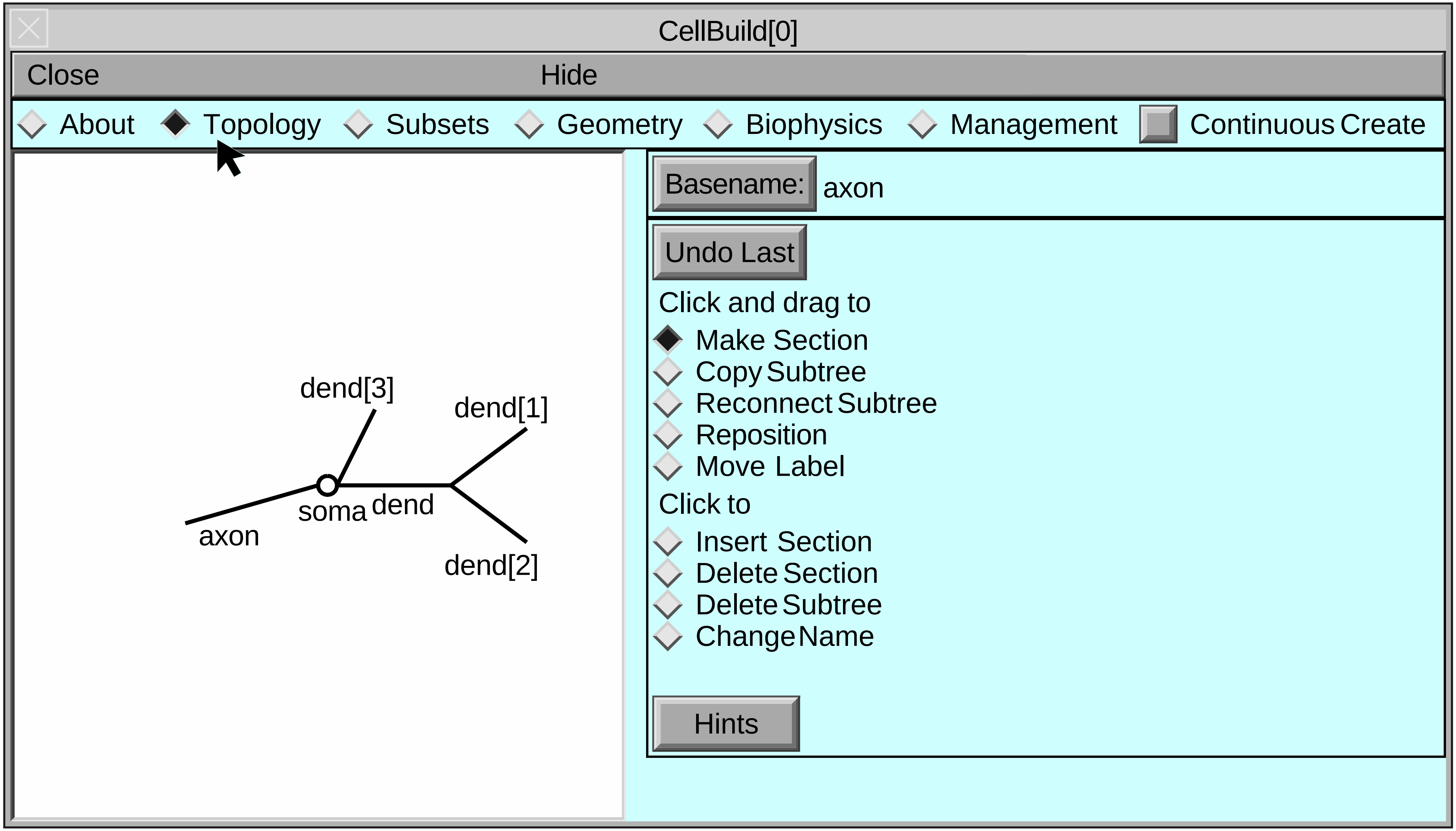
Creation

- Single cell
- Class for use in Network

Starting from scratch

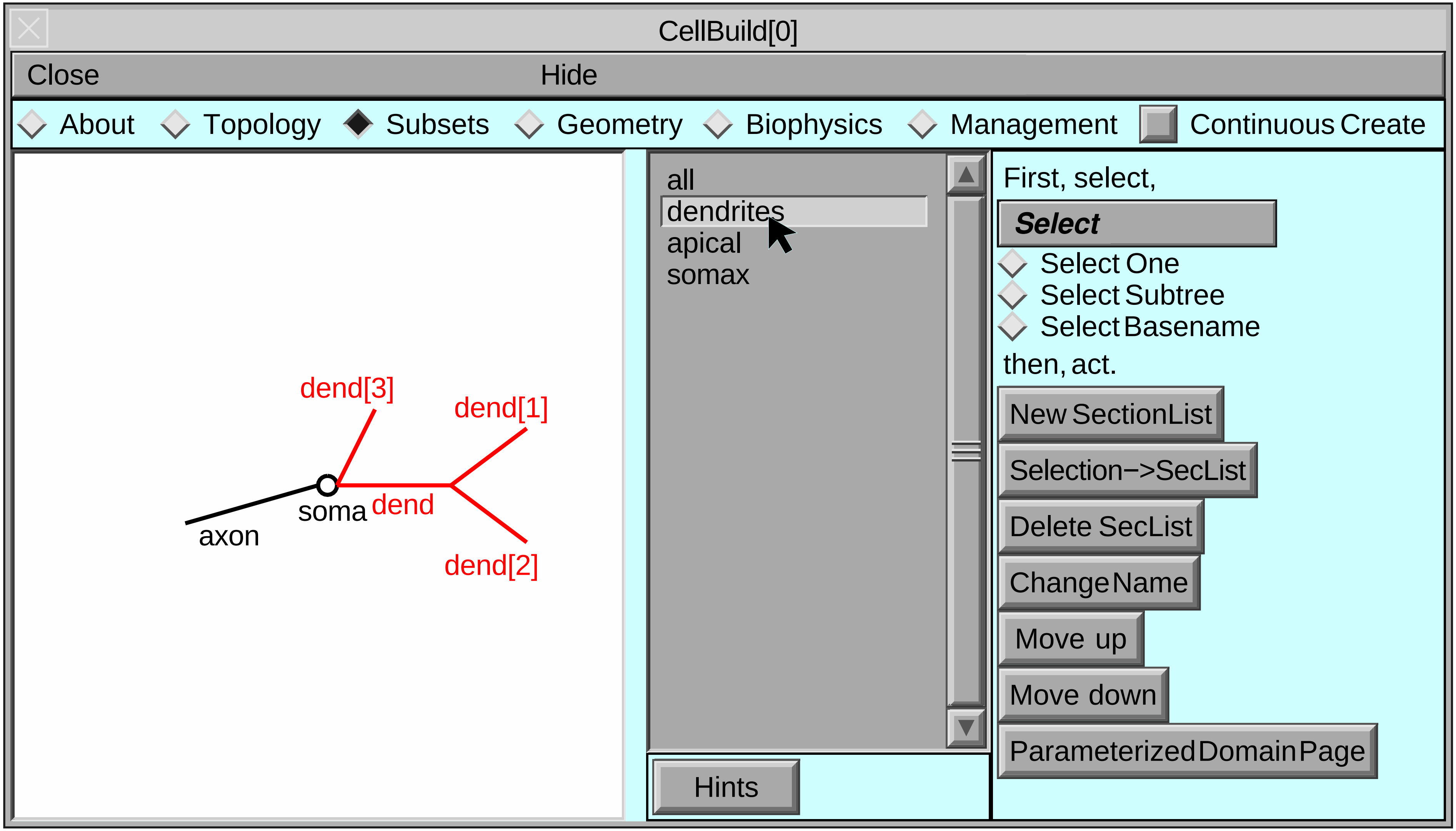


Topology defines section names and connectivity.

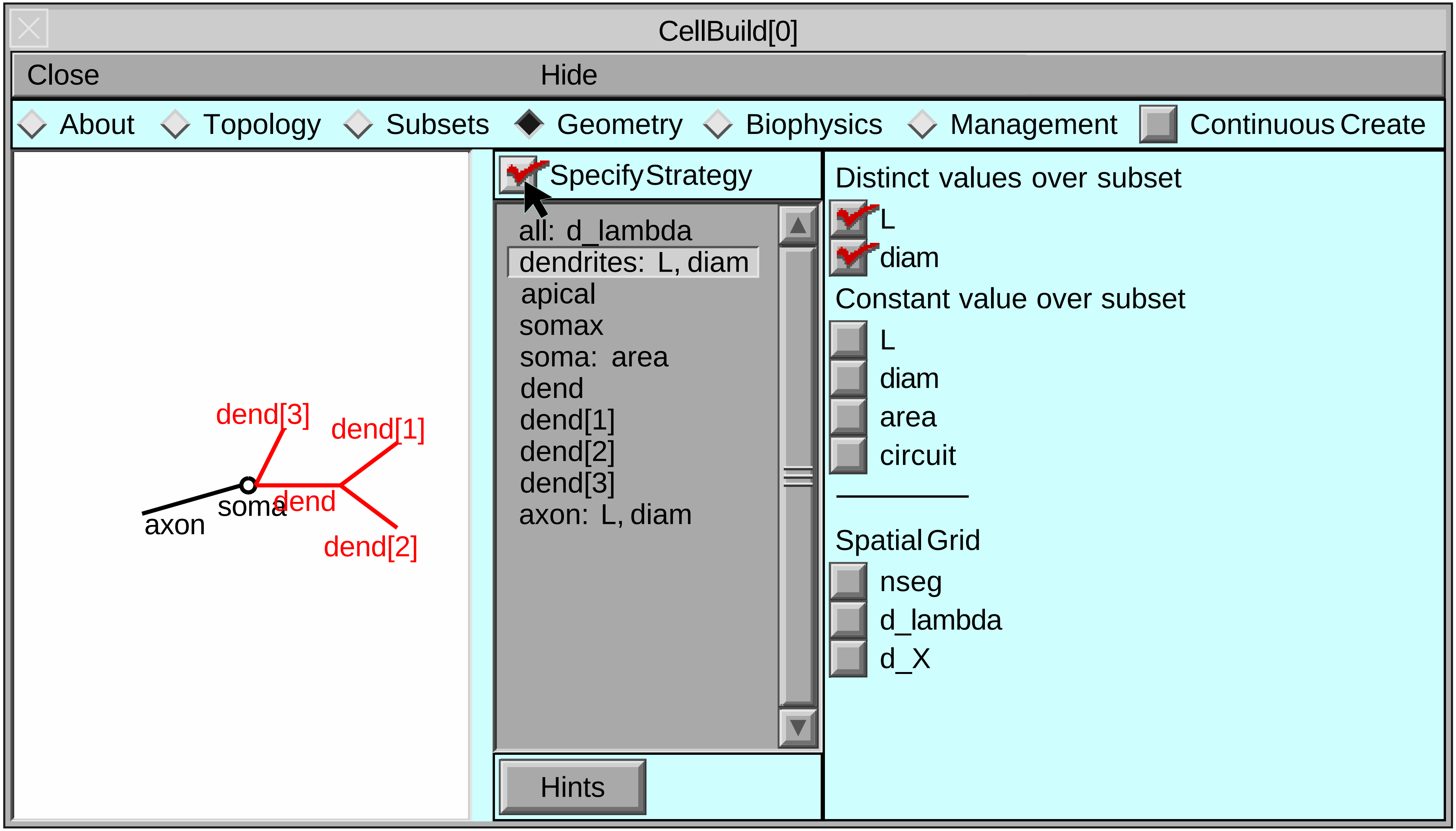


Cell subsets help to concisely specify membrane properties.

Constant and Inhomogenous distributions



Geometry is where Length, Diameter, and spatial discretization are specified.

