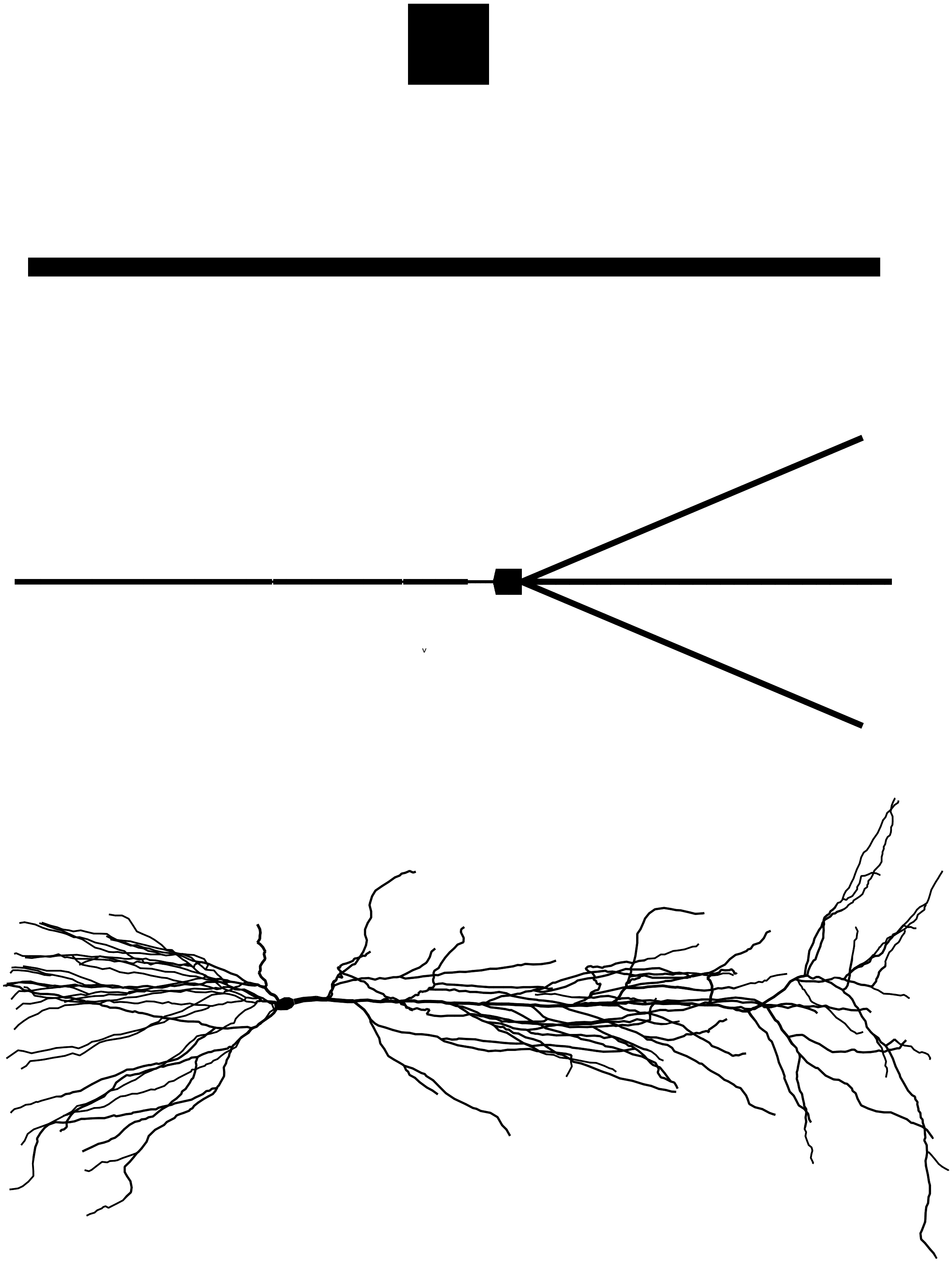
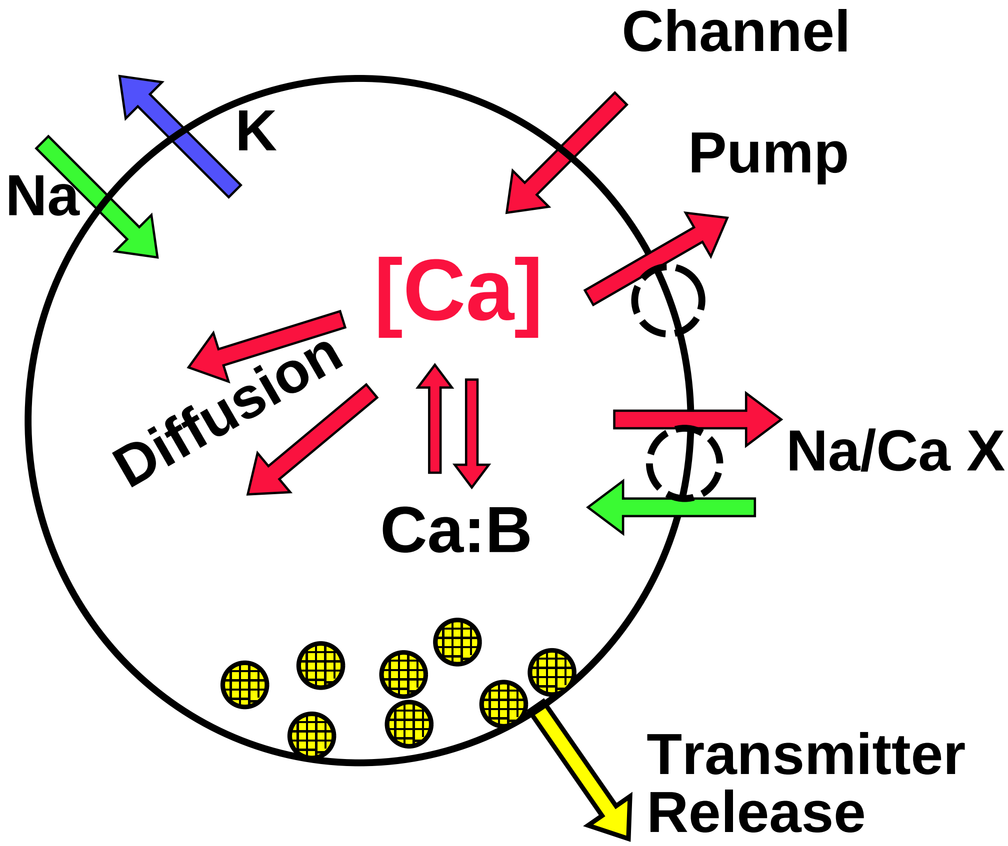
The NEURON Simulation Environment

Ted Carnevale
Michael Hines
John Moore
Gordon Shepherd

http://www.neuron.yale.edu

Supported by NINDS





Extracellular fields
Linear circuits
Synapses
Networks

Physical System

Model

Representation in NEURON

Create Representation

Investigate/Explore/Control/Use Representation

Hodgkin - Huxley axon

Physical System

From http://www.mbl.edu

Model

Hodgkin-Huxley cable equations

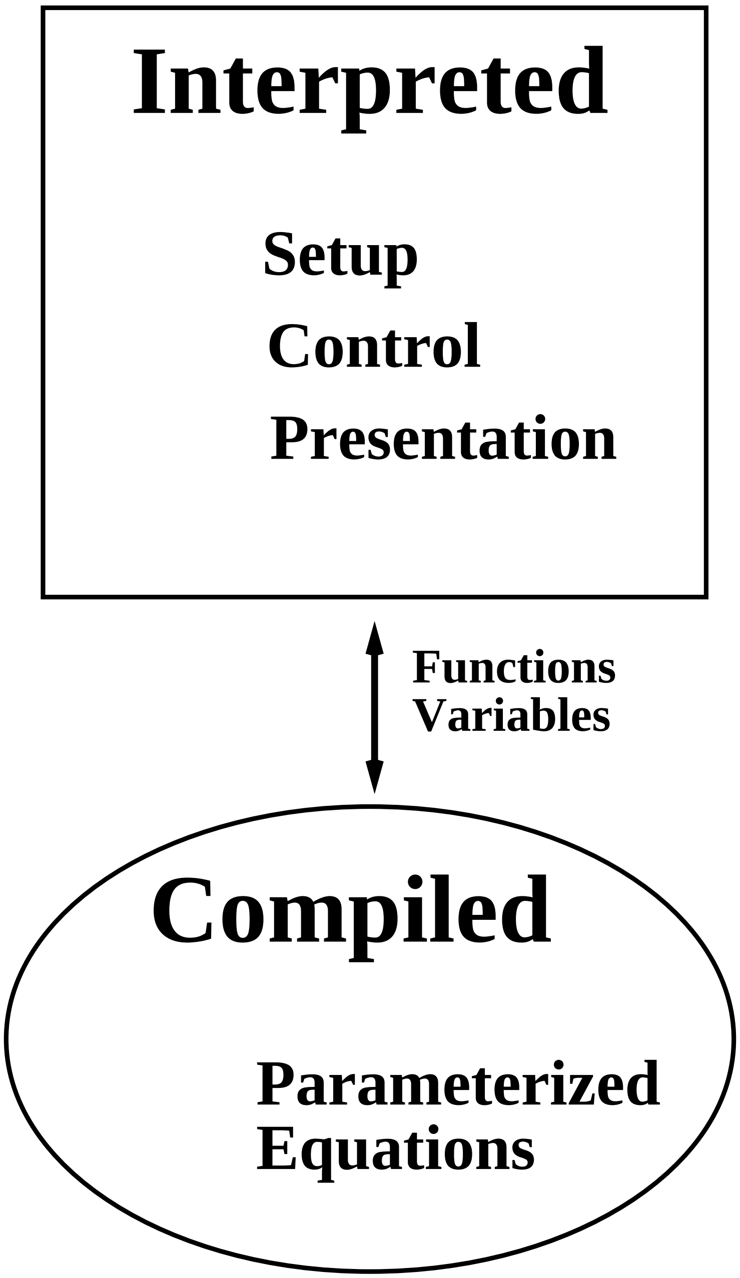
$$\frac{D}{4R_a} \cdot \frac{\partial^2 V}{\partial x^2} = C_m \frac{\partial V}{\partial t} + \overline{g}_{na} m^3 h \cdot (V - E_{na}) + \overline{g}_k n^4 \cdot (V - E_k) + g_l \cdot (V - E_l)$$

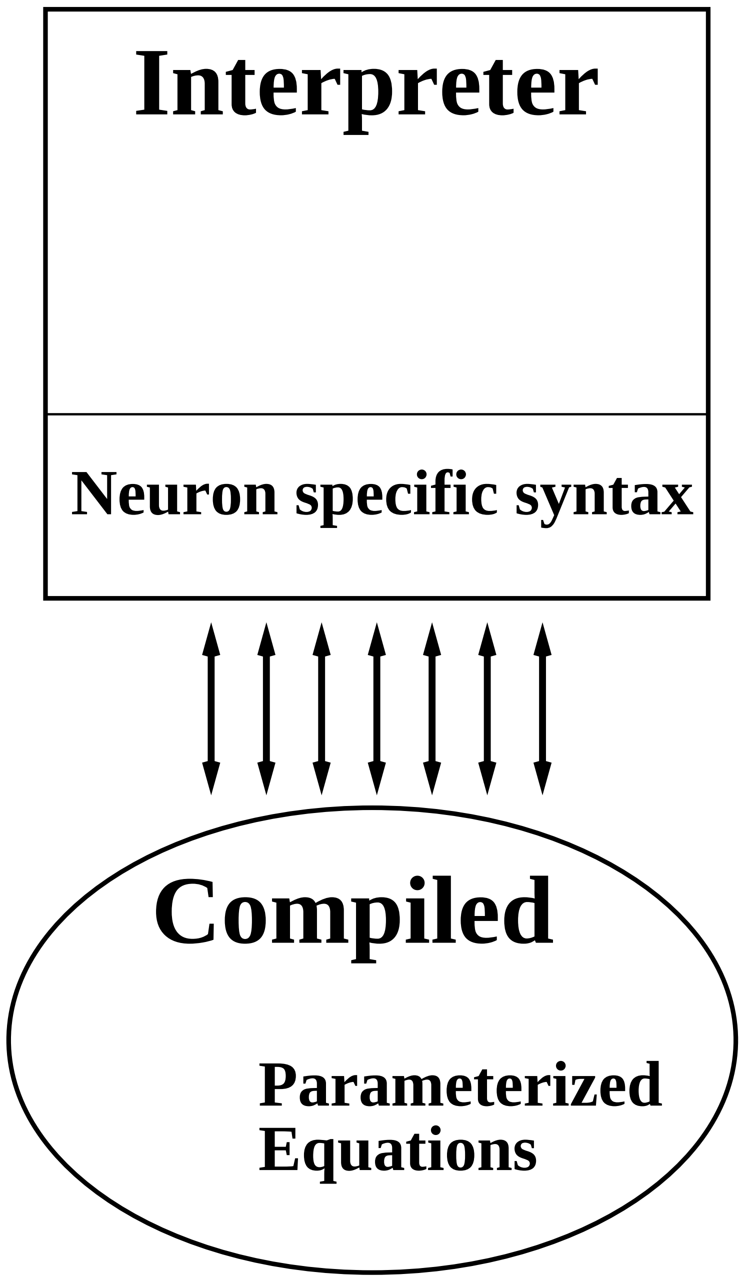
$$\begin{array}{lll} \frac{dm}{dt} = -\alpha_m m + \beta_m \cdot (1-m) & \alpha_m = \frac{.1(V+40)}{1-e^{-.1(V+40)}} & \beta_m = 4e^{-(V+65)/18} \\ \frac{dh}{dt} = -\alpha_h h + \beta_h \cdot (1-h) & \alpha_h = .07e^{-.05(V+65)} & \beta_h = \frac{1}{1+e^{-.1(V+35)}} \\ \frac{dn}{dt} = -\alpha_n n + \beta_n \cdot (1-n) & \alpha_n = \frac{.01(V+55)}{1-e^{-.1(V+55)}} & \beta_n = .125e^{-(V+65)/80} \end{array}$$

