Network Construction

1) Define the types of cells

2) Create each cell in the network

3) Connect the cells

Define the types of cells

"Real" cell types

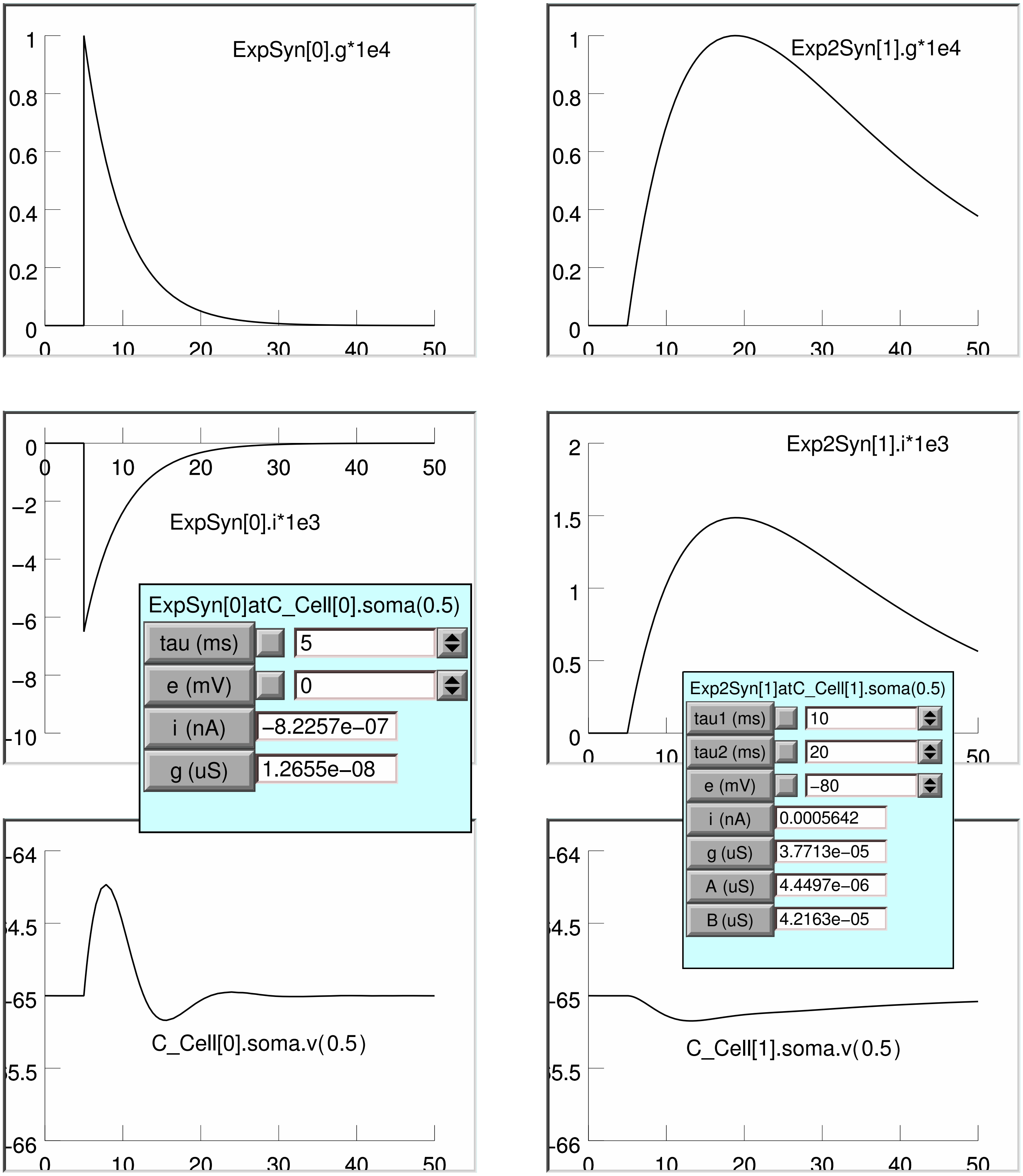
```
Sections + density mechanisms + synapses.

The latter are PointProcesses that
have a NET_RECEIVE block that affects
membrane current.

    ExpSyn
    Exp2Syn
```

Encapsulate in a class

```
begintemplate Cell
    public soma, E, I
    create soma
    objref E, I
    proc init() {
        soma insert hh
        soma { E = new ExpSyn(.5)  I = new ExpSyn(.5) }
        I.e = -80
    }
endtemplate Cell
```

Artificial cell types

```
PointProcesses that have a NET_RECEIVE
block that calls net_event

    NetStim
    IntFire1
    IntFire2
    IntFire4
```



ExpSyn[0]atC_Cell[0].soma(0.5)