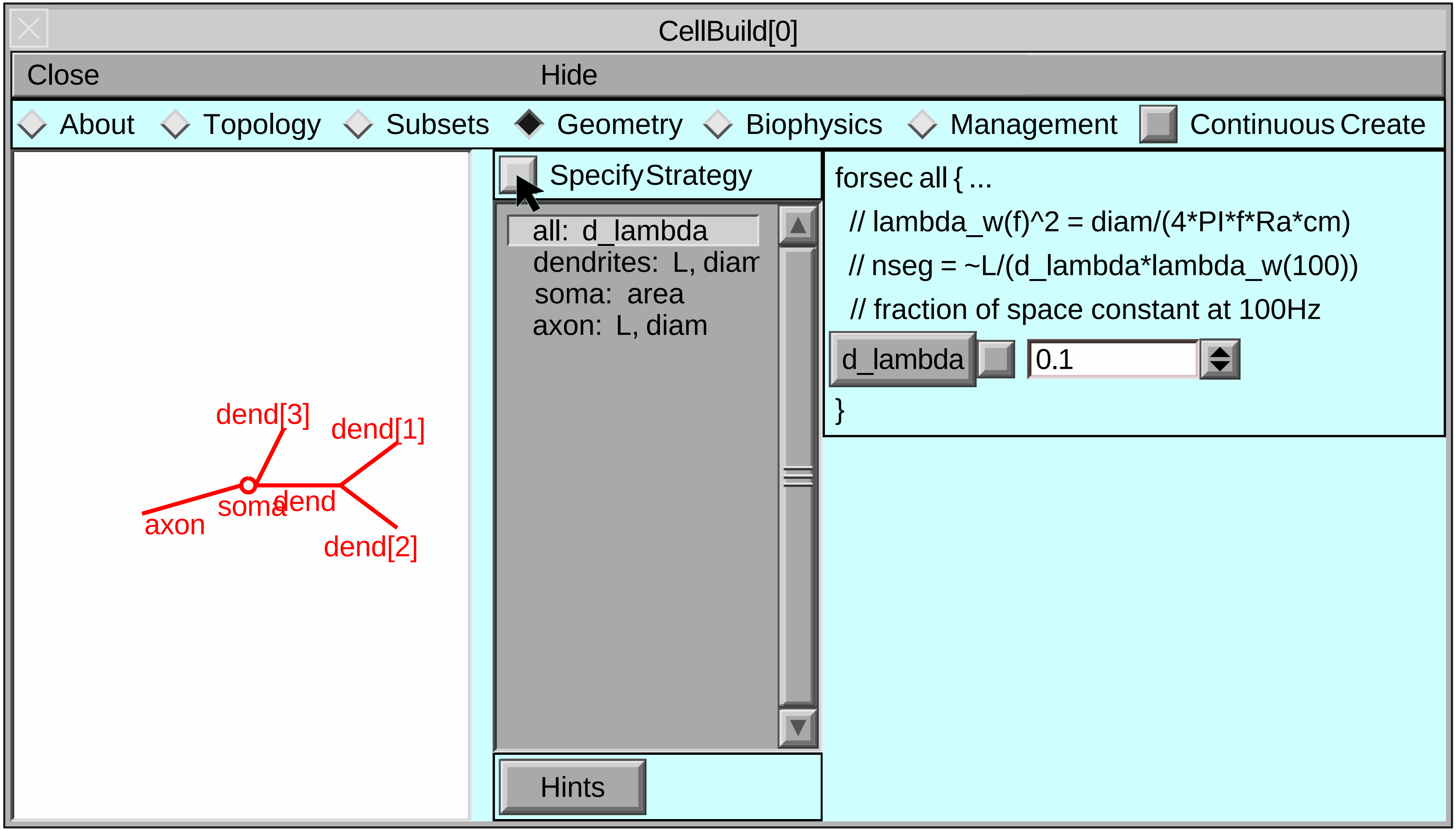


Good strategy

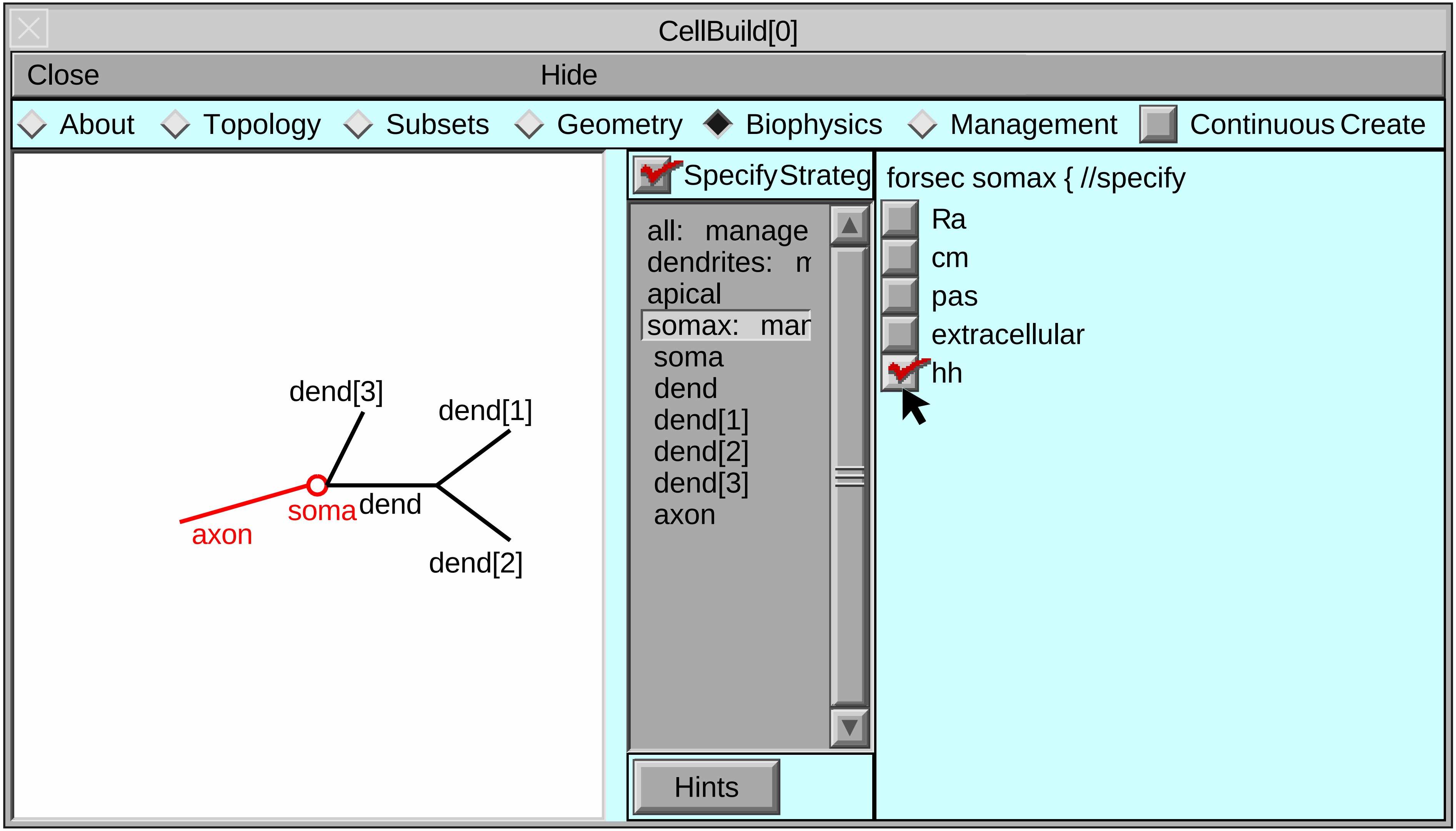
Concise Spec

```
oc>topology()
|-|       soma(0-1)
   `----|       dend[0](0-1)
         `----|       dend[1](0-1)
         `----|       dend[2](0-1)
   `--|       dend[3](0-1)
 `------|       axon(0-1)
```

Note: Compartmentalization based on medium frequency space constant is always a good idea.

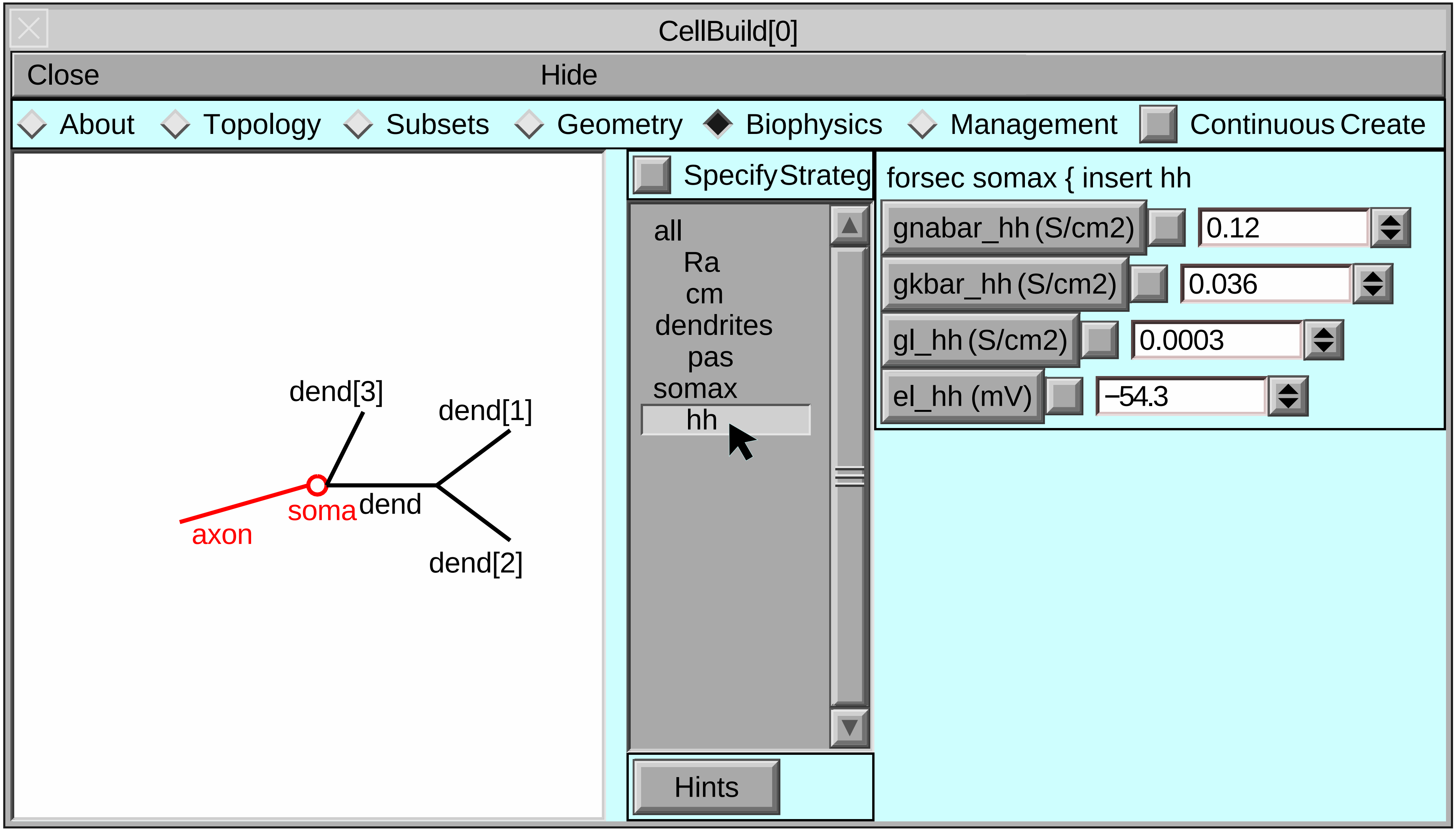


Sprinkling channels onto the cell begins with a strategy.



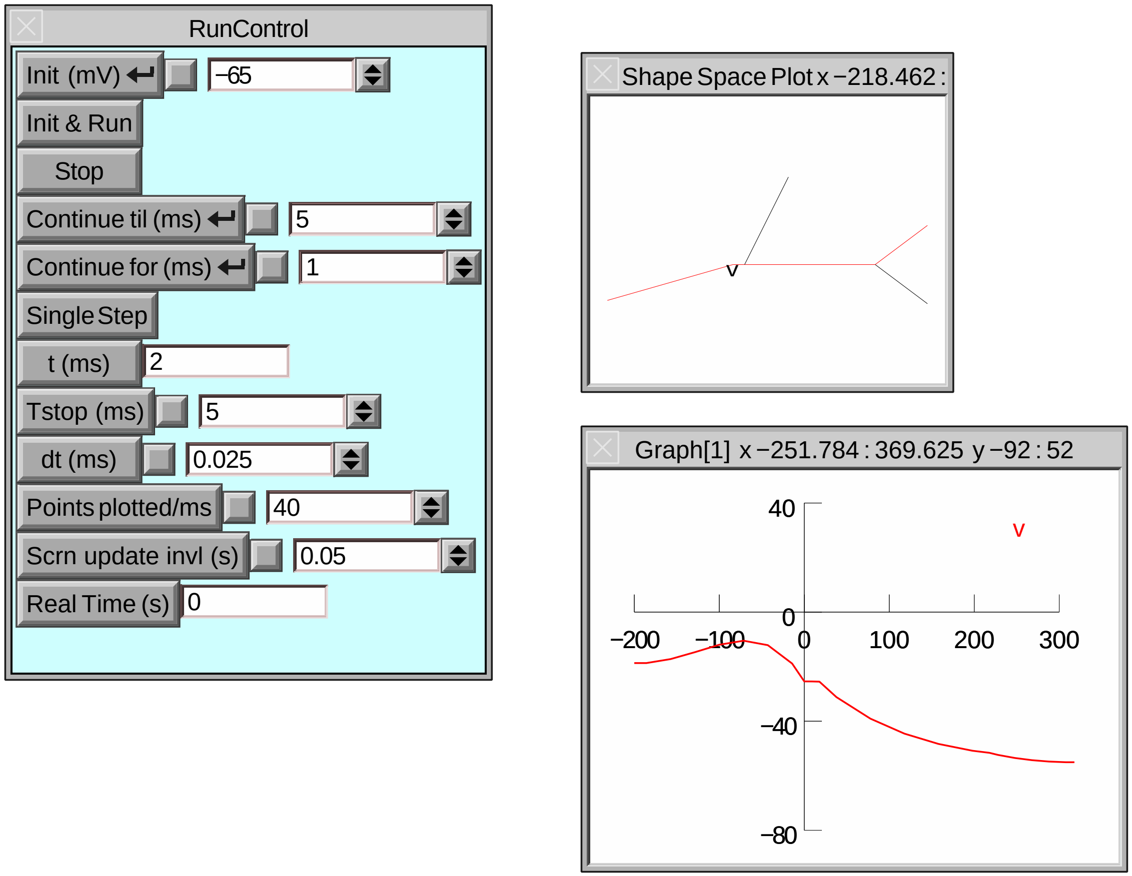
... and ends by specifying a few parameters as constant over a few subsets.