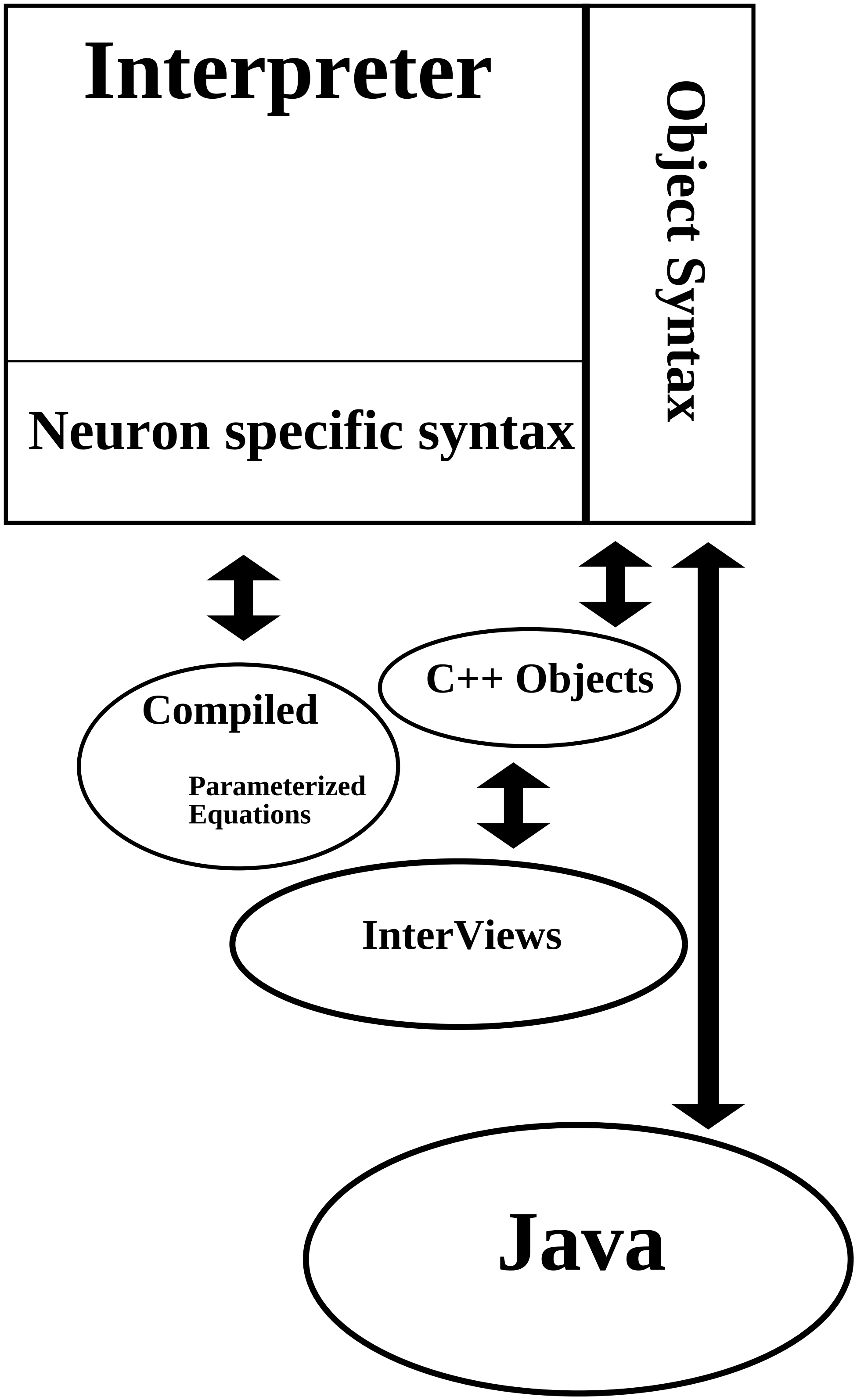
Simulation

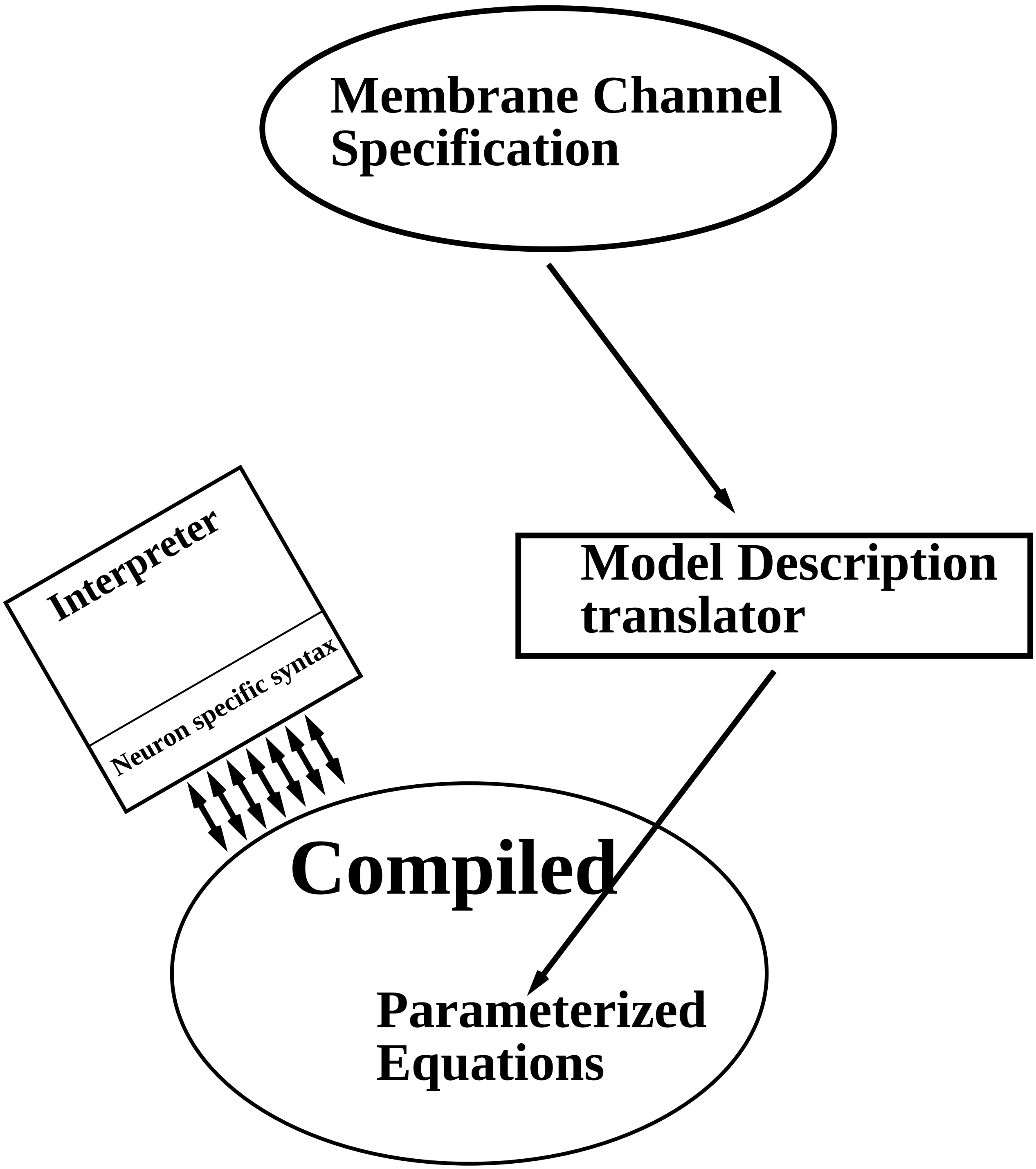
Representation

```
create axon
axon {
        nseg = 50
        diam = 100
        L = 20000
        insert hh
}
```

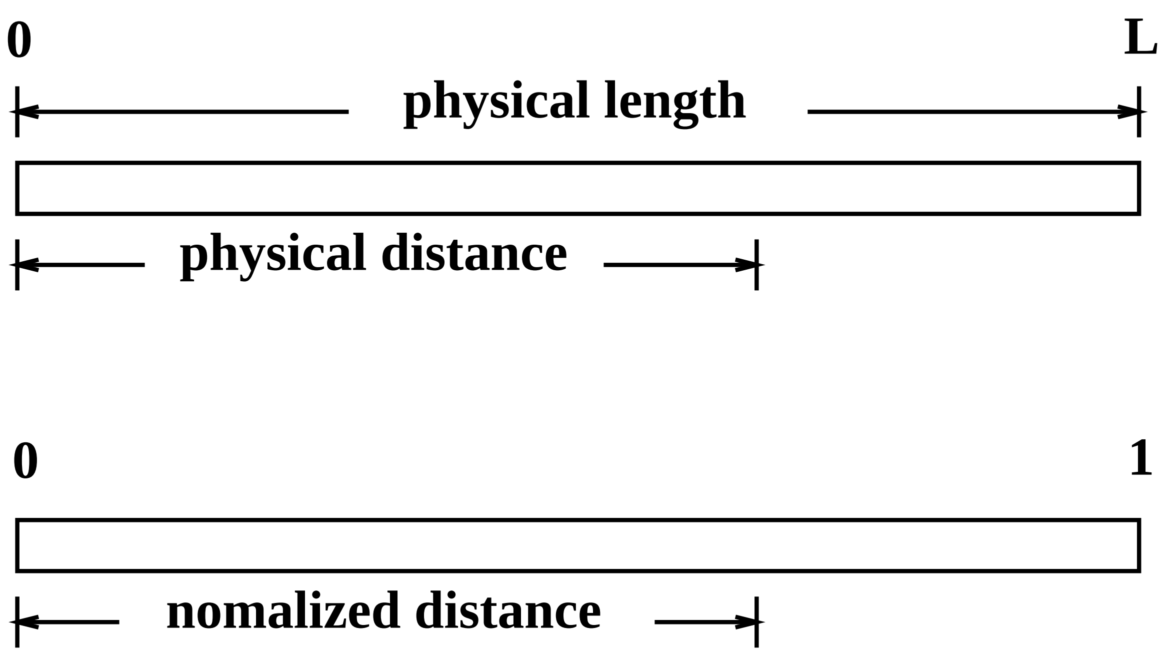

Interpreted
Setup
Control
Presentation
Functions
Variables
Compiled
Parameterized
Equations

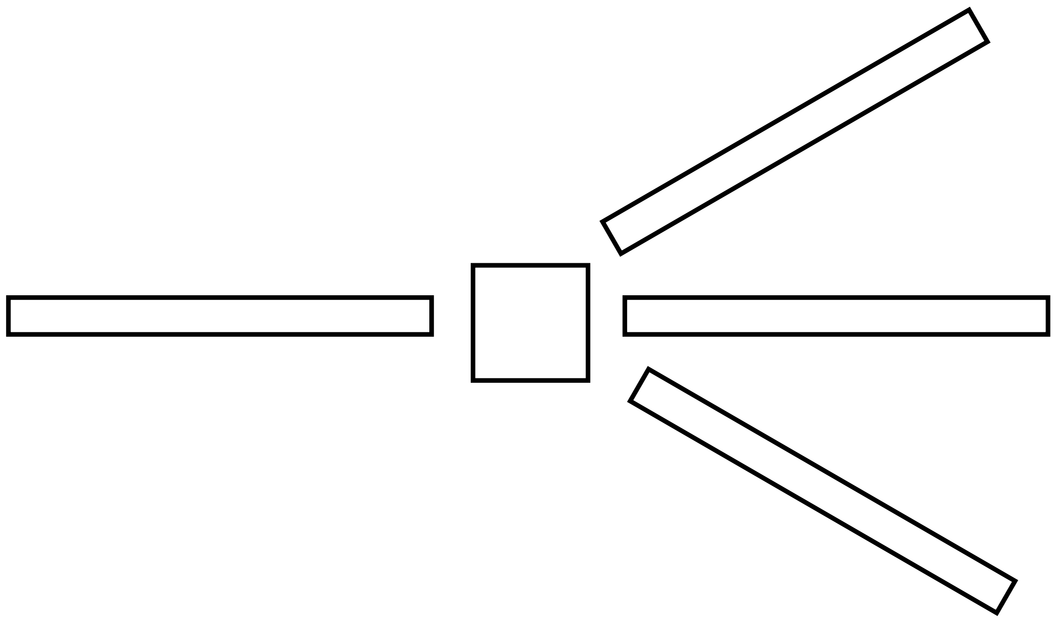

Interpreter
Neuron specific syntax
Compiled
Parameterized
Equations


Interpreter
Neuron specific syntax
Object Syntax
C++ Objects
Compiled
Parameterized
Equations
InterViews
Java


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

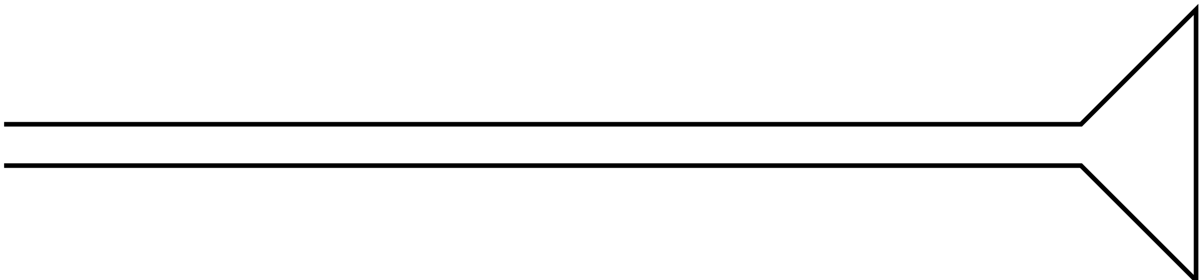
Cable Section



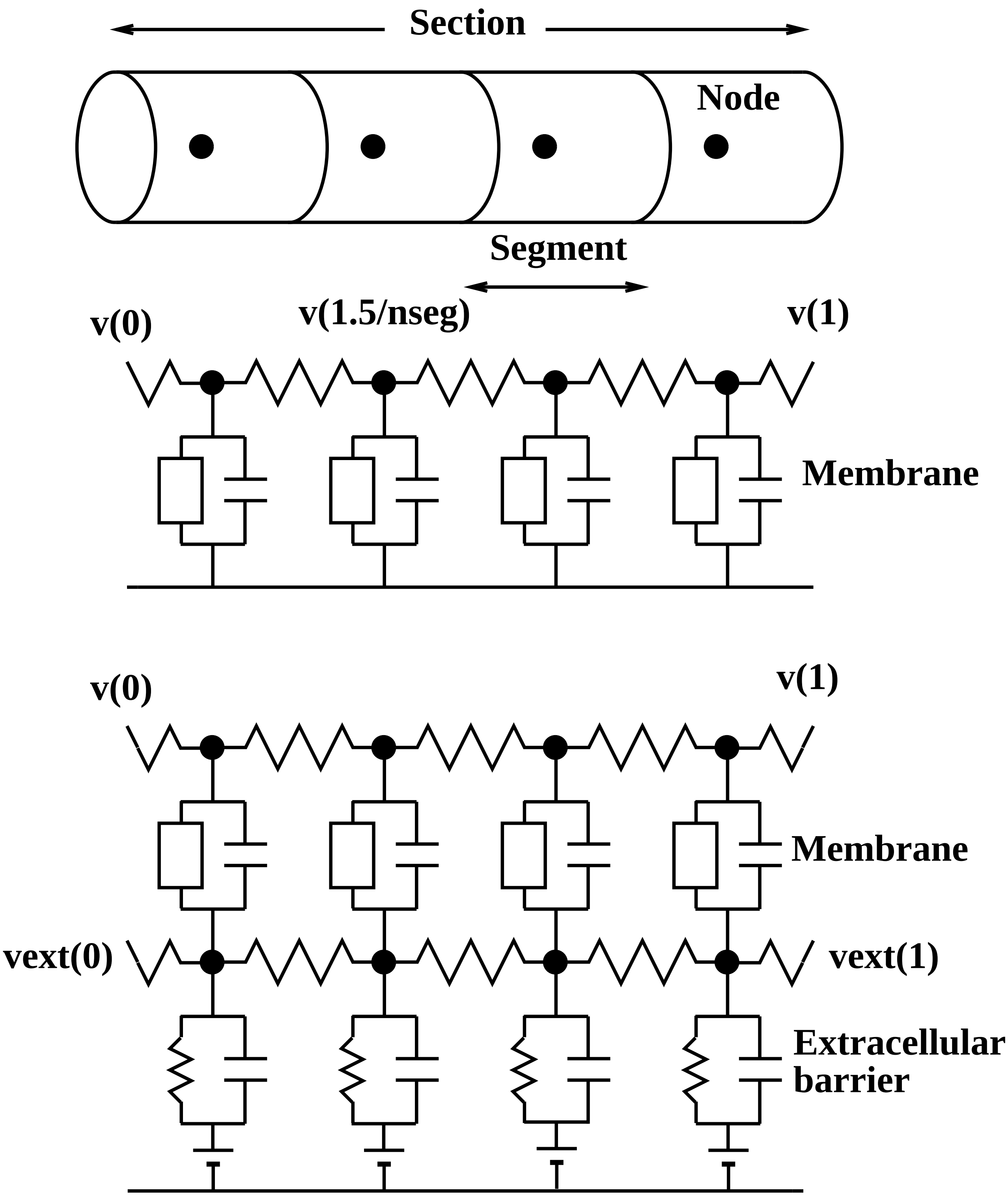


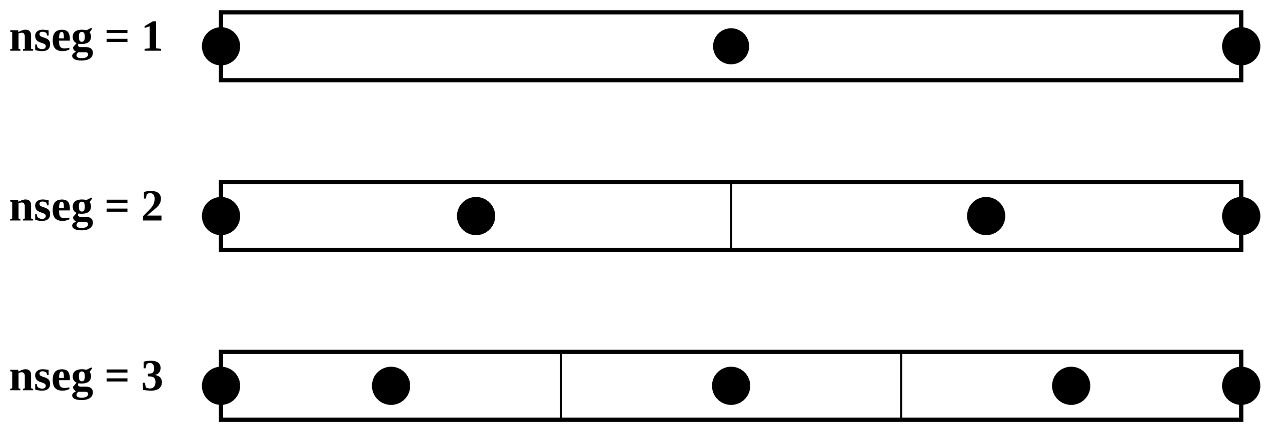
```
create axon, soma, dendrite[3]
connect axon(0), soma(0)
for i=0,2 connect soma(1), dendrite[i](0)
```

```
axon.L = 1000
axon {
  insert hh
  gnabar_hh = .120
}
```

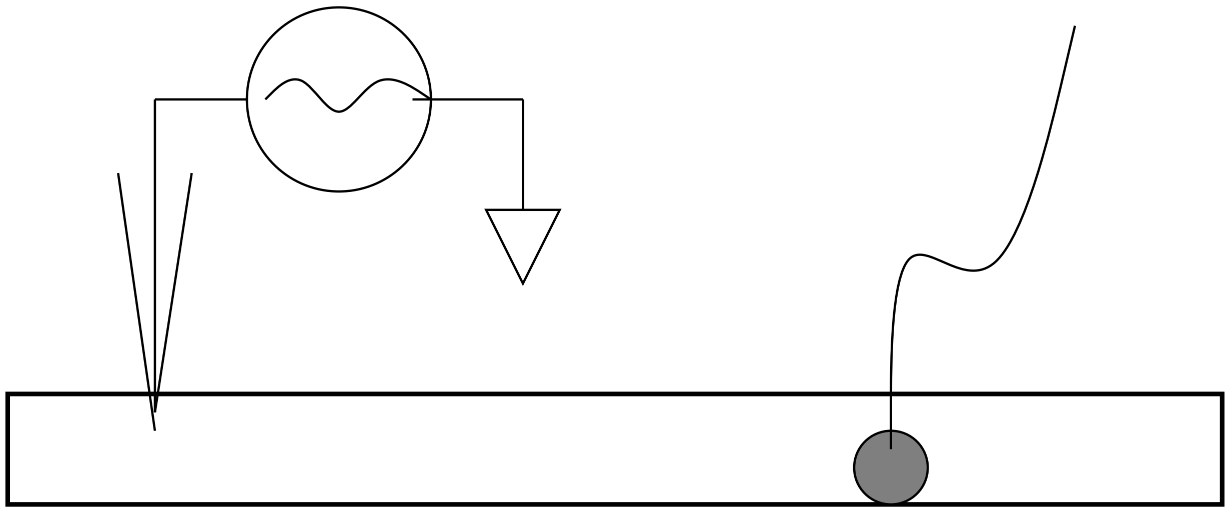


```
nseg = 20
diam(0:.1) = 20:2
diam(.1:1) = 2:2
```


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

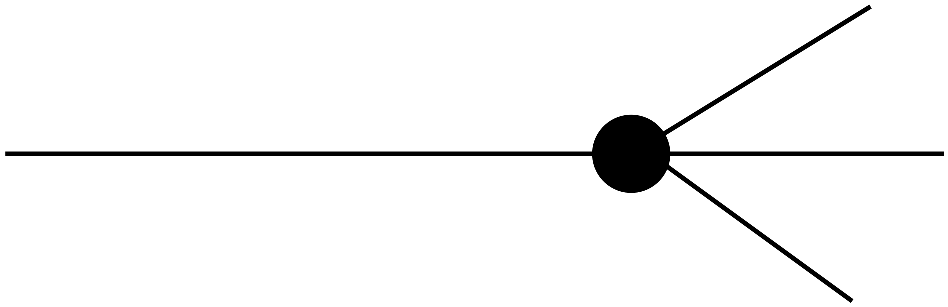



```
forall nseg *= 3
```



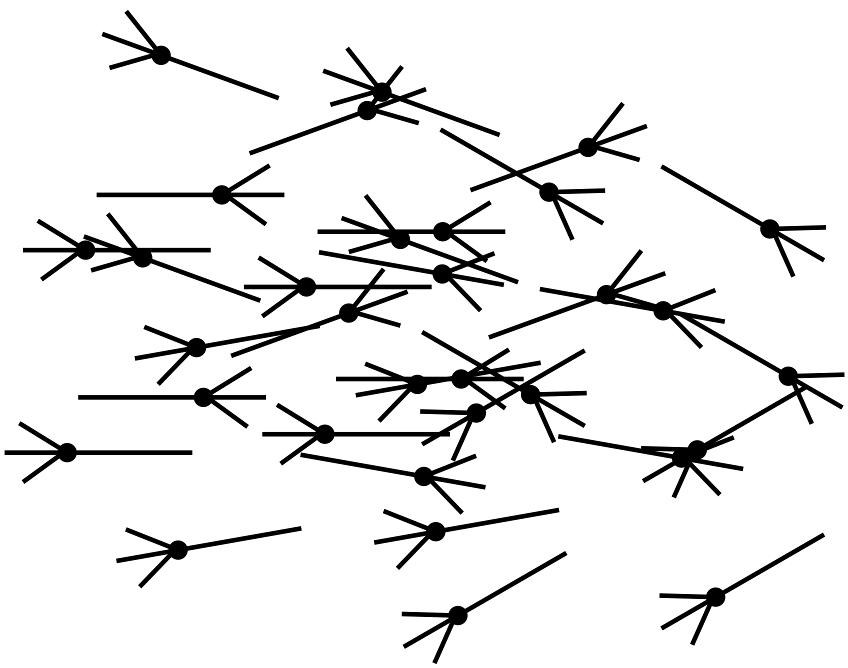
```
objref stim, syn
stim = new IClamp(.1)
syn = new Synapse(.7)


  stim.dur = .1
  stim.amp = 5

  setpointer syn.vpre, cell[15].axon.v(1)
```

```
begintemplate Cell
  public axon, soma, dendrite[3]
```



```
endtemplate Cell
```