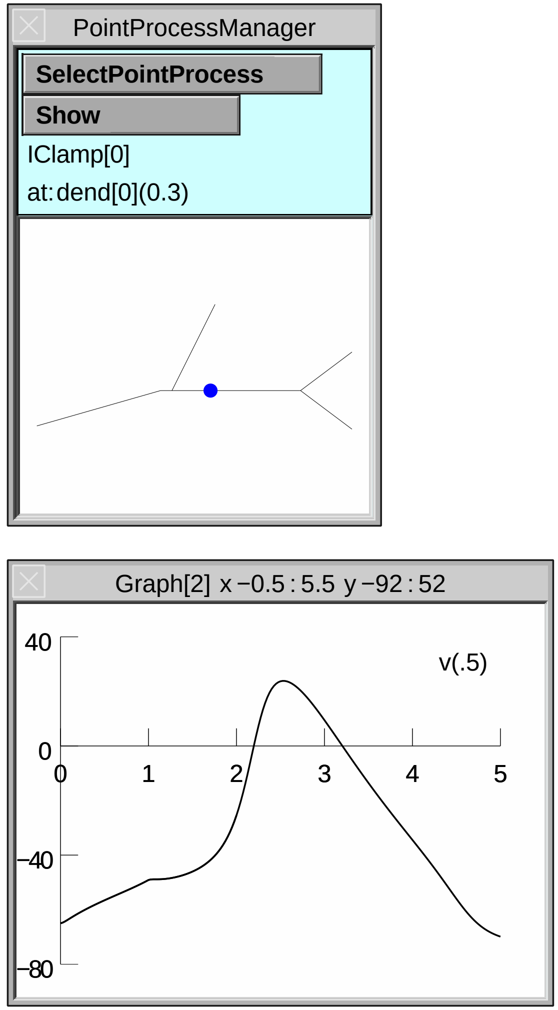
(Inhomogeneities are a bit more complicated)

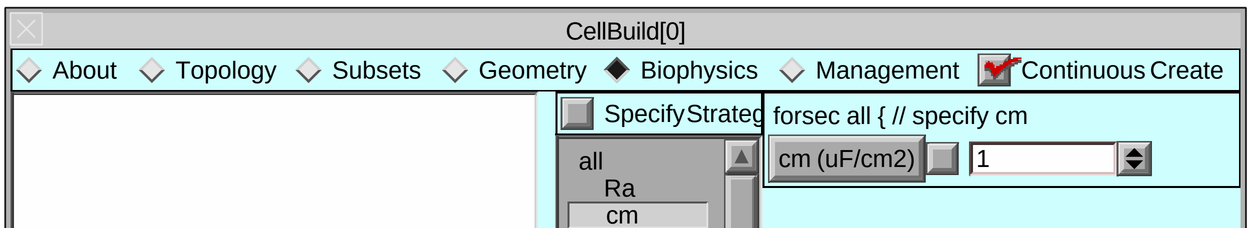


The cell comes into existence when "Continuous Create" is turned on.

(Windows derive from NEURONMainMenu "Tools" and "Graph".)

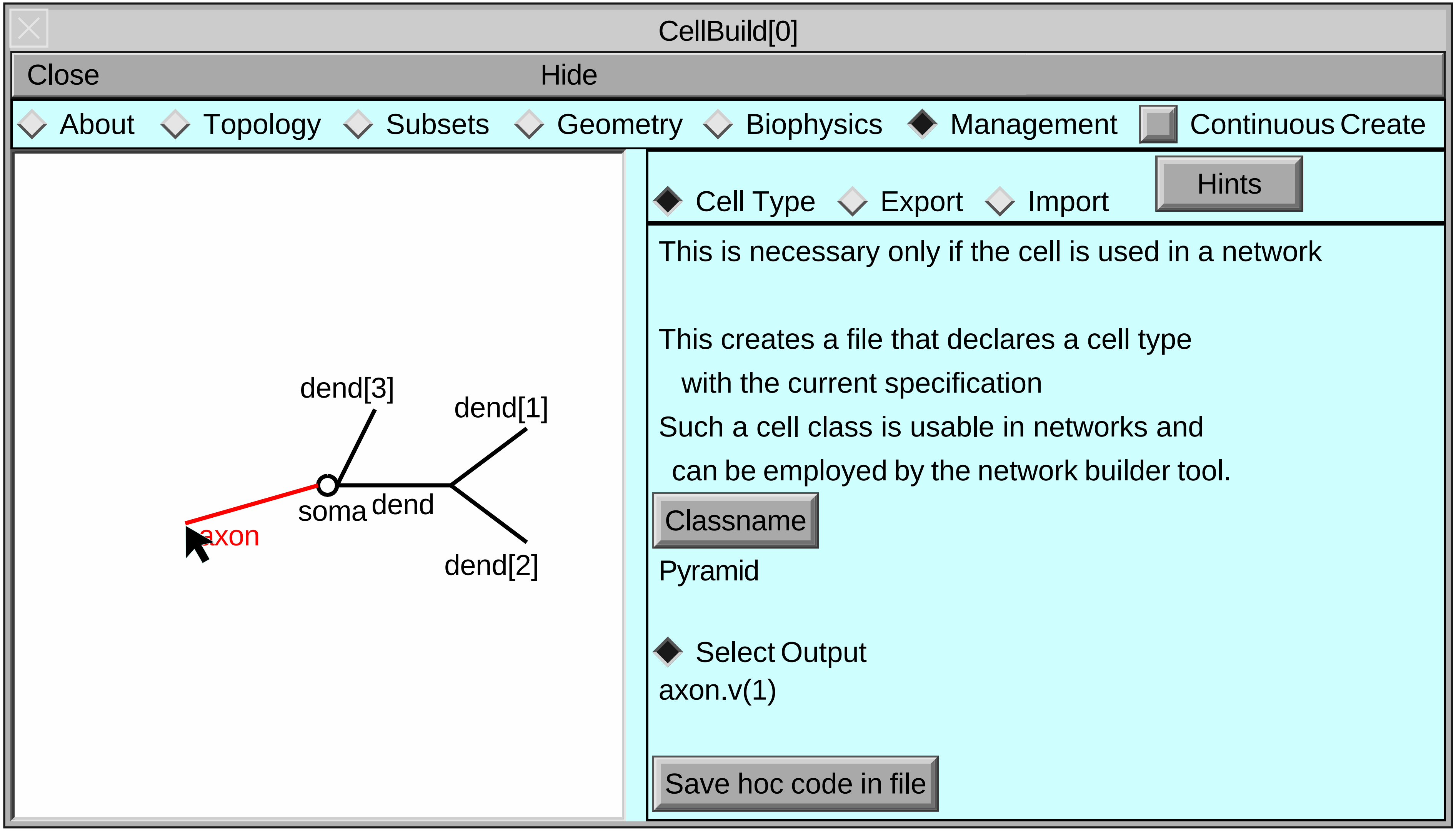






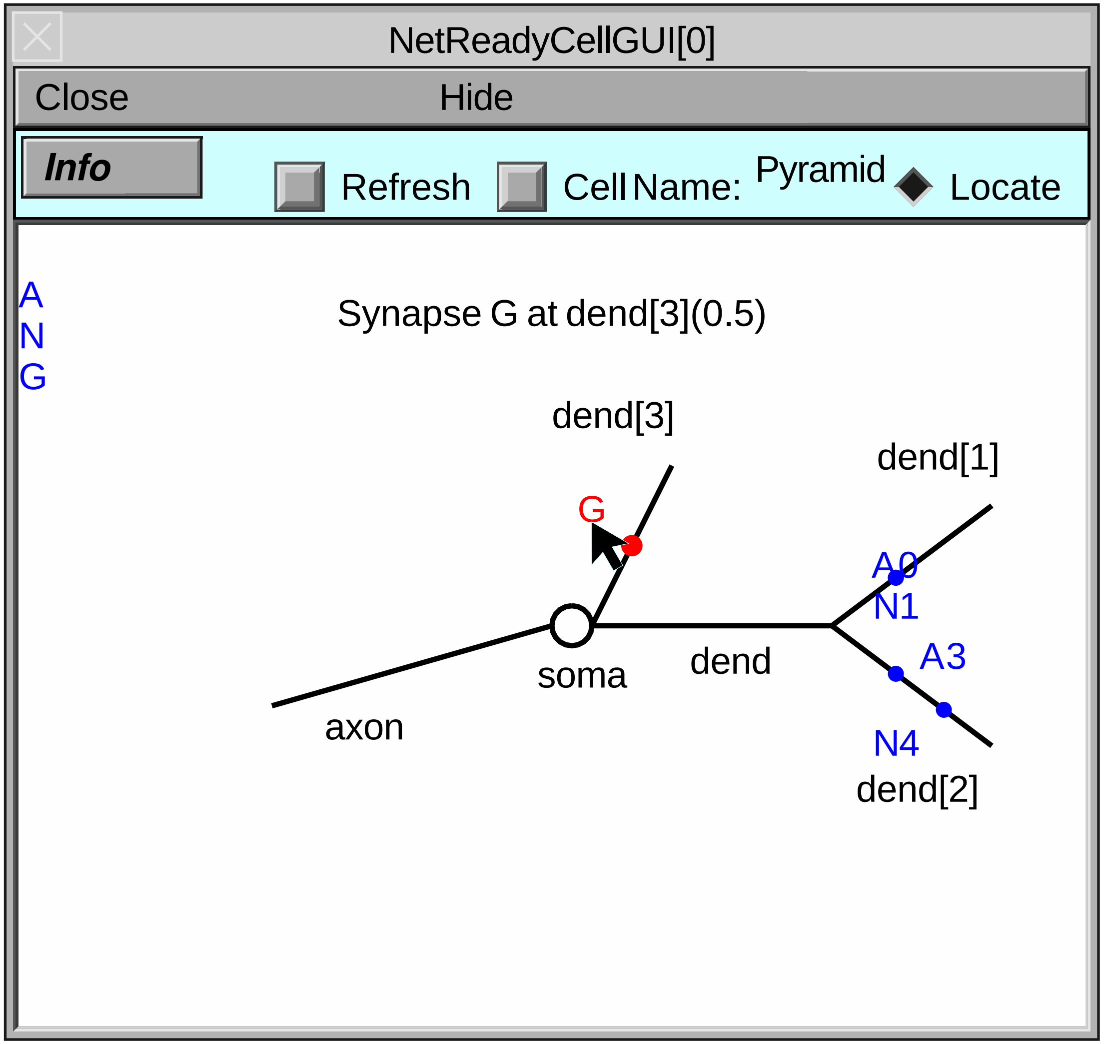
Using the cell in a network takes three steps.

1) Encapsulate in class so many cells can be created with the same type.
2) Specify the location of the output spike detector.



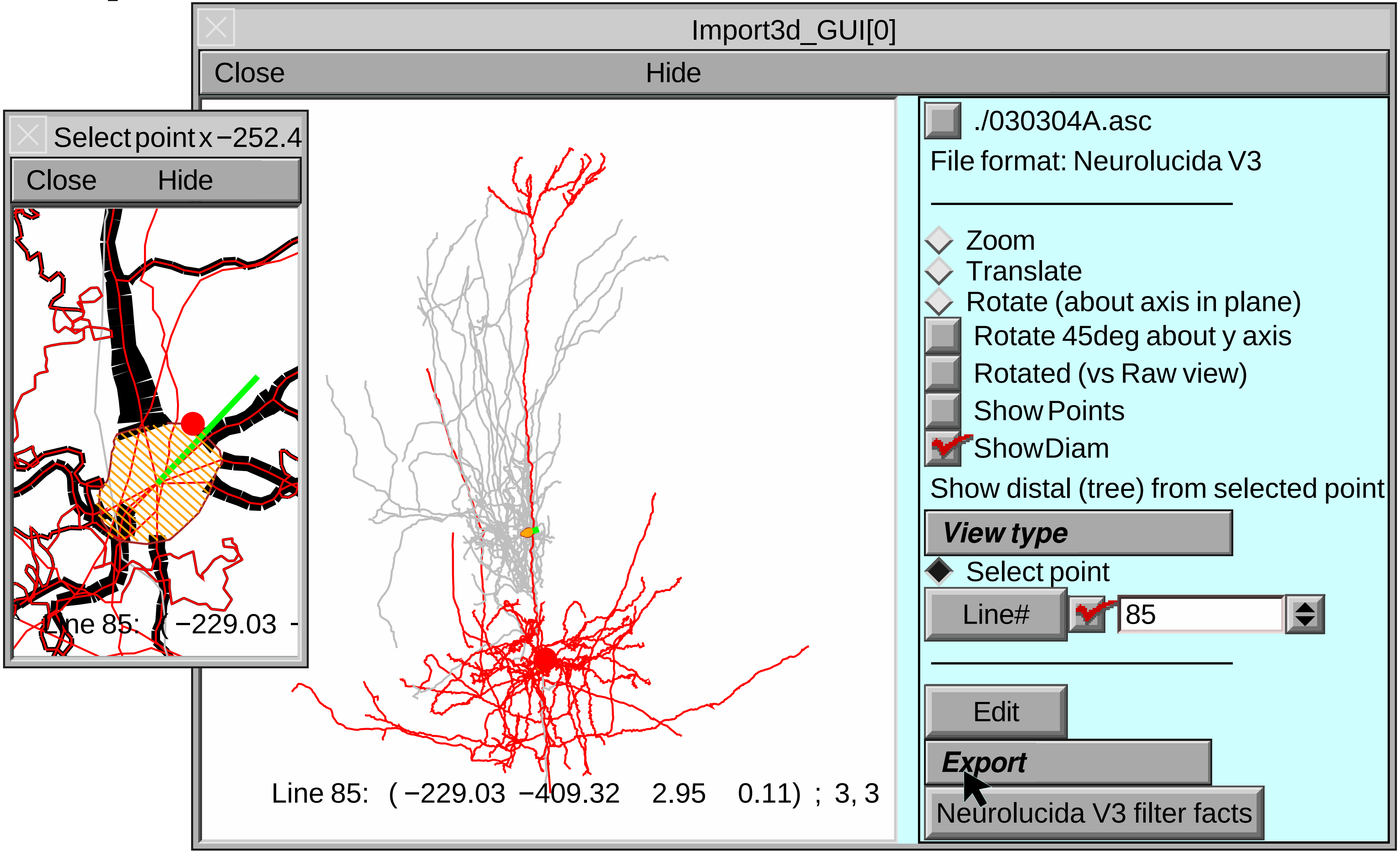
A NetworkReadyCell needs to be able to send and receive spikes.