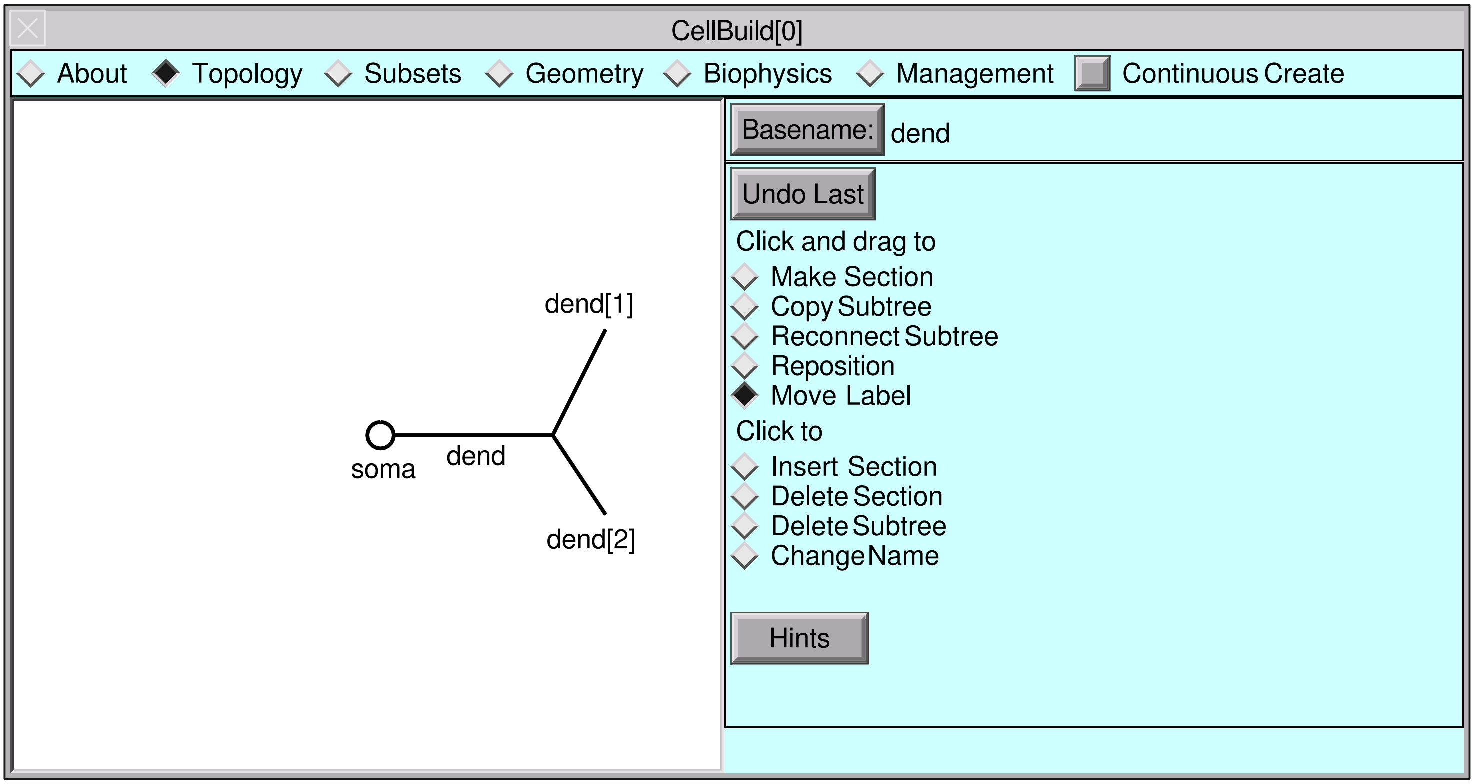
```
objectvar cell[32]
for i=0,31  cell[i] = new Cell()
```

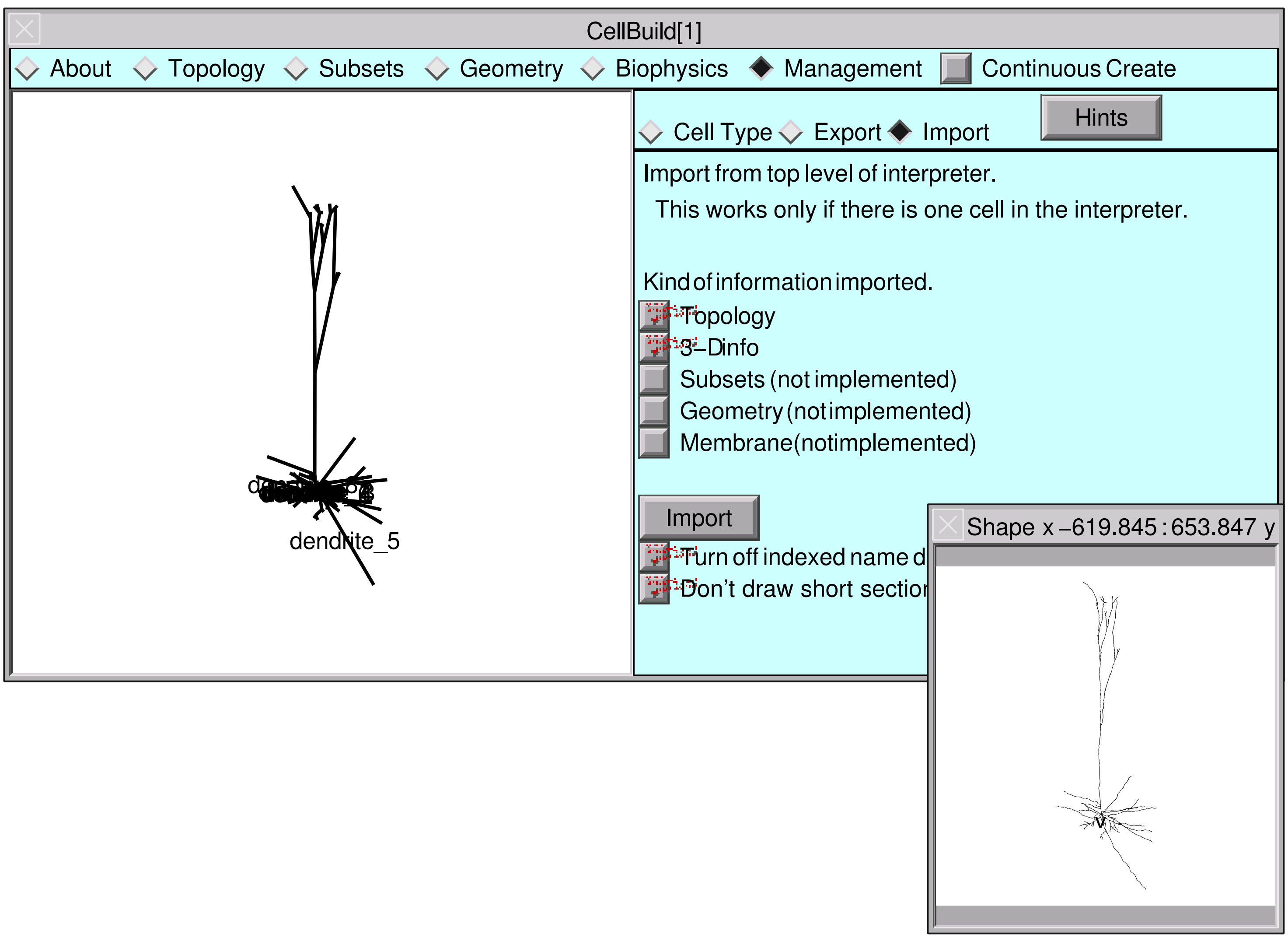


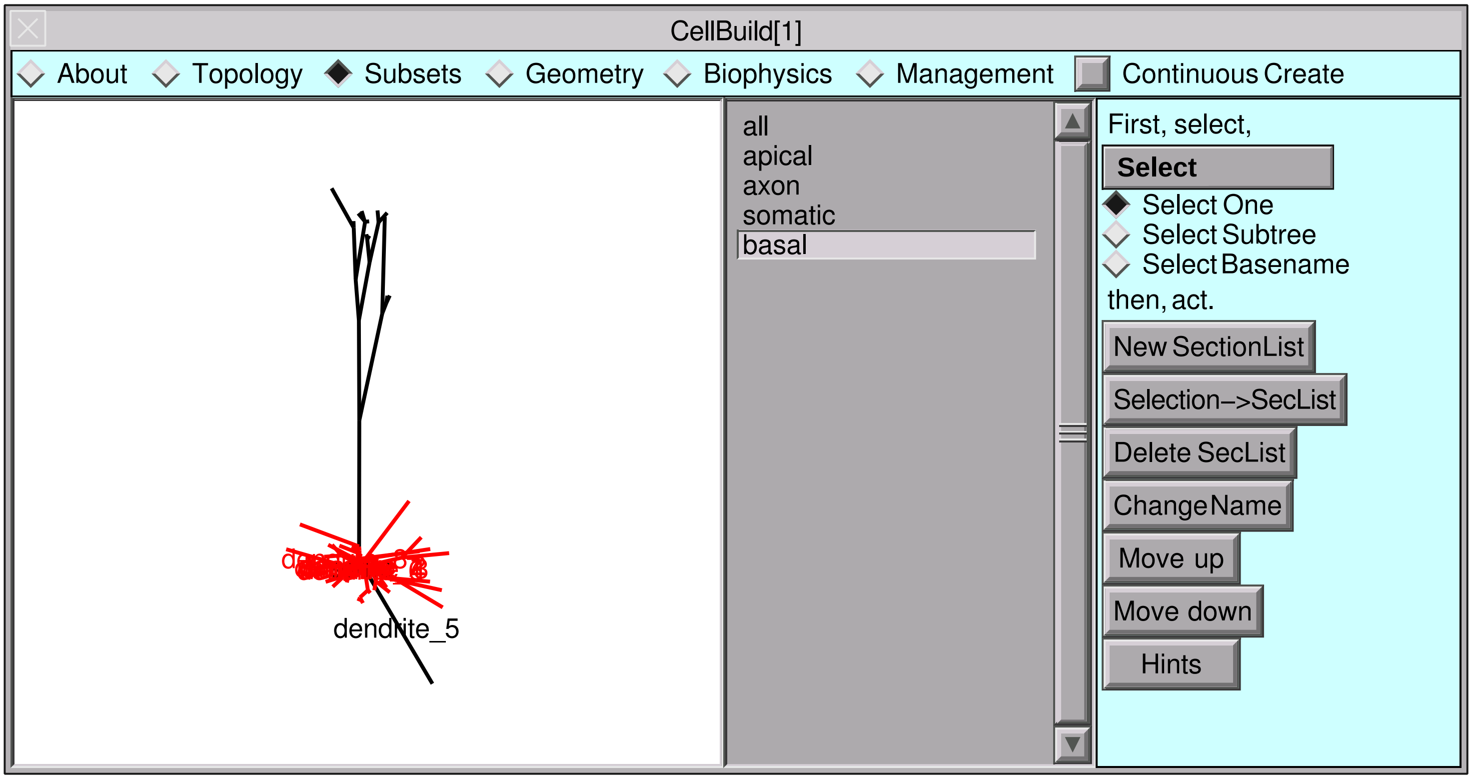
```
cell[15].axon.v(1)
```

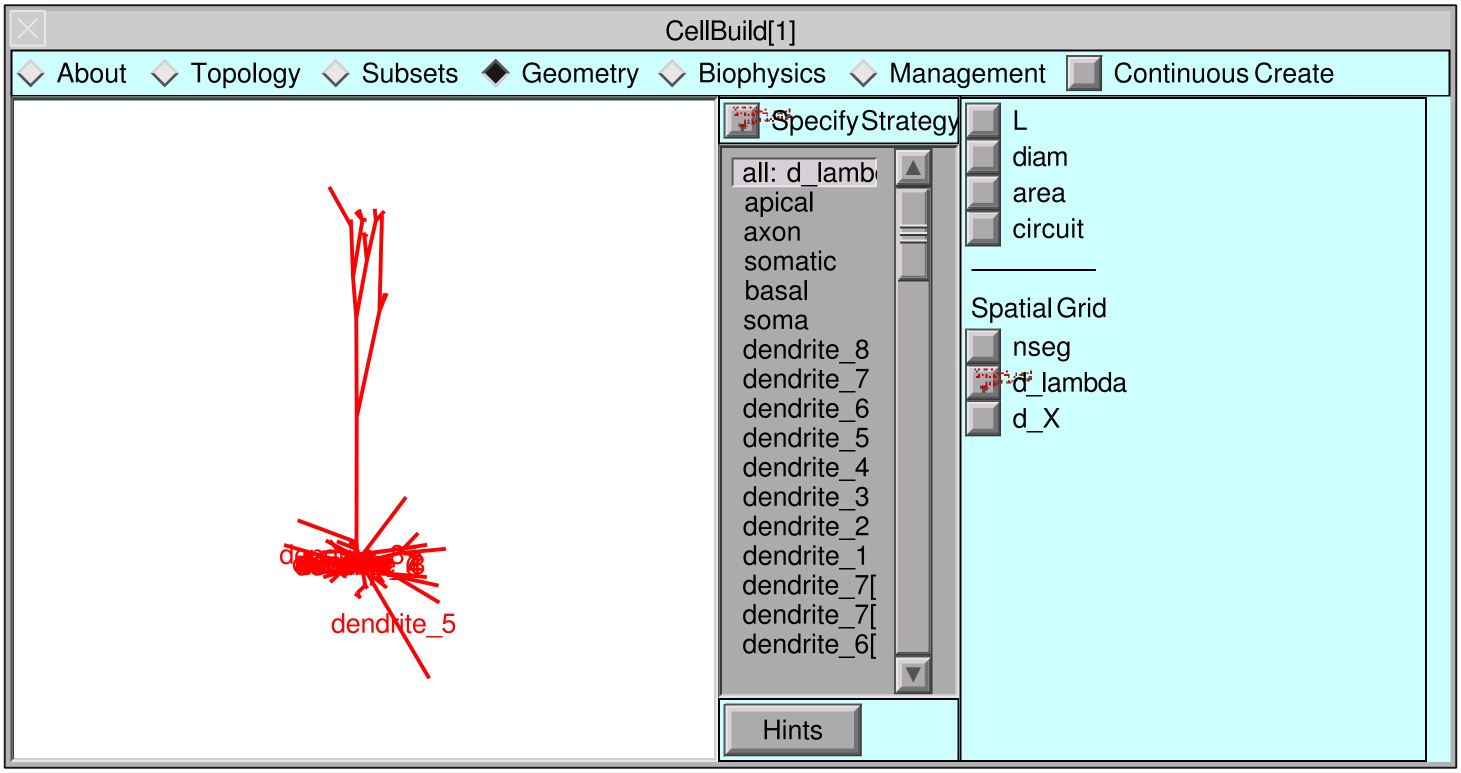
```
proc run() {
  finitialize(-65)
  while (t < tstop) {
    fadvance()
  }
}
```