3) Sprinkle synapses that can receive input spikes.



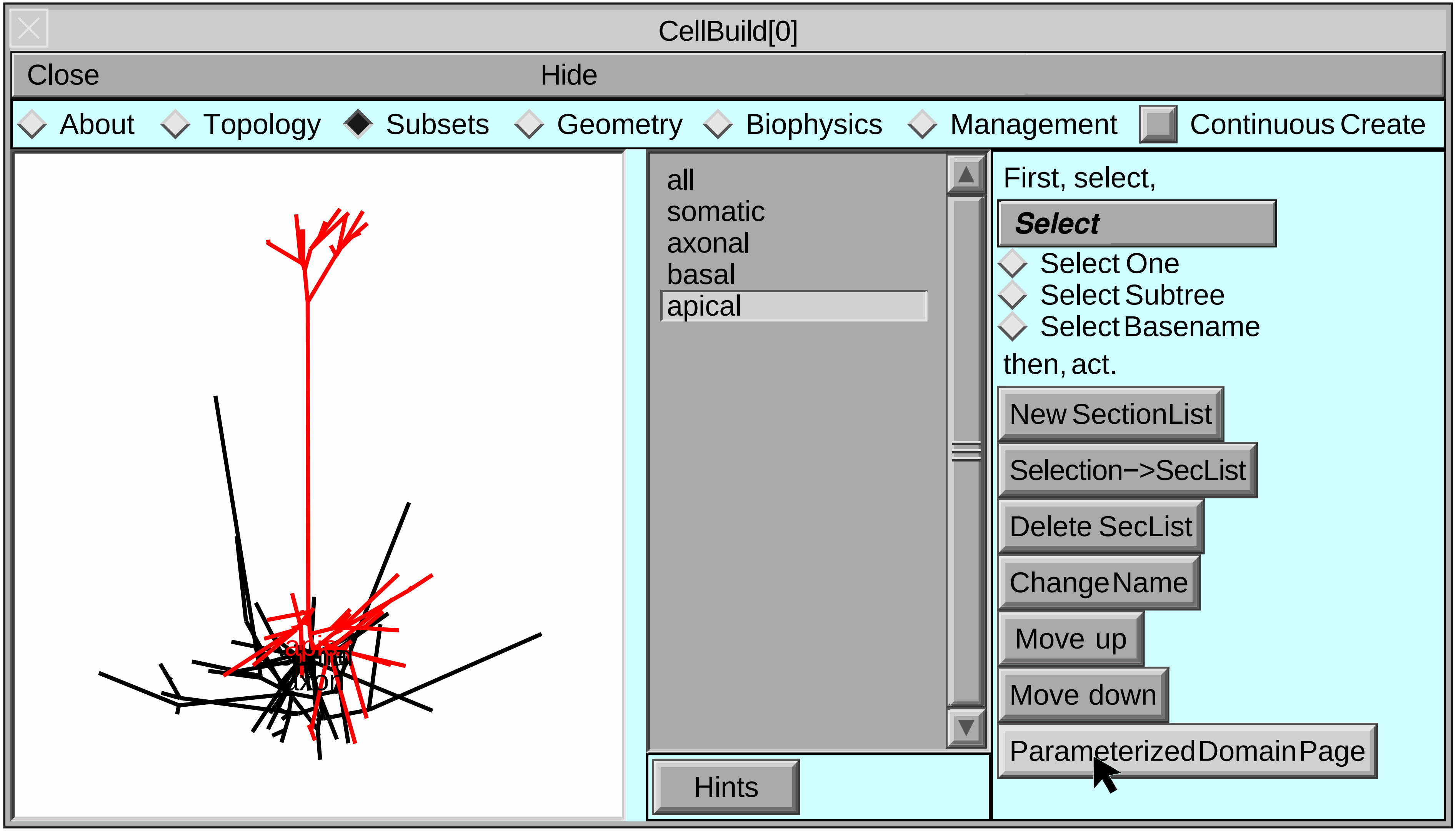
Starting from a 3−D Reconstruction.

Use NEURONMainMenu/Tools/Miscellaneous/Import3D.
Export to Cell Builder.



Import3D export to CellBuild−−−automatically gives Topology, Geometry, and some Subsets.

But don’t forget to compartmentalize with dlambda.



Inhomogeneous parameters.

gnabar_hh(location) = f(p(location))

p: subset domain parameter

path distance from soma

physical distance along line

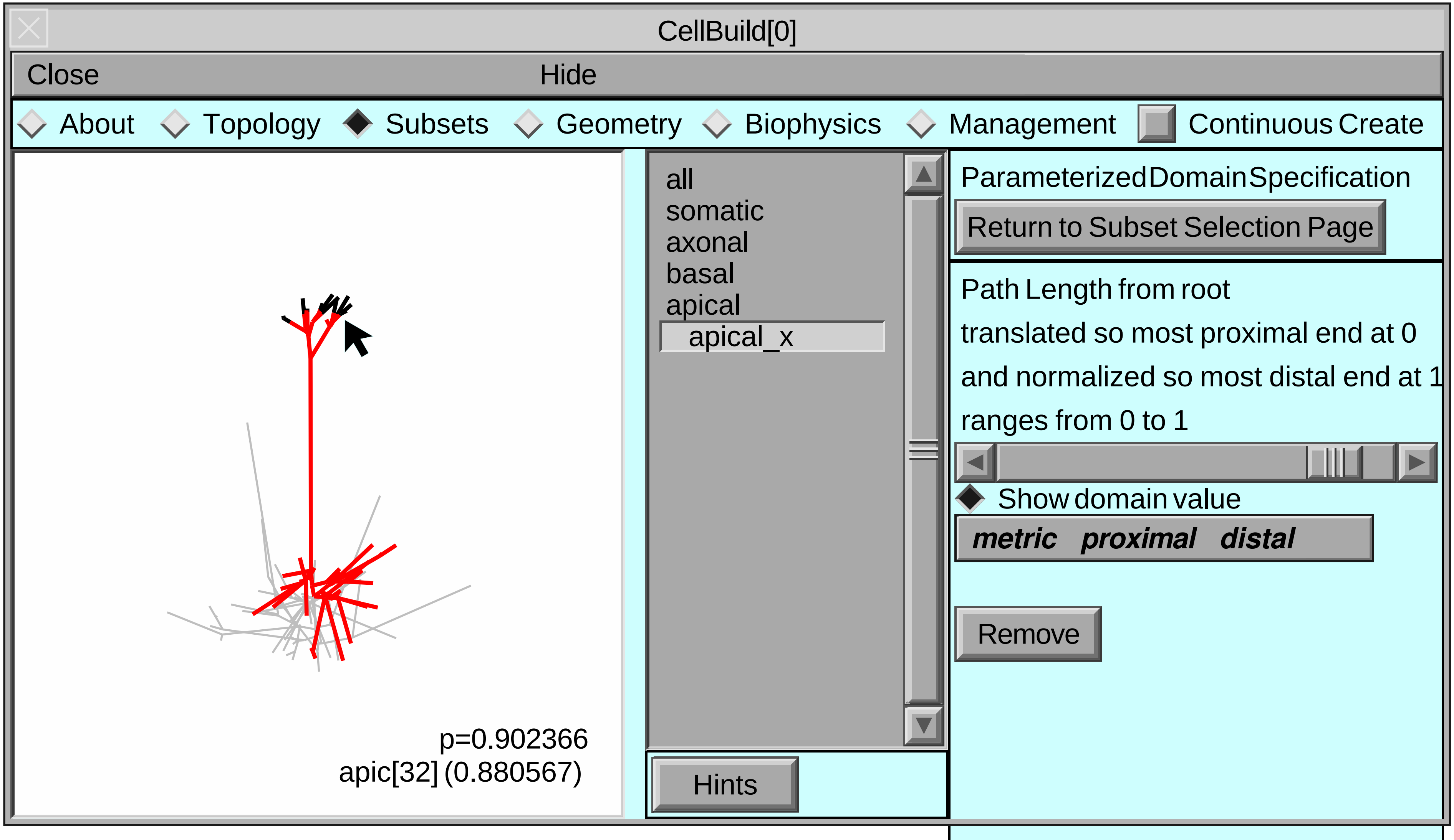
physical distance from point

normalized distance

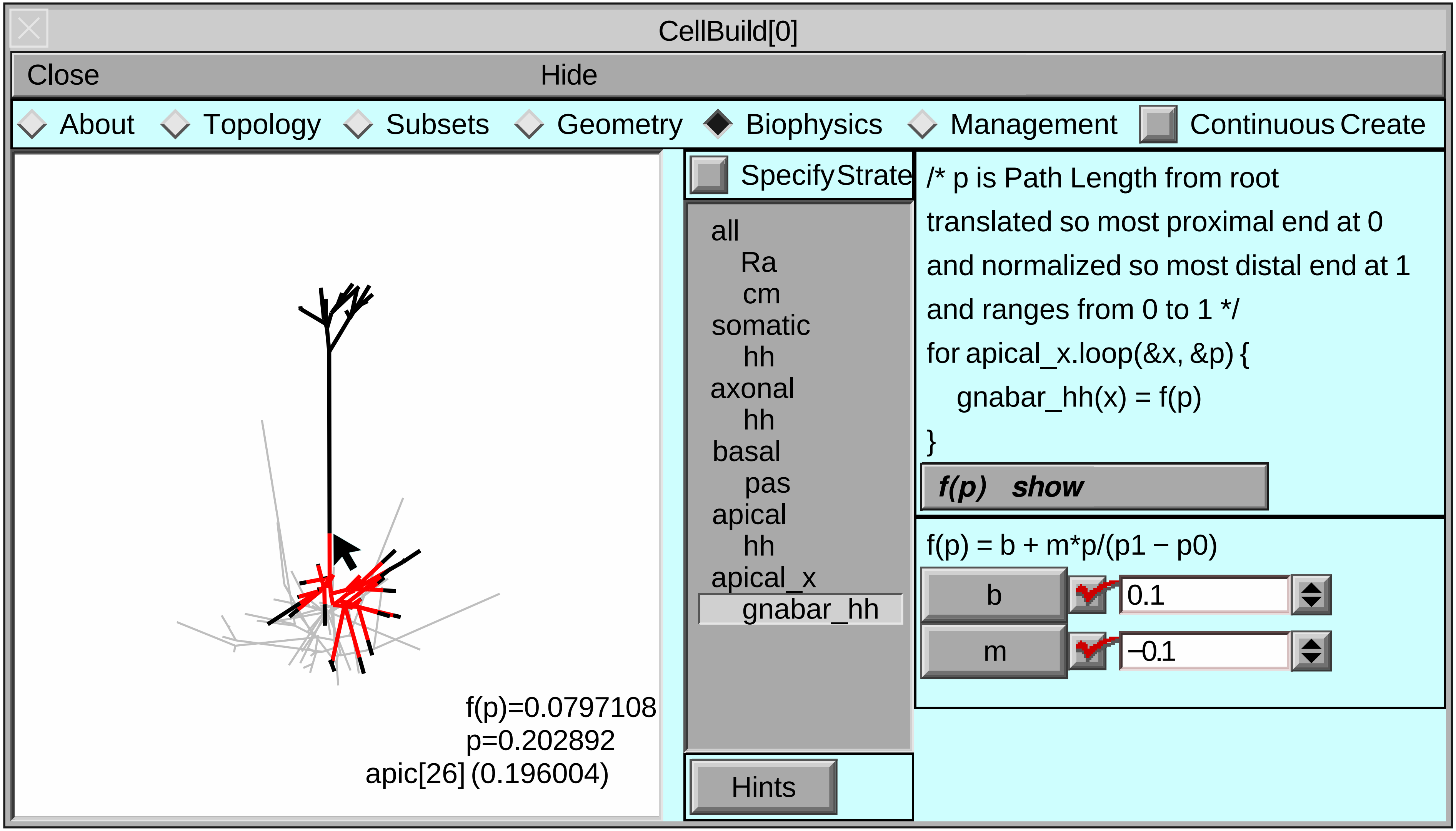
closest point is 0

farthest point is 1

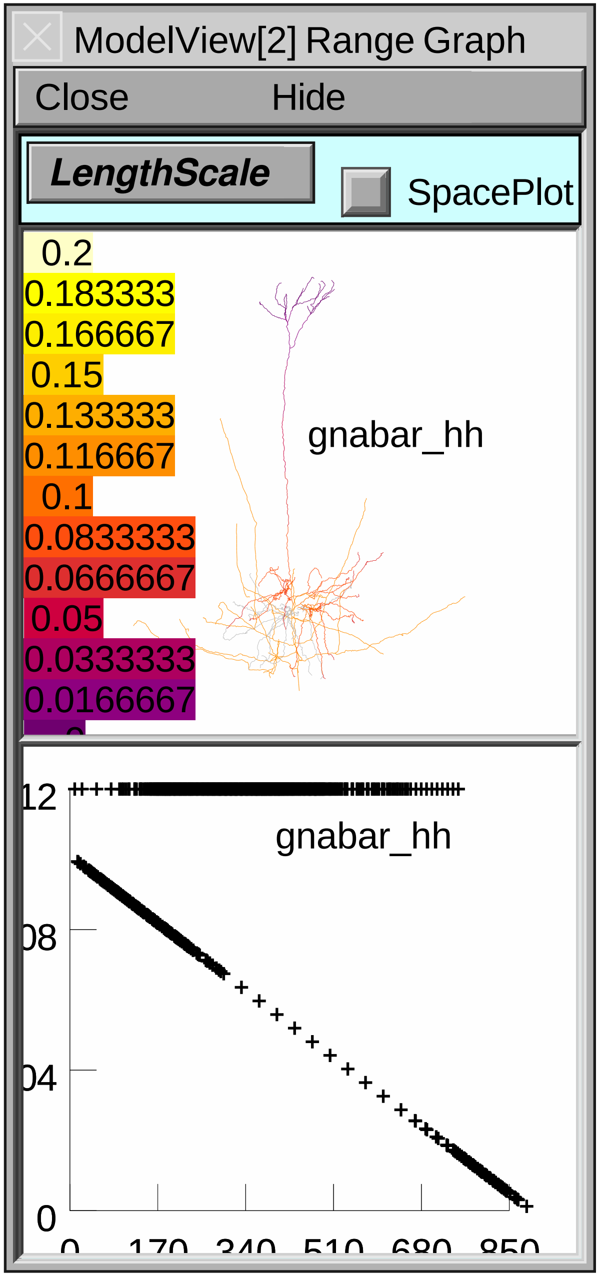
The subset domain parameter defines a value at every location on the subset.