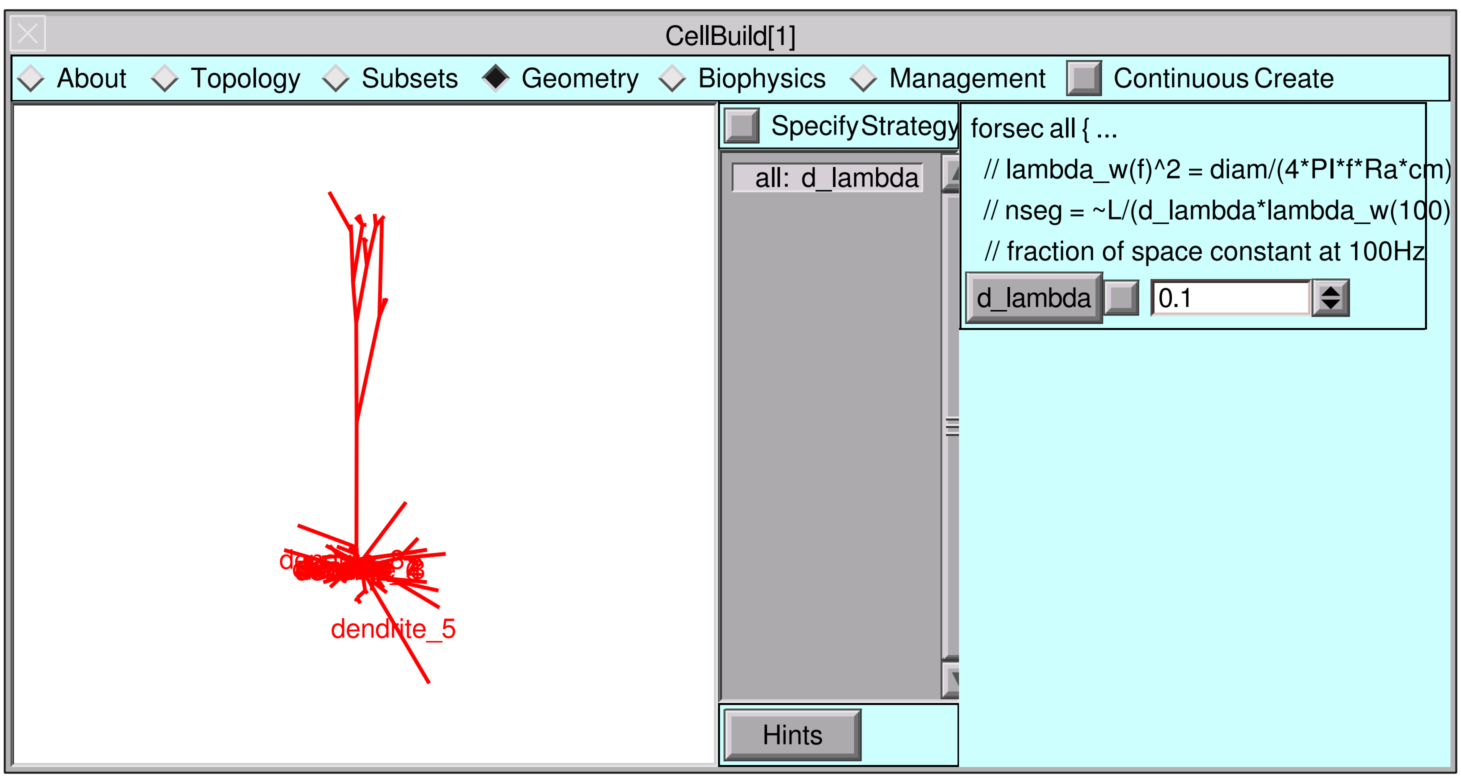
$$t \longrightarrow t + dt$$

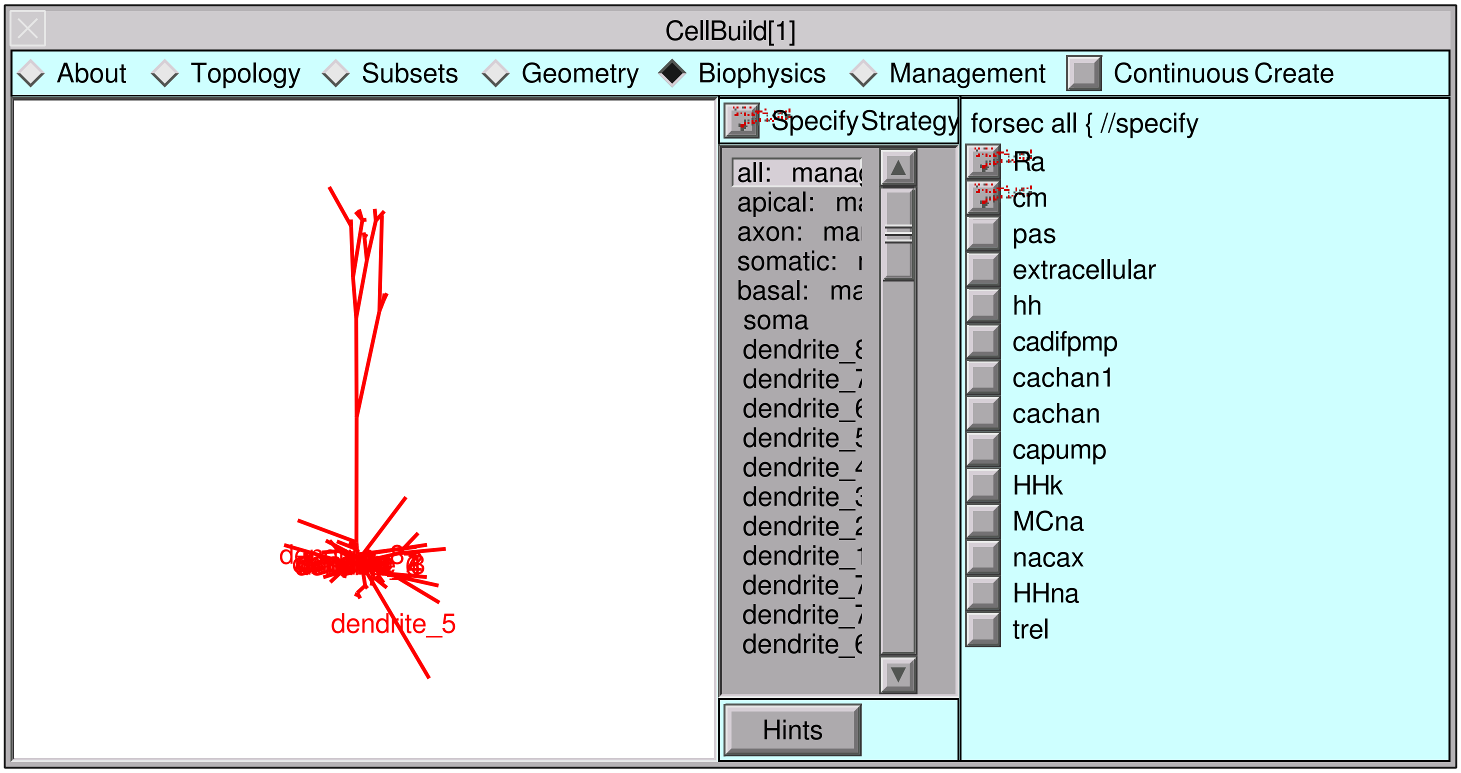

CellBuild[0]
About
Topology
Subsets
Geometry
Biophysics
Management
Continuous Create
Basename:
dend
Undo Last
Click and drag to
Make Section
Copy Subtree
Reconnect Subtree
Reposition
Move Label
Click to
Insert Section
Delete Section
Delete Subtree
Change Name
Hints
dend[1]
soma
dend
dend[2]

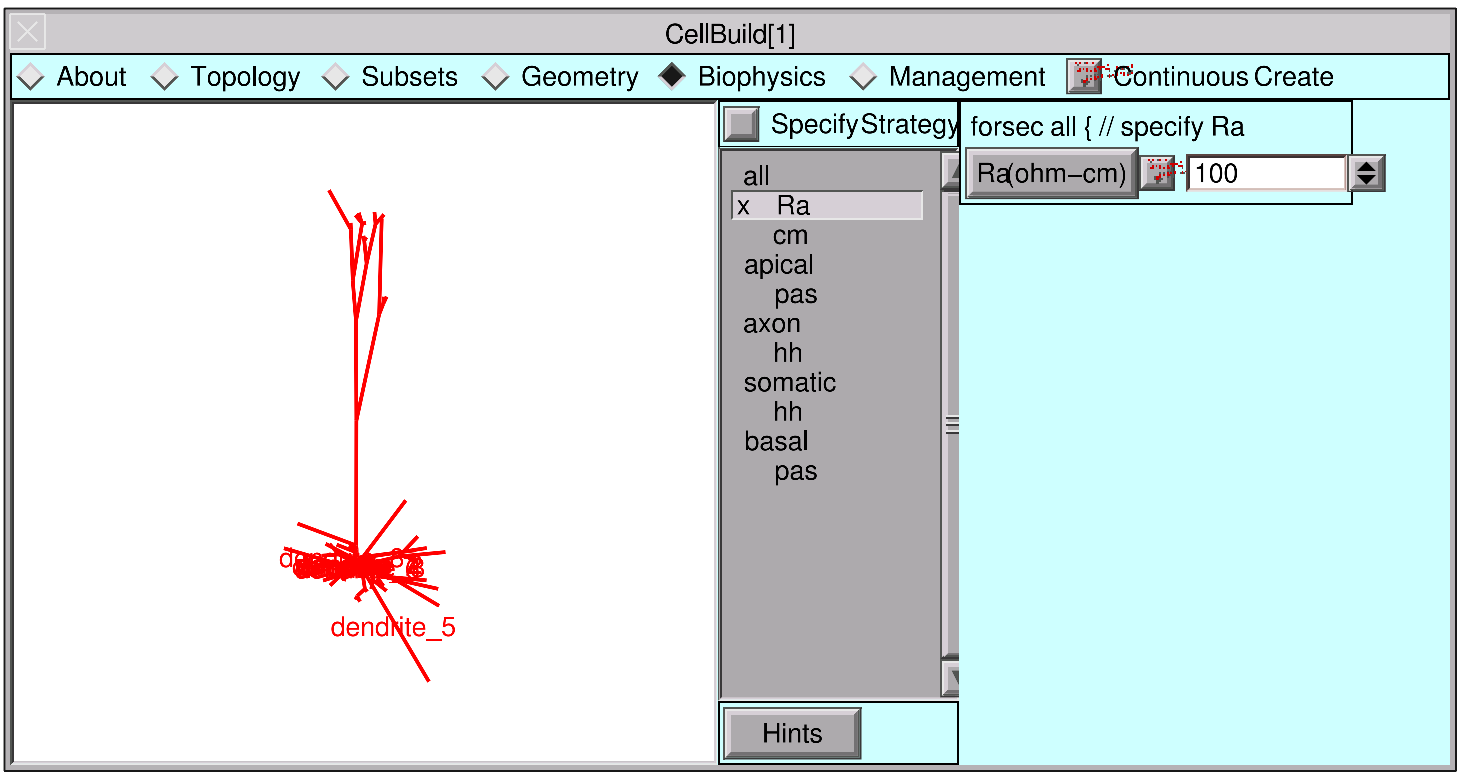

CellBuild[1]
About
Topology
Subsets
Geometry
Biophysics
Management
Continuous Create
Cell Type
Export
Import
Hints
Import from top level of interpreter.
This works only if there is one cell in the interpreter.
Kind of information imported.
Topology
3-Dinfo
Subsets (not implemented)
Geometry (not implemented)
Membrane (not implemented)
Import
Turn off indexed name d
Don't draw short sectio
dendrite_5
Shape x -619.845 : 653.847 y

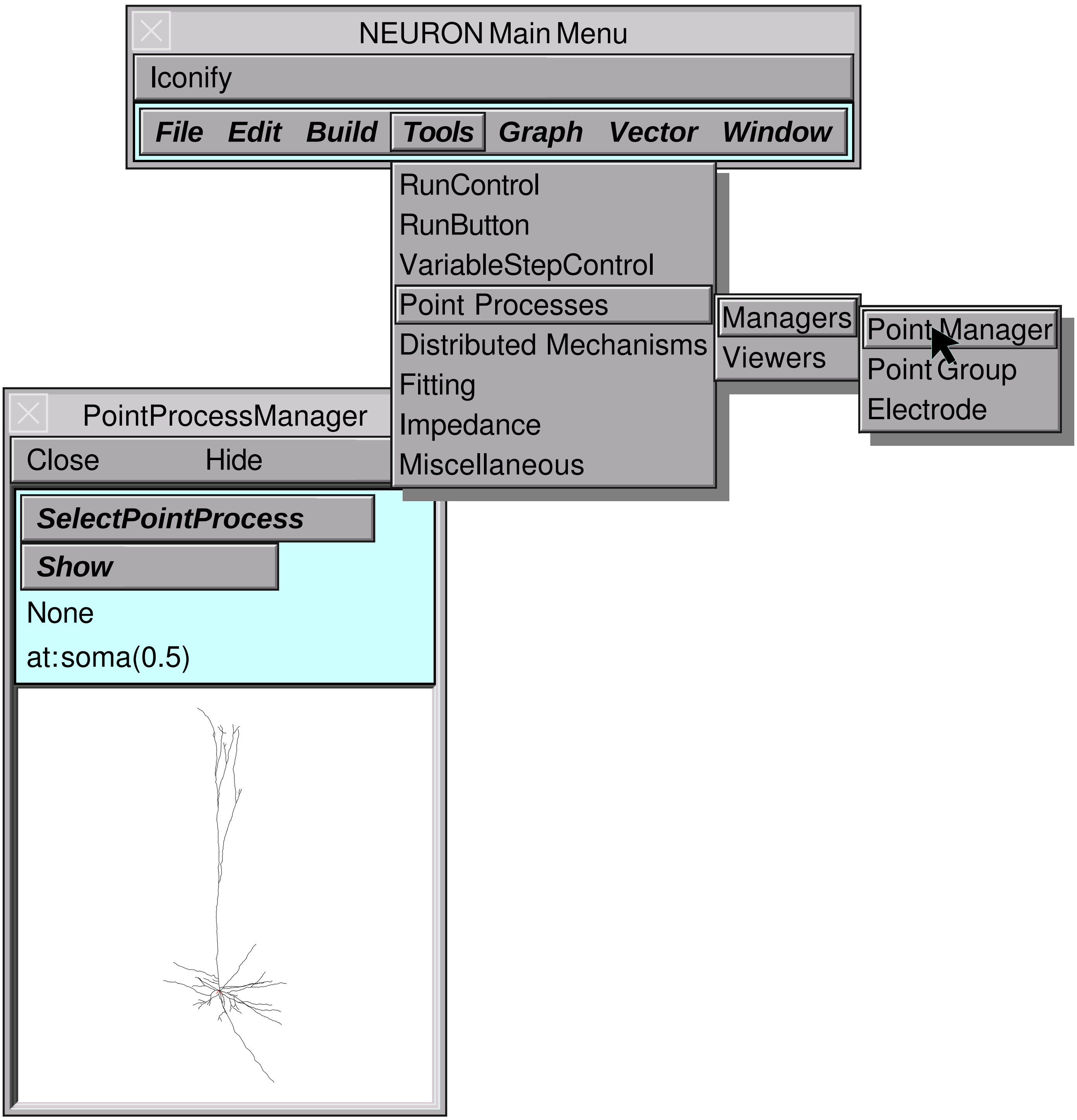

CellBuild[1]
About
Topology
Subsets
Geometry
Biophysics
Management
Continuous Create
dendrite_5
all
apical
axon
somatic
basal
First, select,
Select
Select One
Select Subtree
Select Basename
then, act.
New SectionList
Selection->SecList
Delete SecList
Change Name
Move up
Move down
Hints

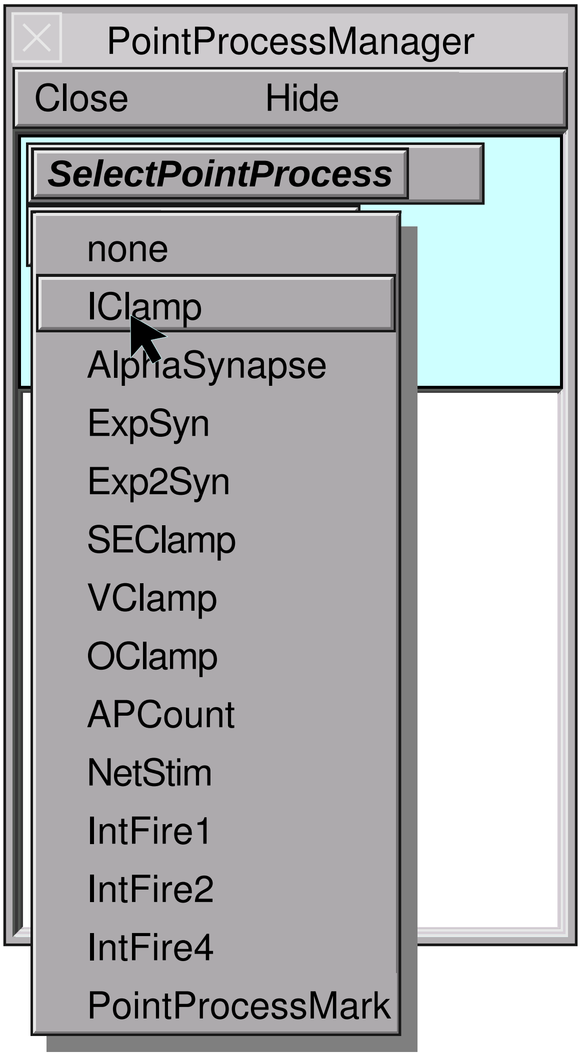

CellBuild[1]
About
Topology
Subsets
Geometry
Biophysics
Management
Continuous Create
Specify Strategy
all: d_lamb
apical
axon
somatic
basal
soma
dendrite_8
dendrite_7
dendrite_6
dendrite_5
dendrite_4
dendrite_3
dendrite_2
dendrite_1
dendrite_7[
dendrite_7[
dendrite_6[
Hints
L
diam
area
circuit
Spatial Grid
nseg
d_lambda
d_X
dendrite_5

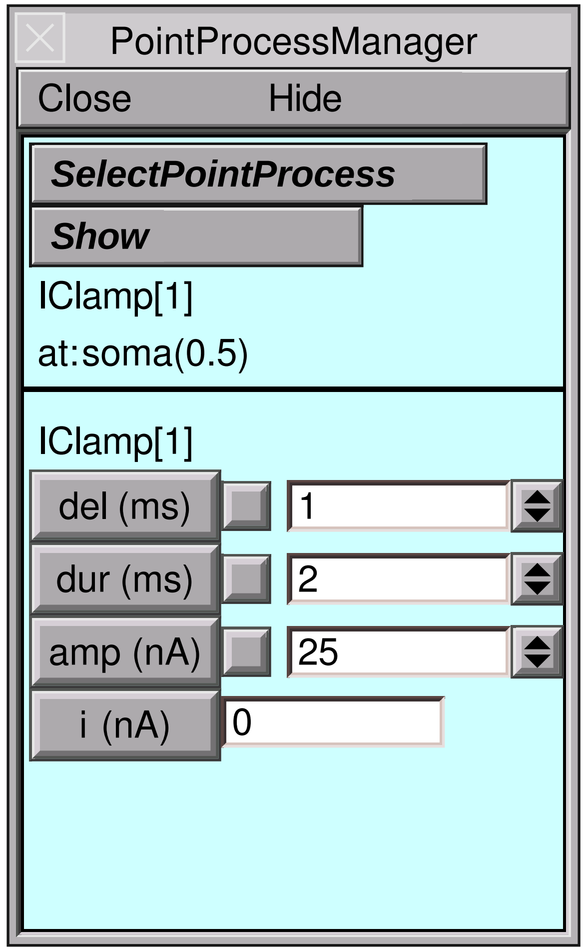

CellBuild[1]
About
Topology
Subsets
Geometry
Biophysics
Management
Continuous Create
Specify Strategy
all: d_lambda
Hints
forsec all { ...
// lambda_w(f)^2 = diam/(4*PI*f*Ra*cm)
// nseg = ~L/(d_lambda*lambda_w(100)
// fraction of space constant at 100Hz
d_lambda
0.1
dendrite_5

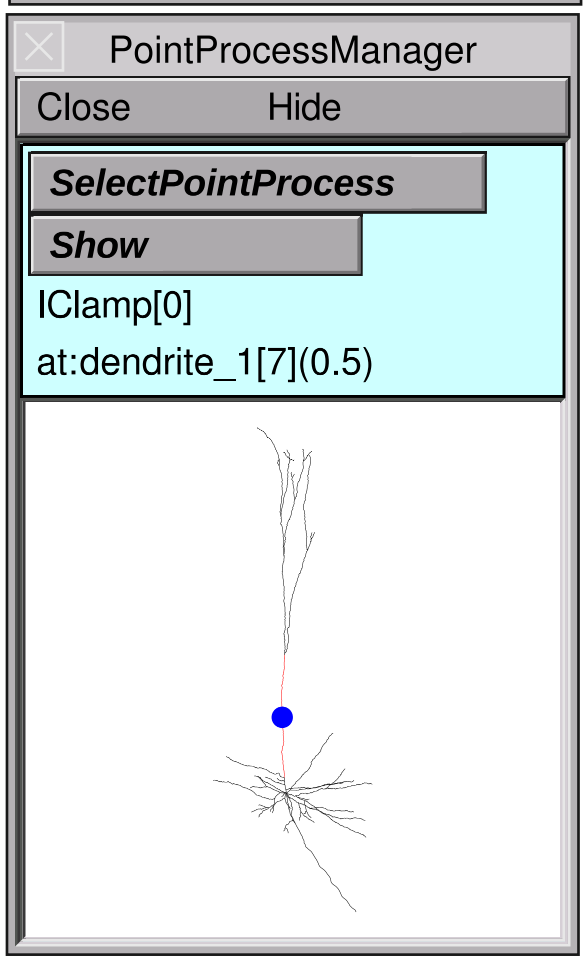

CellBuild[1]
About
Topology
Subsets
Geometry
Biophysics
Management
Continuous Create
SpecifyStrategy
forsec all { //specify
Ra
cm
pas
extracellular
hh
cadifpmp
cachan1
cachan
capump
HHk
MCna
nacax
HHna
trel
Hints
dendrite_5

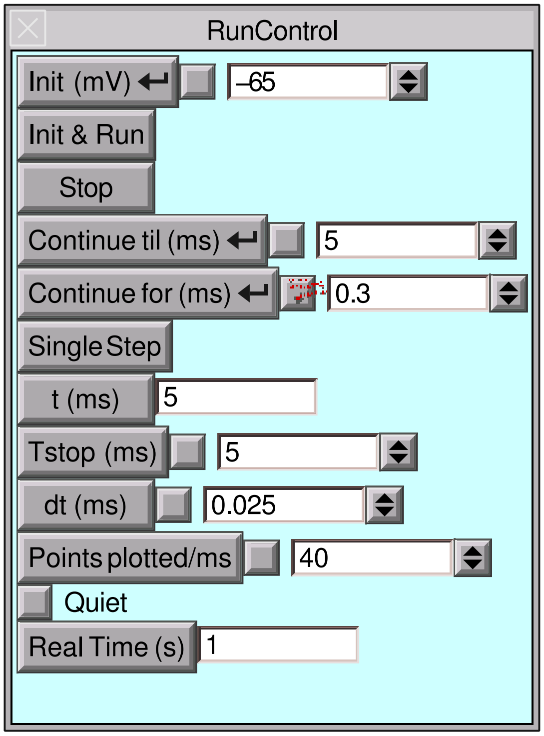

CellBuild[1]
About
Topology
Subsets
Geometry
Biophysics
Management
Continuous Create
SpecifyStrategy
forsec all { // specify Ra
Ra(ohm–cm)
100
all
x Ra
cm
apical
pas
axon
hh
somatic
hh
basal
pas
Hints
dendrite_5