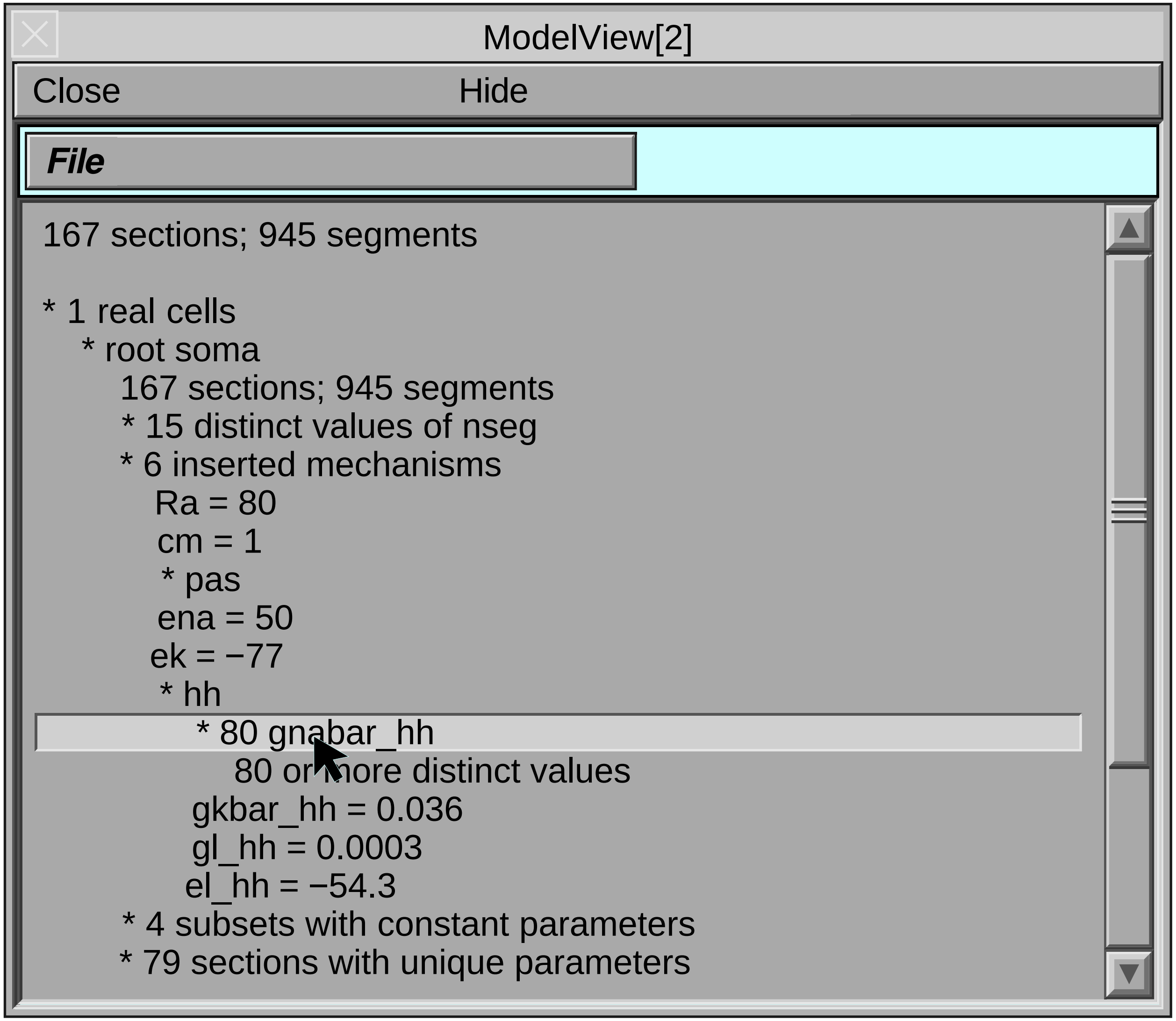


...and arbitrary functions of that parameter can be used to specify a range variable.

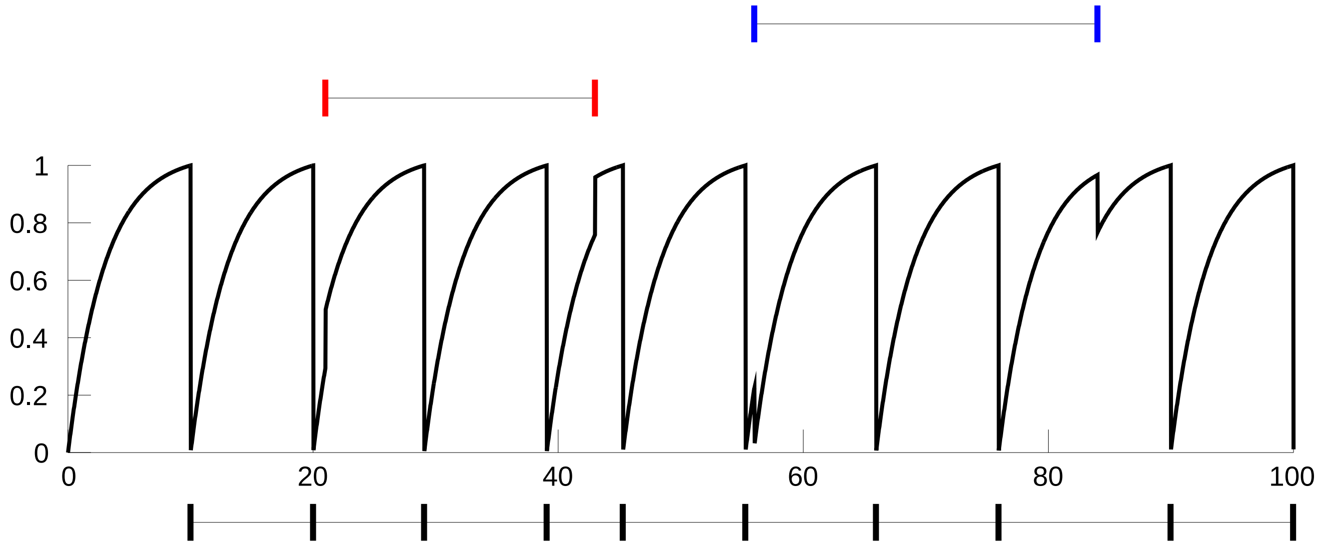


It’s always a good idea to check that expectations are met.



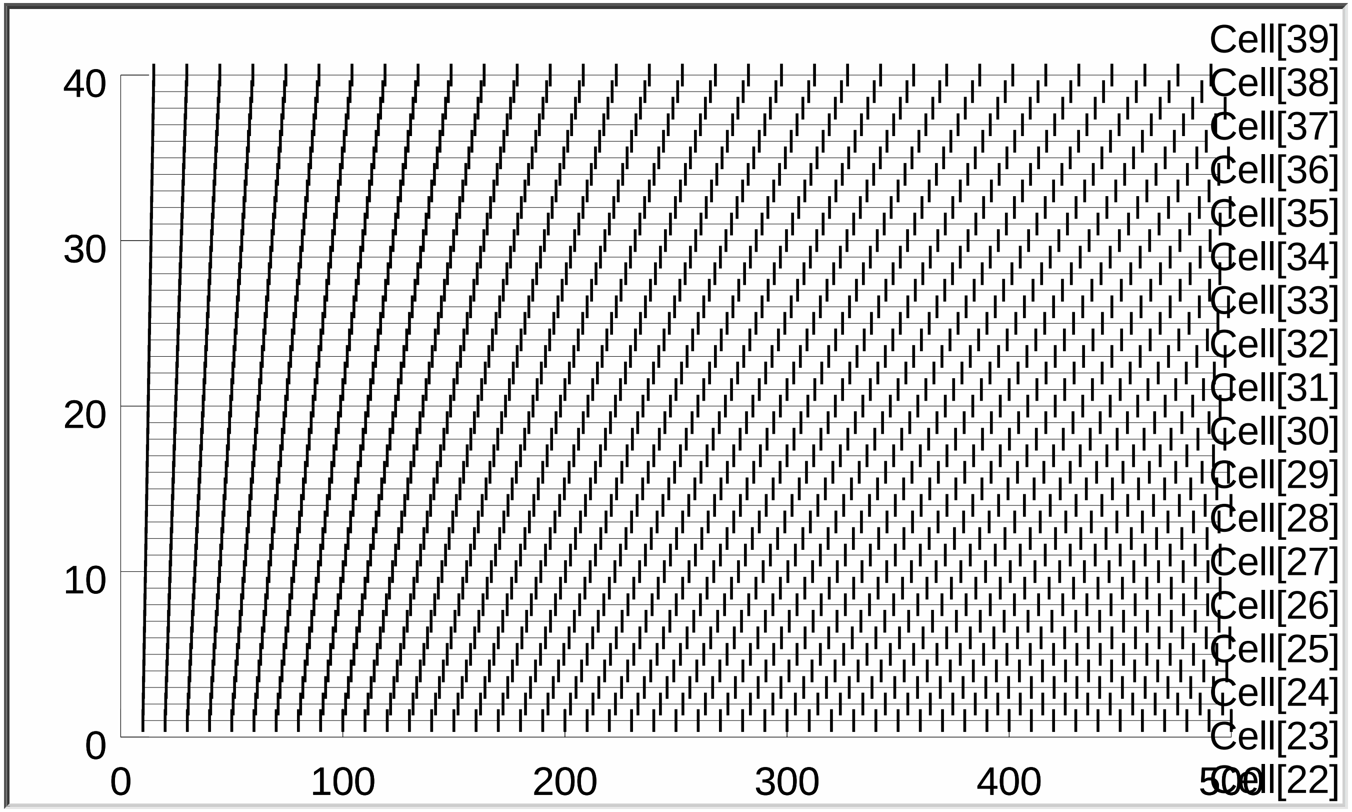


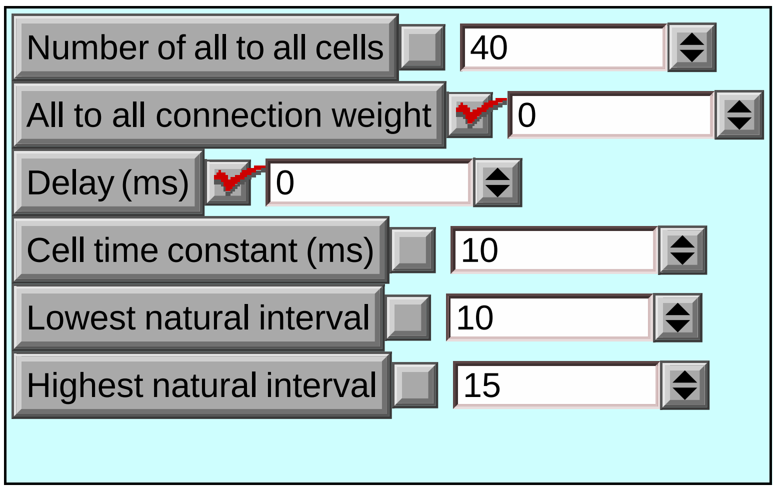
Networks (Inhibitory Synchronization).