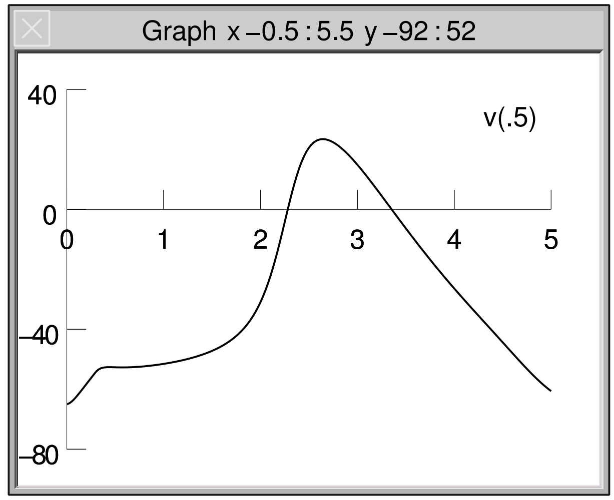

NEURON Main Menu
Iconify
File
Edit
Build
Tools
Graph
Vector
Window
RunControl
RunButton
VariableStepControl
Point Processes
Distributed Mechanisms
Fitting
Impedance
Miscellaneous
Managers
Viewers
Point Manager
Point Group
Electrode
PointProcessManager
Close
Hide
SelectPointProcess
Show
None
at:soma(0.5)

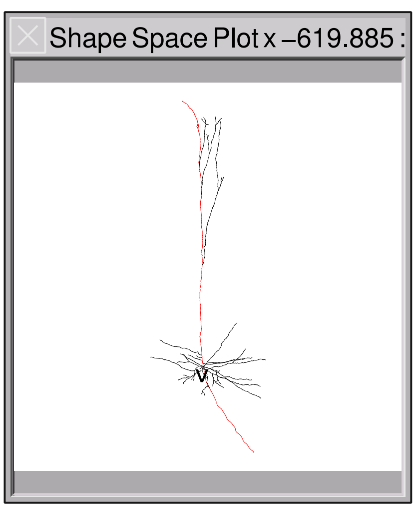

PointProcessManager
Close
Hide
SelectPointProcess
none
IClamp
AlphaSynapse
ExpSyn
Exp2Syn
SEClamp
VClamp
OClamp
APCount
NetStim
IntFire1
IntFire2
IntFire4
PointProcessMark

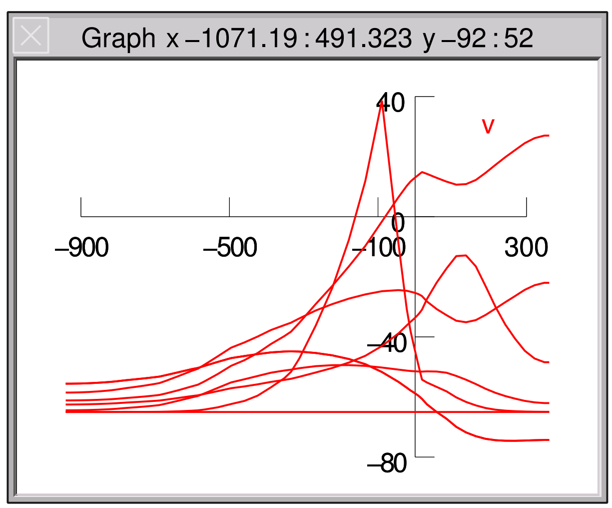

PointProcessManager
Close
Hide
SelectPointProcess
Show
IClamp[1]
at:soma(0.5)
IClamp[1]
del (ms)
1
dur (ms)
2
amp (nA)
25
i (nA)
0

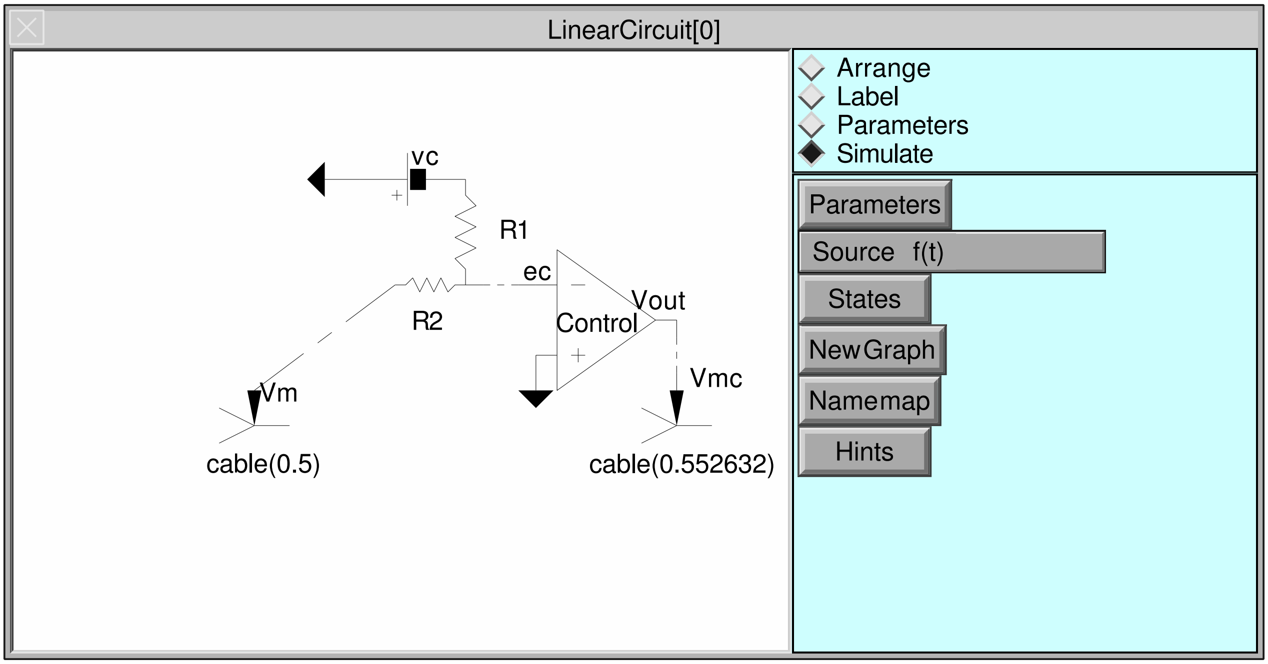

PointProcessManager
Close
Hide
SelectPointProcess
Show
IClamp[0]
at:dendrite_1[7](0.5)


RunControl
Init (mV)
–65
Init & Run
Stop
Continue til (ms)
5
Continue for (ms)
0.3
Single Step
t (ms)
5
Tstop (ms)
5
dt (ms)
0.025
Points plotted/ms
40
Quiet
Real Time (s)
1


Graph x –0.5 : 5.5 y –92 : 52
40
0
–40
–80
0
1
2
3
4
5
v(.5)


Shape Space Plot x –619.885 :
v


Graph x –1071.19 : 491.323 y –92 : 52
40
0
–40
–80
–900
–500
–100
300
v



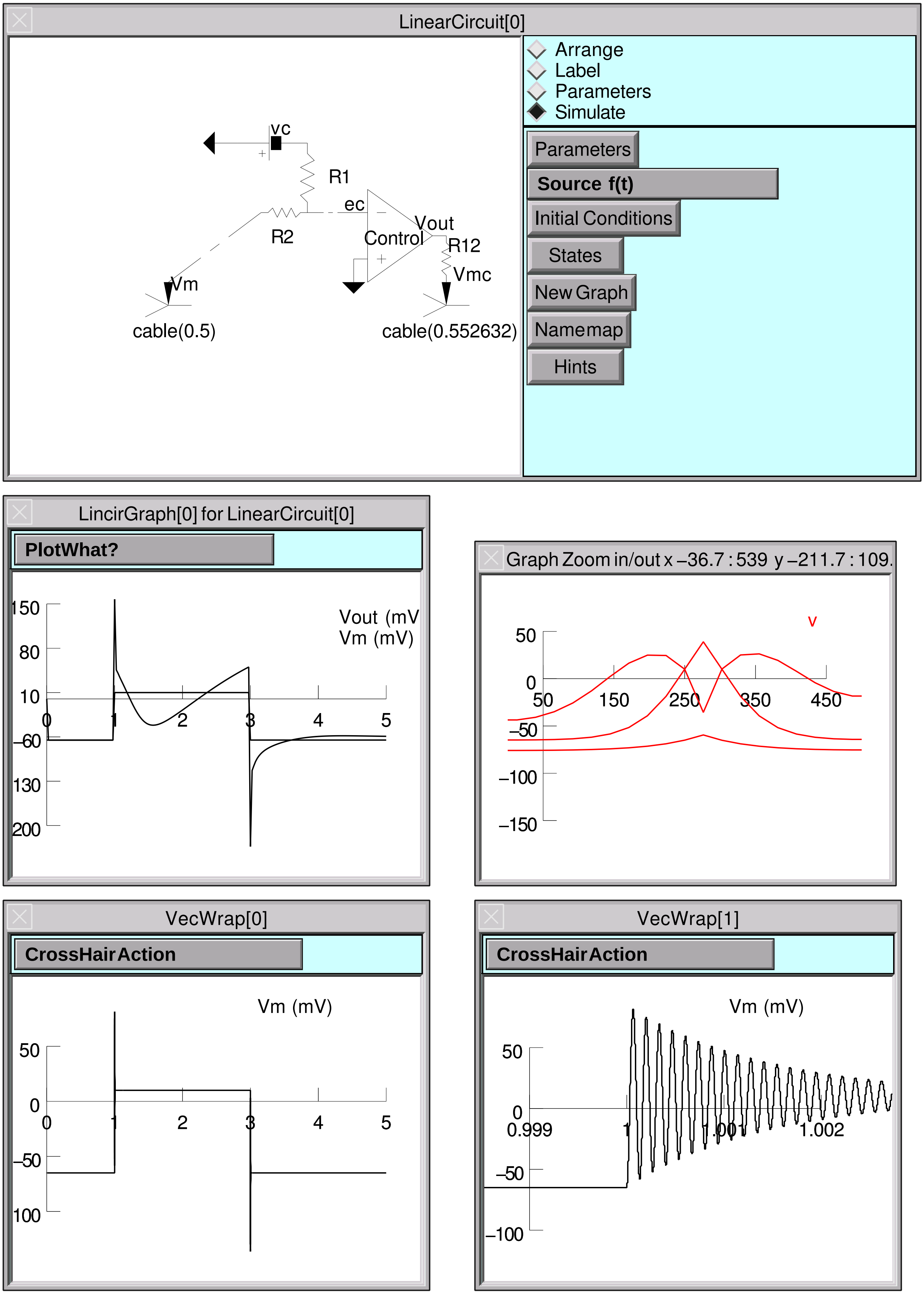
Cable

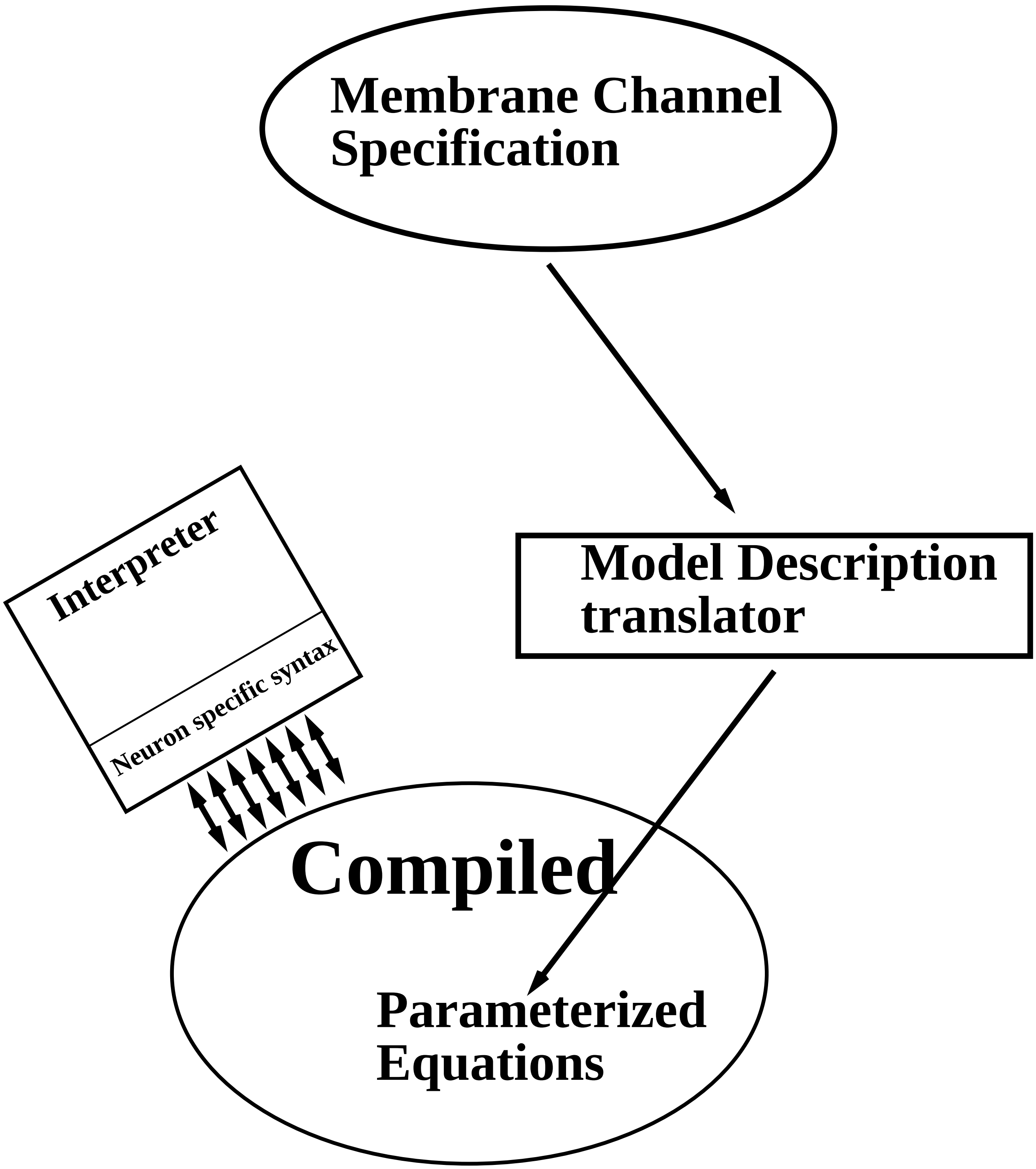
$$\begin{bmatrix} d & a & & & \\ b & d & a & & \\ & b & d & a & \\ & & b & d & a \\ & & & b & d \end{bmatrix} \quad \begin{bmatrix} v1 \\ vm \\ v3 \\ vmc \\ v5 \end{bmatrix}$$