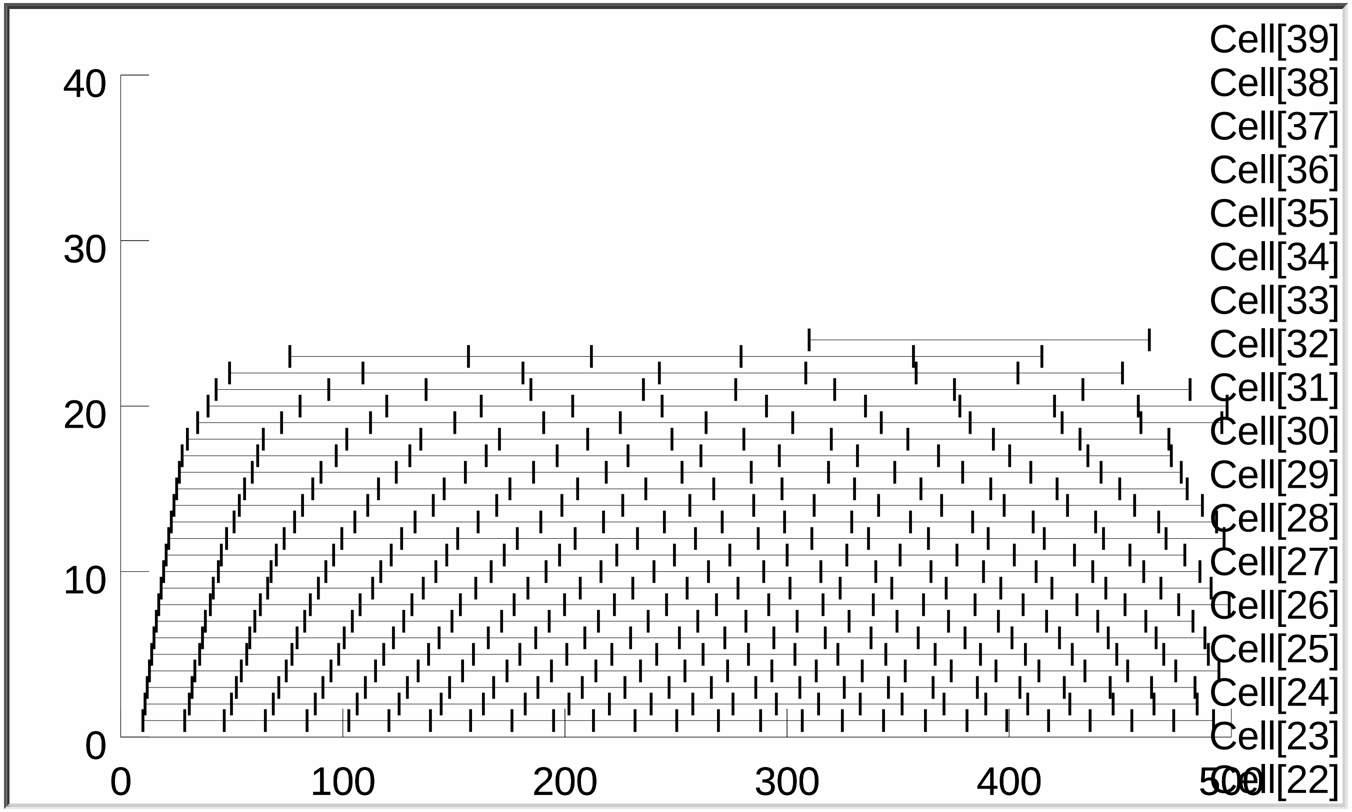


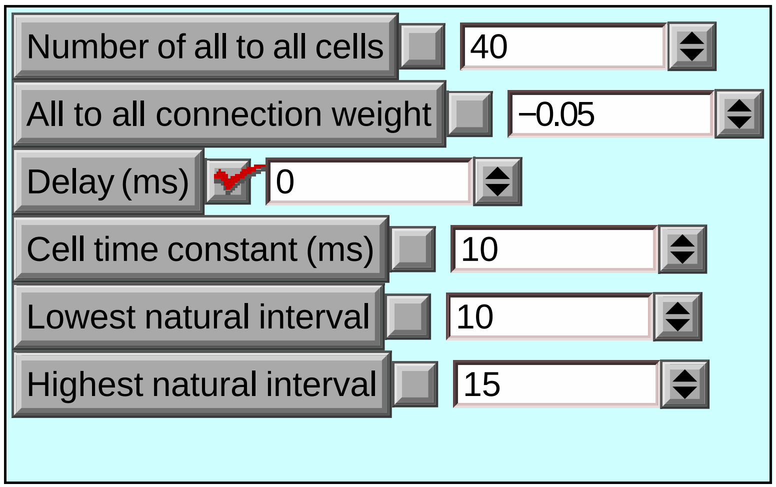

```
: dm/dt = (minf - m)/tau
: input event adds w to m
: when m = 1, or event
:  makes m >= 1, cell fires
: minf is calculated so
:  that the natural interval
:  between spikes is invl

INITIAL {
   minf = 1/(1 - exp(-invl/tau))
   m = 0
   t0 = t
   net_send(firetime(), 1)
}
```

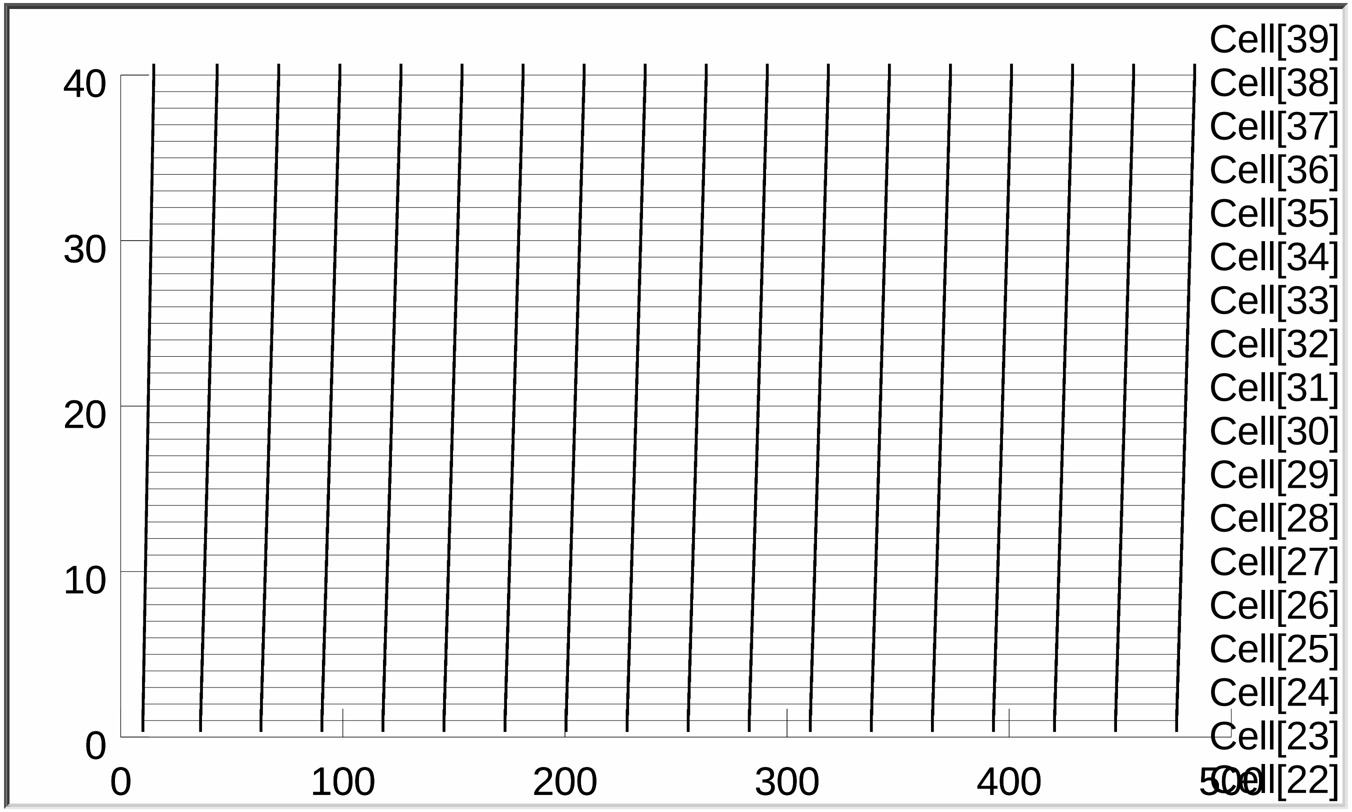
```
NET_RECEIVE (w) {
   m = minf + (m - minf)*exp(-(t - t0)/tau)
   t0 = t
   if (flag == 0) {
      m = m + w
      if (m > 1) {
         m = 0
         net_event(t)
      }
      net_move(t+firetime())
   }else{
      net_event(t)
      m = 0
      net_send(firetime(), 1)
   }
}

FUNCTION firetime() { : m < 1 < minf
   firetime = tau*log((minf-m)/(minf - 1))
}
```

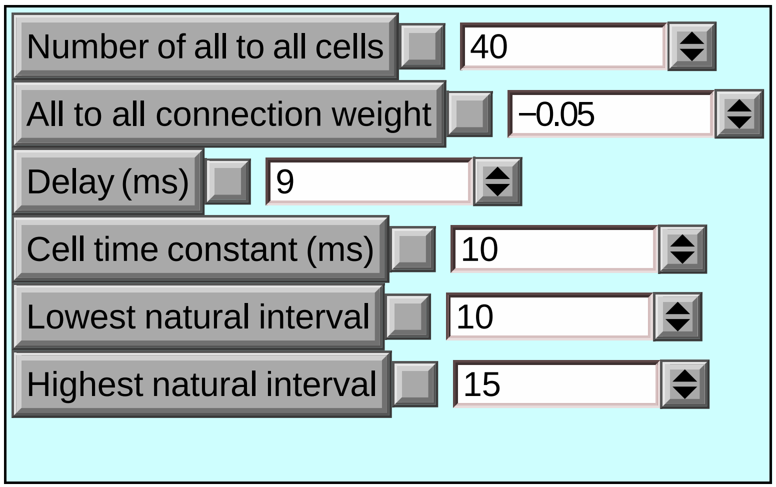

40
30
20
10
0
0
100
200
300
400
500
Cell[39]
Cell[38]
Cell[37]
Cell[36]
Cell[35]
Cell[34]
Cell[33]
Cell[32]
Cell[31]
Cell[30]
Cell[29]
Cell[28]
Cell[27]
Cell[26]
Cell[25]
Cell[24]
Cell[23]
Cell[22]

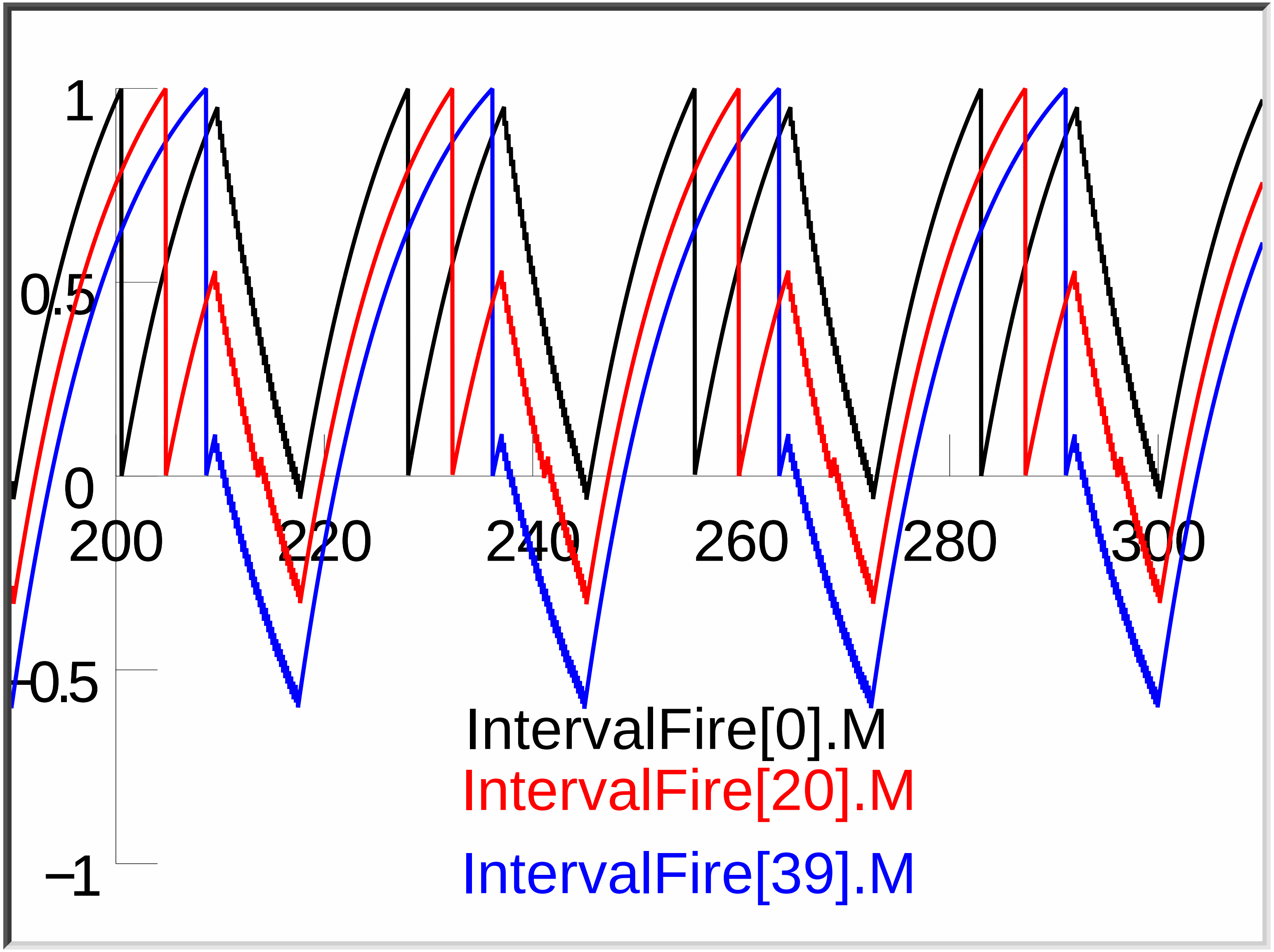

Number of all to all cells
40
All to all connection weight
0
Delay (ms)
0
Cell time constant (ms)
10
Lowest natural interval
10
Highest natural interval
15


40
30
20
10
0
0
100
200
300
400
500
Cell[39]
Cell[38]
Cell[37]
Cell[36]
Cell[35]
Cell[34]
Cell[33]
Cell[32]
Cell[31]
Cell[30]
Cell[29]
Cell[28]
Cell[27]
Cell[26]
Cell[25]
Cell[24]
Cell[23]
Cell[22]


Number of all to all cells
40
All to all connection weight
−0.05
Delay (ms)
0
Cell time constant (ms)
10
Lowest natural interval
10
Highest natural interval
15

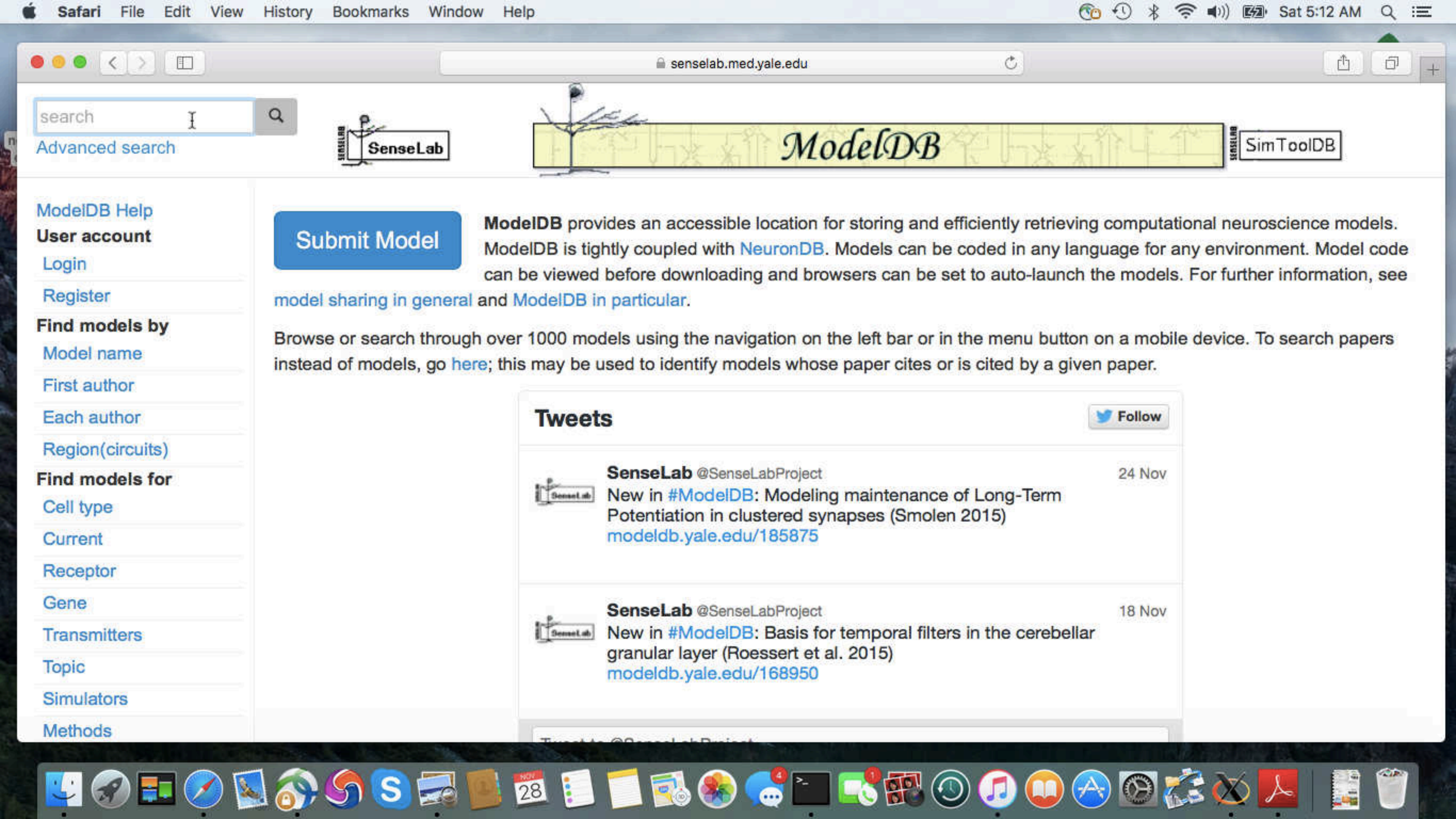

Cell[39]
Cell[38]
Cell[37]
Cell[36]
Cell[35]
Cell[34]
Cell[33]
Cell[32]
Cell[31]
Cell[30]
Cell[29]
Cell[28]
Cell[27]
Cell[26]
Cell[25]
Cell[24]
Cell[23]
Cell[22]
40
30
20
10
0
0
100
200
300
400
500

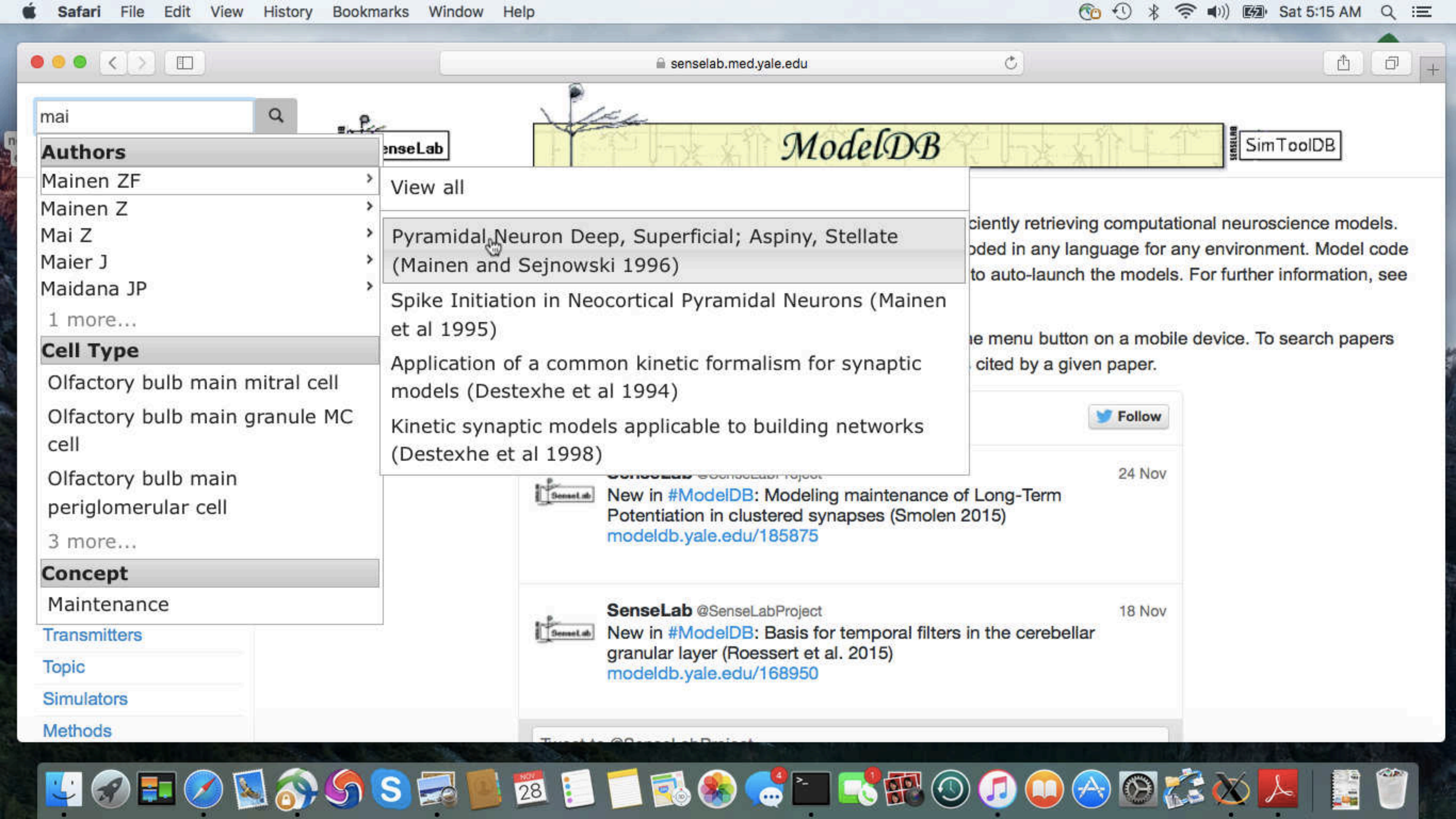

Number of all to all cells
40
All to all connection weight
–0.05
Delay (ms)
9
Cell time constant (ms)
10
Lowest natural interval
10
Highest natural interval
15



1
0.5
0
−0.5
−1
200
220
240
260
280
300
IntervalFire[0].M
IntervalFire[20].M
IntervalFire[39].M

Published models: ModelDB.


Safari File Edit View History Bookmarks Window Help
Sat 5:12 AM
senselab.med.yale.edu
search
Advanced search
SenseLab
ModelDB
SimToolDB
ModelDB Help
User account
Login
Register
Find models by
Model name
First author
Each author
Region(circuits)
Find models for
Cell type
Current
Receptor
Gene
Transmitters
Topic
Simulators
Methods
Submit Model
ModelDB provides an accessible location for storing and efficiently retrieving computational neuroscience models. ModelDB is tightly coupled with NeuronDB. Models can be coded in any language for any environment. Model code can be viewed before downloading and browsers can be set to auto-launch the models. For further information, see model sharing in general and ModelDB in particular.
Browse or search through over 1000 models using the navigation on the left bar or in the menu button on a mobile device. To search papers instead of models, go here; this may be used to identify models whose paper cites or is cited by a given paper.
Tweets
Follow
SenseLab @SenseLabProject
24 Nov
New in #ModelDB: Modeling maintenance of Long-Term Potentiation in clustered synapses (Smolen 2015)
modeldb.yale.edu/185875
SenseLab @SenseLabProject
18 Nov
New in #ModelDB: Basis for temporal filters in the cerebellar granular layer (Roessert et al. 2015)
modeldb.yale.edu/168950
28


Safari File Edit View History Bookmarks Window Help
Sat 5:15 AM
senselab.med.yale.edu
mai
Authors
Mainen ZF
Mainen Z
Mai Z
Maier J
Maidana JP
1 more...
Cell Type
Olfactory bulb main mitral cell
Olfactory bulb main granule MC cell
Olfactory bulb main periglomerular cell
3 more...
Concept
Maintenance
Transmitters
Topic
Simulators
Methods
ModelDB
SimToolDB
View all
Pyramidal Neuron Deep, Superficial; Aspiny, Stellate (Mainen and Sejnowski 1996)
Spike Initiation in Neocortical Pyramidal Neurons (Mainen et al 1995)
Application of a common kinetic formalism for synaptic models (Destexhe et al 1994)
Kinetic synaptic models applicable to building networks (Destexhe et al 1998)
ciently retrieving computational neuroscience models.
oded in any language for any environment. Model code
to auto-launch the models. For further information, see
e menu button on a mobile device. To search papers
cited by a given paper.
Follow
24 Nov
New in #ModelDB: Modeling maintenance of Long-Term Potentiation in clustered synapses (Smolen 2015)
modeldb.yale.edu/185875
SenseLab @SenseLabProject
18 Nov
New in #ModelDB: Basis for temporal filters in the cerebellar granular layer (Roessert et al. 2015)
modeldb.yale.edu/168950

Safari File Edit View History Bookmarks Window Help Sat 5:16 AM

senselab.med.yale.edu

search

Advanced search

SenseLab ModelDB SimToolDB

ModelDB Help

User account

Login

Register

Find models by

Model name

First author

Each author

Region(circuits)

Find models for

Cell type

Current

Receptor

Gene

Transmitters

Topic

Simulators

Methods

Pyramidal Neuron Deep, Superficial; Aspiny, Stellate (Mainen and Sejnowski 1996)

Accession:2488

This package contains compartmental models of four reconstructed neocortical neurons (layer 3 Aspiny, layer 4 Stellate, layer 3 and layer 5 Pyramidal neurons) with active dendritic currents using NEURON. Running this simulation demonstrates that an entire spectrum of firing patterns can be reproduced in this set of model neurons which share a common distribution of ion channels and differ only in their dendritic geometry. The reference paper is: Z. F. Mainen and T. J. Sejnowski (1996) Influence of dendritic structure on firing pattern in model neocortical neurons. Nature 382: 363-366. See also http://www.cnl.salk.edu/~zach/methods.html and http://www.cnl.salk.edu/~zach/ More info in readme.txt file below made visible by clicking on the patdemo folder and then on the readme.txt file.

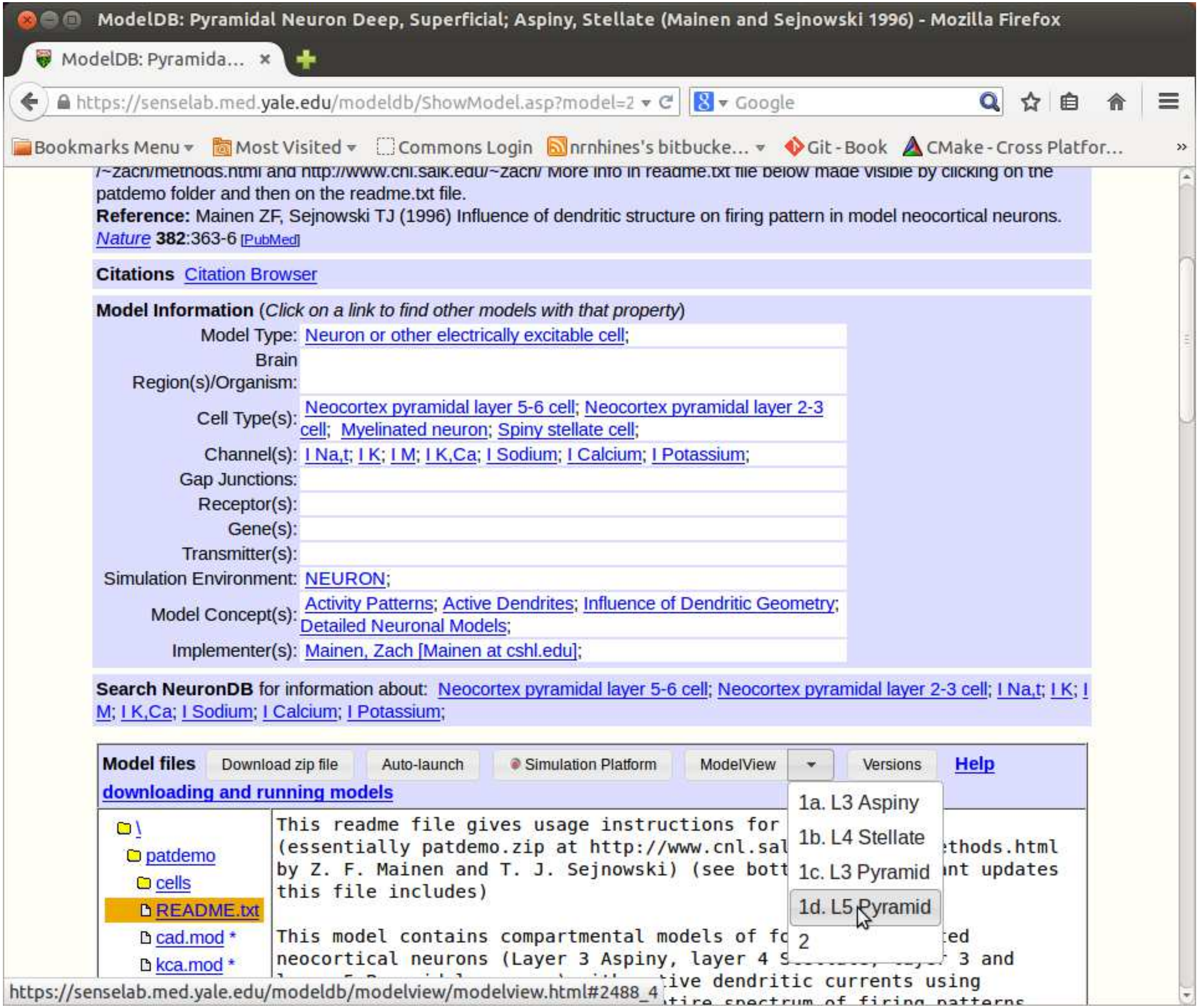
Reference:

1 . Mainen ZF, Sejnowski TJ (1996) Influence of dendritic structure on firing pattern in model neocortical neurons. *Nature* **382**:363-6 [PubMed]

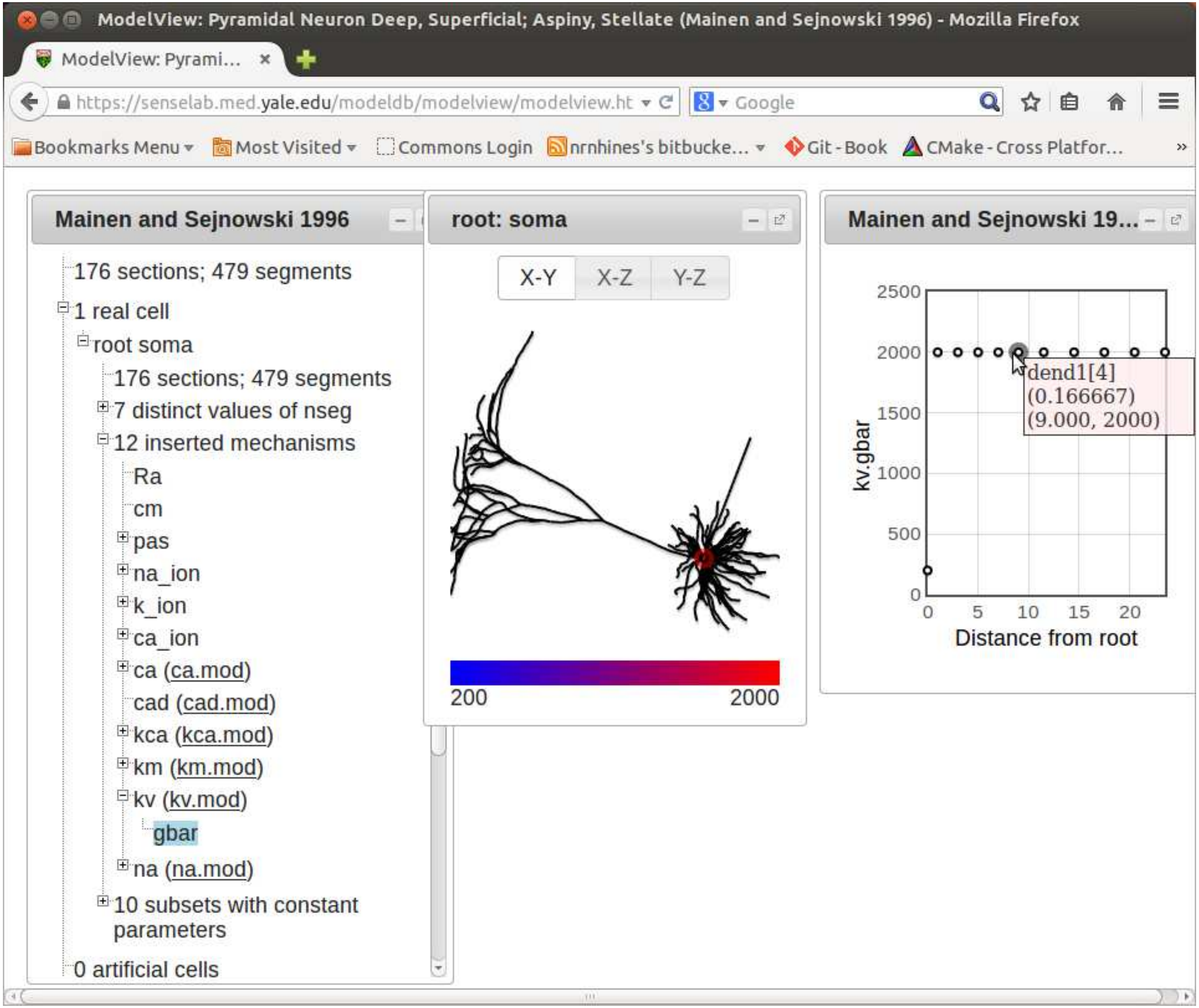
Citations Citation Browser

Model Information (Click on a link to find other models with that property)

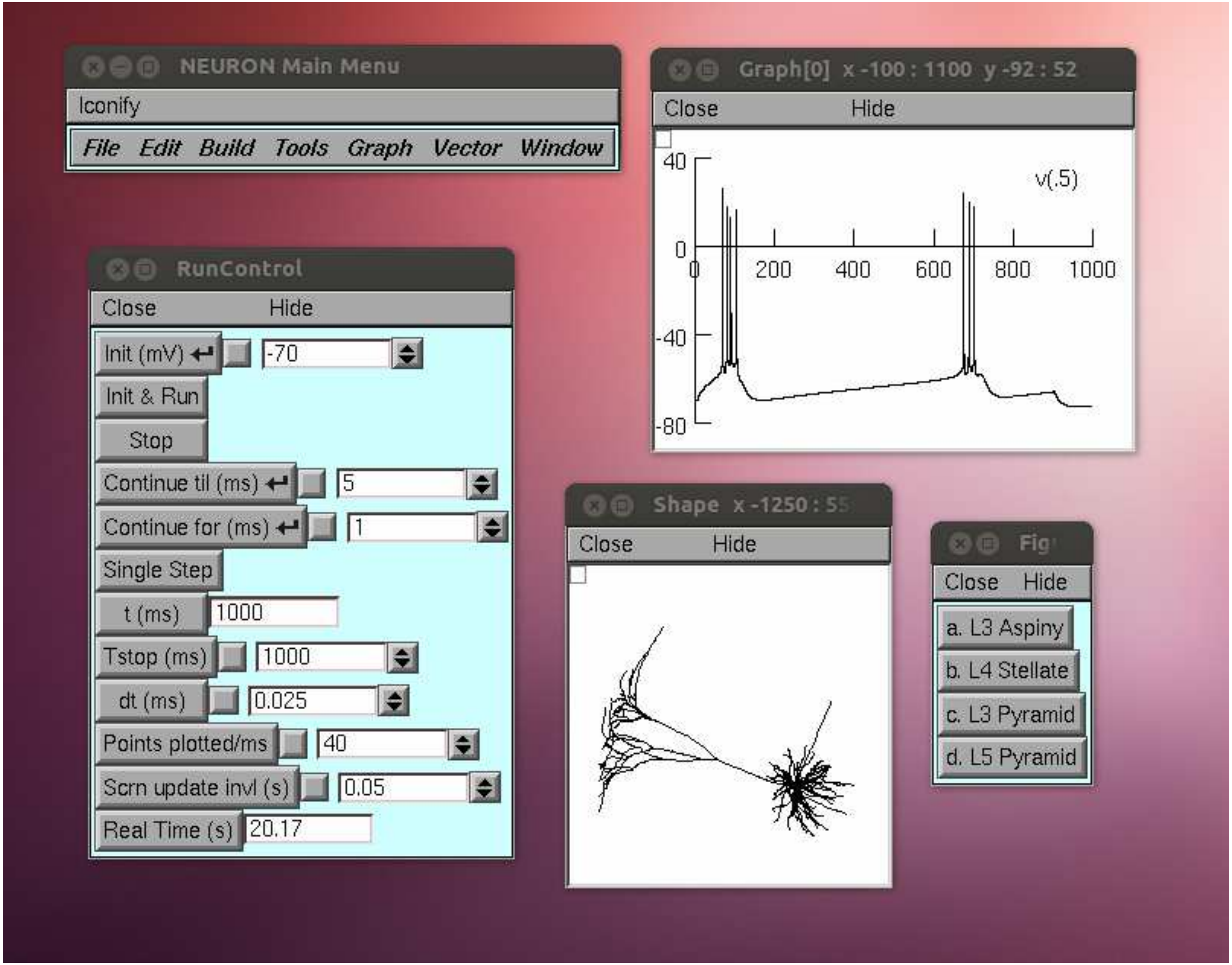
Model Type:	Neuron or other electrically excitable cell;
Brain Region(s)/Organism:	
Cell Type(s):	Neocortex layer 5-6 pyramidal cell; Neocortex layer 2-3 pyramidal cell; Myelinated neuron; Neocortex spiny stellate cell;



ModelDB: Pyramidal Neuron Deep, Superficial; Aspiny, Stellate (Mainen and Sejnowski 1996) - Mozilla Firefox
ModelDB: Pyramida...
https://senselab.med.yale.edu/modeldb/ShowModel.asp?model=2
Google
Bookmarks Menu
Most Visited
Commons Login
nrnhines's bitbucke...
Git - Book
CMake - Cross Platfor...
patdemo folder and then on the readme.txt file.
Reference: Mainen ZF, Sejnowski TJ (1996) Influence of dendritic structure on firing pattern in model neocortical neurons. Nature 382:363-6 [PubMed]
Citations Citation Browser
Model Information (Click on a link to find other models with that property)
Model Type: Neuron or other electrically excitable cell;
Brain Region(s)/Organism:
Cell Type(s): Neocortex pyramidal layer 5-6 cell; Neocortex pyramidal layer 2-3 cell; Myelinated neuron; Spiny stellate cell;
Channel(s): I Na,t; I K; I M; I K,Ca; I Sodium; I Calcium; I Potassium;
Gap Junctions:
Receptor(s):
Gene(s):
Transmitter(s):
Simulation Environment: NEURON;
Model Concept(s): Activity Patterns; Active Dendrites; Influence of Dendritic Geometry; Detailed Neuronal Models;
Implementer(s): Mainen, Zach [Mainen at cshl.edu];
Search NeuronDB for information about: Neocortex pyramidal layer 5-6 cell; Neocortex pyramidal layer 2-3 cell; I Na,t; I K; I M; I K,Ca; I Sodium; I Calcium; I Potassium;
Model files
Download zip file
Auto-launch
Simulation Platform
ModelView
Versions
Help downloading and running models
1a. L3 Aspiny
1b. L4 Stellate
1c. L3 Pyramid
1d. L5 Pyramid
2
\
patdemo
cells
README.txt
cad.mod *
kca.mod *
This readme file gives usage instructions for
(essentially patdemo.zip at http://www.cnl.sal
by Z. F. Mainen and T. J. Sejnowski) (see bott
this file includes)
This model contains compartmental models of fo
neocortical neurons (Layer 3 Aspiny, layer 4 S
https://senselab.med.yale.edu/modeldb/modelview/modelview.html#2488_4



ModelView: Pyramidal Neuron Deep, Superficial; Aspiny, Stellate (Mainen and Sejnowski 1996) - Mozilla Firefox
ModelView: Pyrami...
https://senselab.med.yale.edu/modeldb/modelview/modelview.ht
Google
Bookmarks Menu
Most Visited
Commons Login
nrnhines's bitbucke...
Git - Book
CMake - Cross Platfor...
Mainen and Sejnowski 1996
176 sections; 479 segments
1 real cell
root soma
176 sections; 479 segments
7 distinct values of nseg
12 inserted mechanisms
Ra
cm
pas
na_ion
k_ion
ca_ion
ca (ca.mod)
cad (cad.mod)
kca (kca.mod)
km (km.mod)
kv (kv.mod)
gbar
na (na.mod)
10 subsets with constant parameters
0 artificial cells
root: soma
X-Y
X-Z
Y-Z
200
2000
Mainen and Sejnowski 19...
2500
2000
1500
1000
500
0
kv.gbar
0
5
10
15
20
Distance from root
dend1[4]
(0.166667)
(9.000, 2000)


NEURON Main Menu
Iconify
File Edit Build Tools Graph Vector Window
Graph[0] x -100 : 1100 y -92 : 52
Close Hide
40
0
-40
-80
0 200 400 600 800 1000
v(.5)
RunControl
Close Hide
Init (mV)
-70
Init & Run
Stop
Continue til (ms)
5
Continue for (ms)
1
Single Step
t (ms)
1000
Tstop (ms)
1000
dt (ms)
0.025
Points plotted/ms
40
Scrn update invl (s)
0.05
Real Time (s)
20.17
Shape x -1250 : 55
Close Hide
Close Hide
a. L3 Aspiny
b. L4 Stellate
c. L3 Pyramid
d. L5 Pyramid