C y' + G y = b

$$\begin{bmatrix} x & & & & & \\ & x & & & x & \\ x & & x & x & & \\ & & x & x & & x \\ & & x & & & \\ & & & x & & \end{bmatrix} \quad \begin{bmatrix} vm \\ vmc \\ ec \\ vc \\ IC \\ Ivc \end{bmatrix}$$

$$\begin{bmatrix} d & a & & & & & & & \\ b & d & a & & & & & & \\ & b & d & a & & & & & \\ & & b & d & a & & & x & \\ & & & b & d & & & & \\ & x & & & & x & x & & \\ & & & & & x & x & & x \\ & & & & & x & & & \\ & & & & & & x & & \end{bmatrix} \quad \begin{bmatrix} v1 \\ vm \\ v3 \\ vmc \\ v5 \\ ec \\ vc \\ IC \\ Ivc \end{bmatrix}$$


LinearCircuit[0]
Arrange
Label
Parameters
Simulate
Parameters
Source f(t)
Initial Conditions
States
New Graph
Namemap
Hints
vc
R1
ec
R2
Vout
Control
R12
Vmc
Vm
cable(0.5)
cable(0.552632)
LincirGraph[0] for LinearCircuit[0]
PlotWhat?
Vout (mV
Vm (mV)
150
80
10
-60
130
200
0
1
2
3
4
5
Graph Zoom in/out x –36.7 : 539 y –211.7 : 109.
v
50
0
–50
–100
–150
50
150
250
350
450
VecWrap[0]
CrossHair Action
Vm (mV)
50
0
–50
100
0
1
2
3
4
5
VecWrap[1]
CrossHair Action
Vm (mV)
50
0
–50
–100
0.999
1
1.001
1.002



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

NMODL

NEURON Model Description Language

Add new membrane mechanisms to NEURON

Density mechanisms

- Distributed Channels
- Ion accumulation

Point Processes

- Electrodes
- Synapses

Described by

- Differential equations
- Kinetic schemes
- Algebraic equations

Benefits

- Specification only -- independent of solution method.
- Efficient -- translated into C.
- Compact
  - One NMODL statement -> many C statements.
  - Interface code automatically generated.
- Consistent ion current/concentration interactions.
- Consistent Units

NMODL general block structure

What the model looks like from outside

```
NEURON {
        SUFFIX kchan
        USEION k  READ ek WRITE ik
        RANGE gbar, ...
}
```

What names are manipulated by this model

```
UNITS { mV = (millivolt) ... }

PARAMETER { gbar = .036 (mho/cm2) <0, 1e9>... }

STATE { n ... }

ASSIGNED { ik (mA/cm2) ... }
```

Initial default values for states

```
INITIAL {
        rates(v)
        n = ninf
}
```

Calculate currents (if any) as function of v, t, states

(and specify how states are to be integrated)

```
BREAKPOINT {
        SOLVE deriv METHOD cnexp
        ik = gbar * n^4 * (v - ek)
}
```

State equations

```
DERIVATIVE deriv {
        rates(v)
        n' = (ninf - n)/ntau
}
```

Functions and procedures

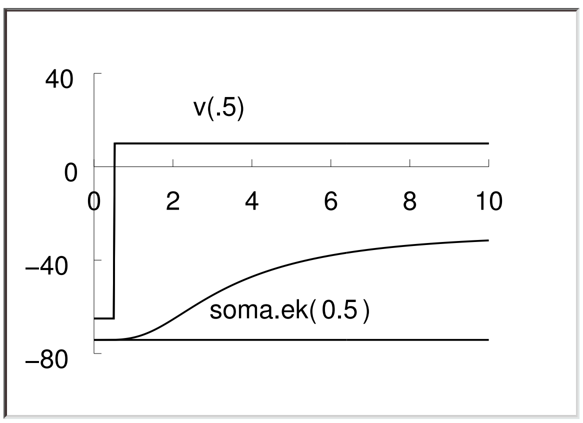
```
PROCEDURE rates(v(mV)) {
        ...
}
```

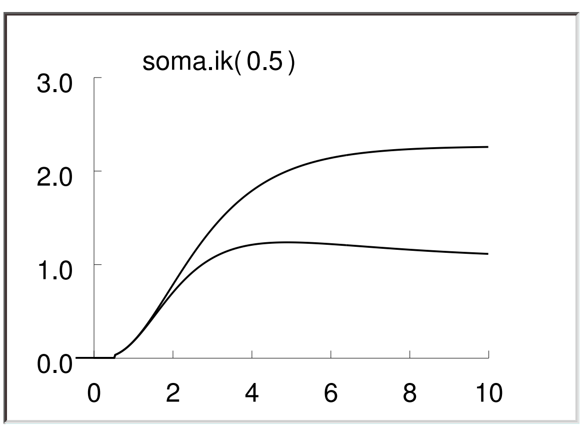
Ion Channel

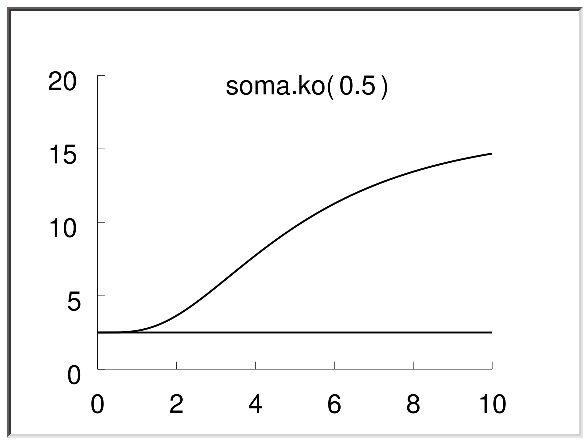
```
NEURON {
  USEION k READ ek WRITE ik
}
BREAKPOINT {
  SOLVE states METHOD cnexp
  ik = gbar*n*n*n*n*(v - ek)
}
DERIVATIVE states {
  rate(v*1(/mV))
  n’ = (inf - n)/tau
}
```

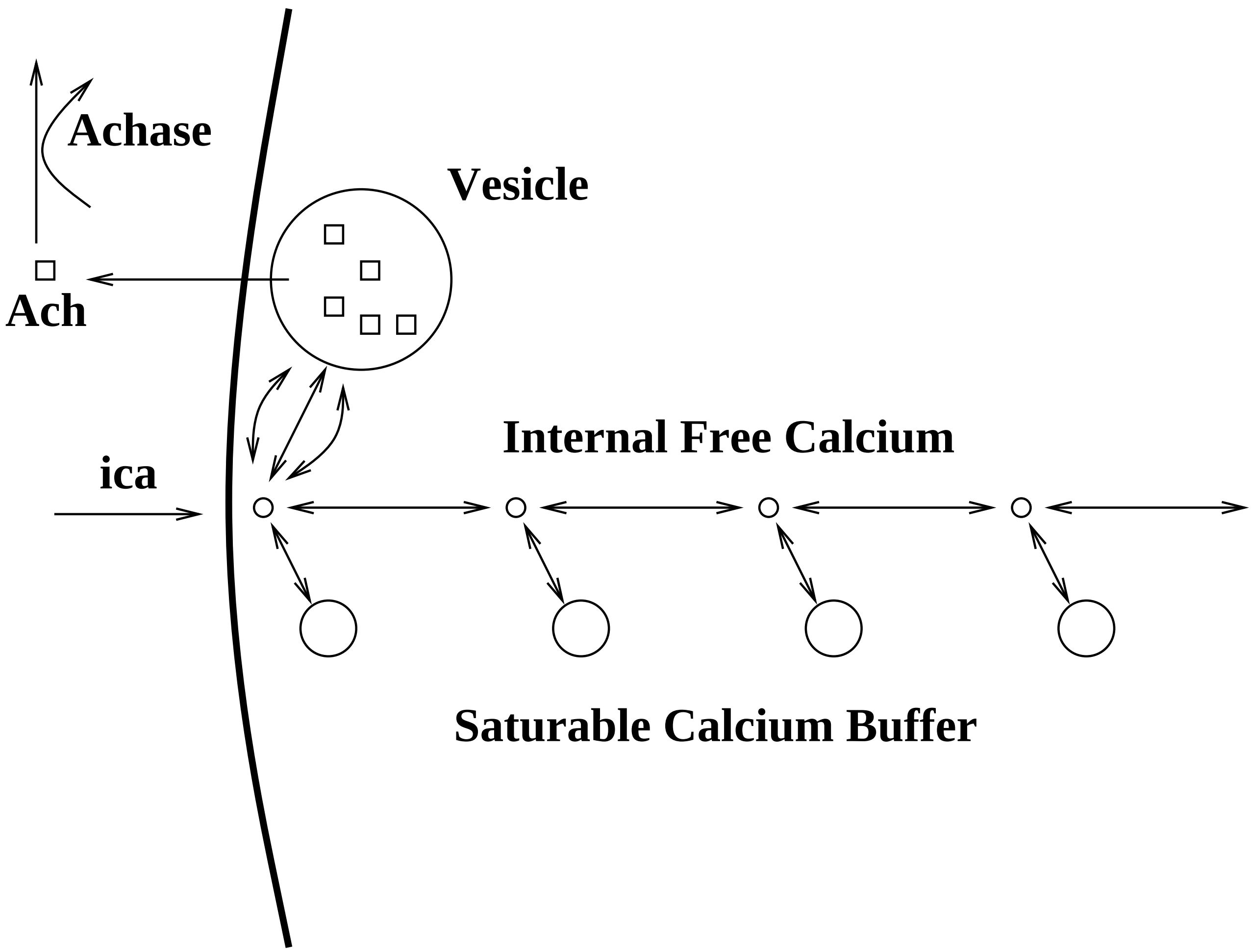
Ion Accumulation

```
NEURON {
  USEION k READ ik WRITE ko
}
BREAKPOINT {
  SOLVE state METHOD cnexp

}
DERIVATIVE state {
  ko’ = ik/fhspace/F*(1e8)
        + k*(kbath - ko)
}
```










```
STATE {
Vesicle Ach Achase Ach2ase X Buffer[N] CaBuffer[N] Ca[N]
}

KINETIC calcium_evoked_release {
    : release
 ~  Vesicle + 3Ca[0] <-> Ach   (Agen, Arev)
 ~  Ach + Achase <-> Ach2ase   (Aase2, 0)  :idiom for enzyme reaction
 ~  Ach2ase <-> X + Achase     (Aase2, 0)  : requires two reactions

   : Buffering
   FROM i = 0 TO N-1 {
    ~  Ca[i] + Buffer[i] <-> CaBuffer[i]   (kCaBuffer, kmCaBuffer)
   }

   :Diffusion
   FROM i = 1 TO N-1 {
    ~  Ca[i-1] <-> Ca[i]      (Dca*a[i-1], Dca*b[i])
   }

    : inward flux
 ~  Ca[0] <<     (ica)
}
```

UNITS Checking

```
NEURON { POINT_PROCESS Shunt ... }

PARAMETER {
        e = 0 (millivolt)
        r = 1 (gigaohm) <1e-9,1e9>
}

ASSIGNED {
        i (nanoamp)
        v (millivolt)
}

BREAKPOINT {
        i = (v - e)/r
}
```

Units are incorrect in the "`i = ...`" current assignment.
The output from

```
modlunit shunt
```

is:

```
Checking units of shunt.mod
The previous primary expression with units: 1-12 coul/sec
is missing a conversion factor and should read:
  (0.001)*()
 at line 14 in file shunt.mod
        i = (v - e)/r<>
```

To fix the problem replace the line with:

```
    i = (.001)*(v - e)/r
```

What conversion factor will make the following consistent?

```
  nai'  =   ina    /    FARADAY  * (c/radius)
(uM/ms)   (mA/cm2) / (coulomb/mole) / (um)
```