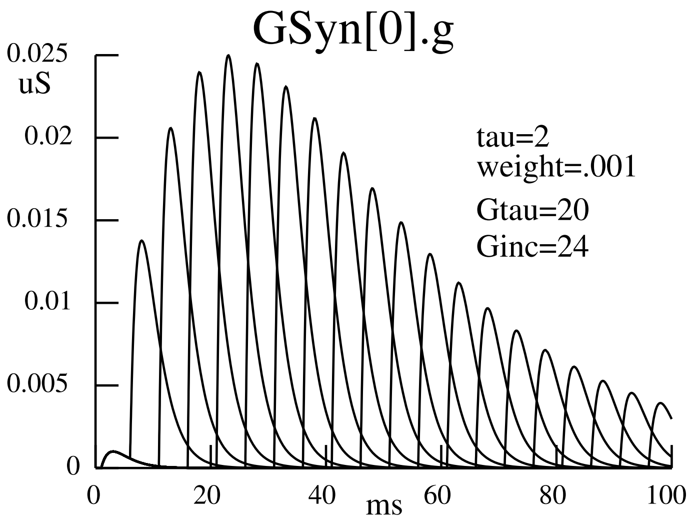
tau (ms)	5
e (mV)	0
i (nA)	–8.2257e–07
g (uS)	1.2655e–08

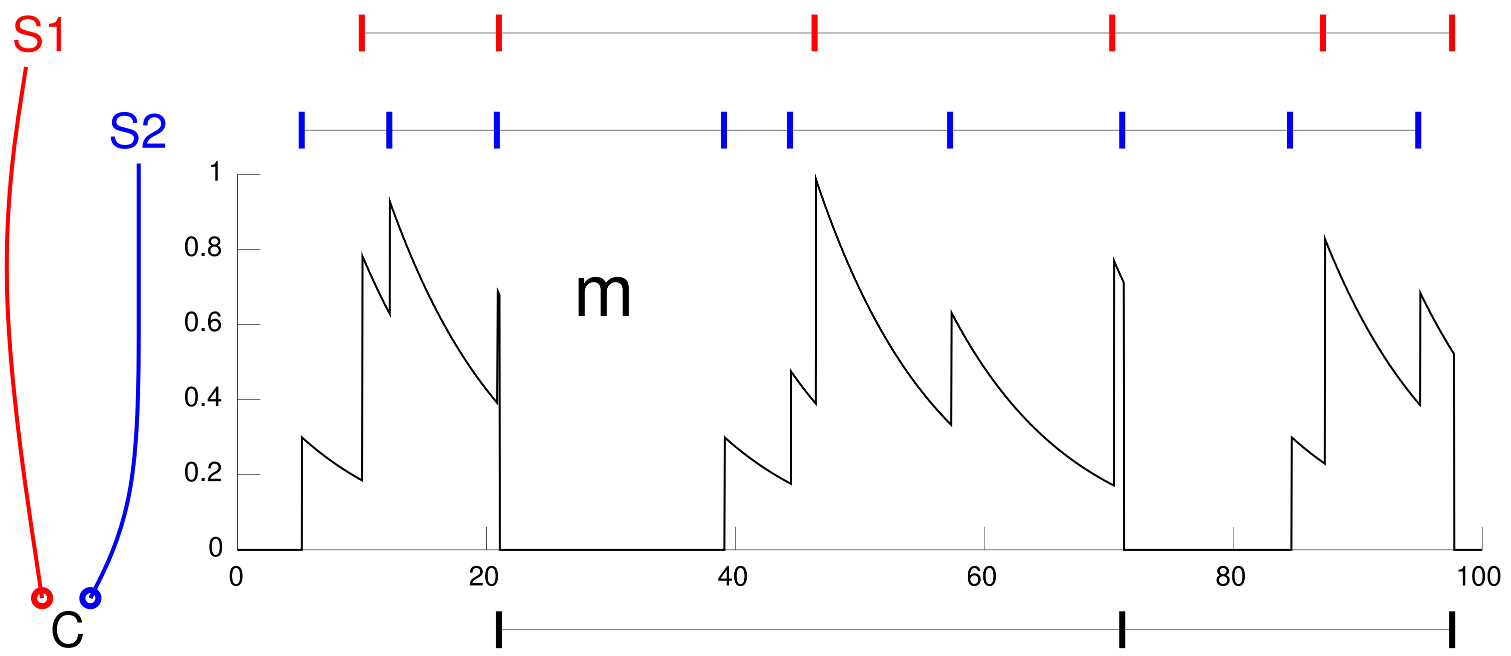
Exp2Syn[1]atC_Cell[1].soma(0.5)

tau1 (ms)	10
tau2 (ms)	20
e (mV)	–80
i (nA)	0.0005642
g (uS)	3.7713e–05
A (uS)	4.4497e–06
B (uS)	4.2163e–05

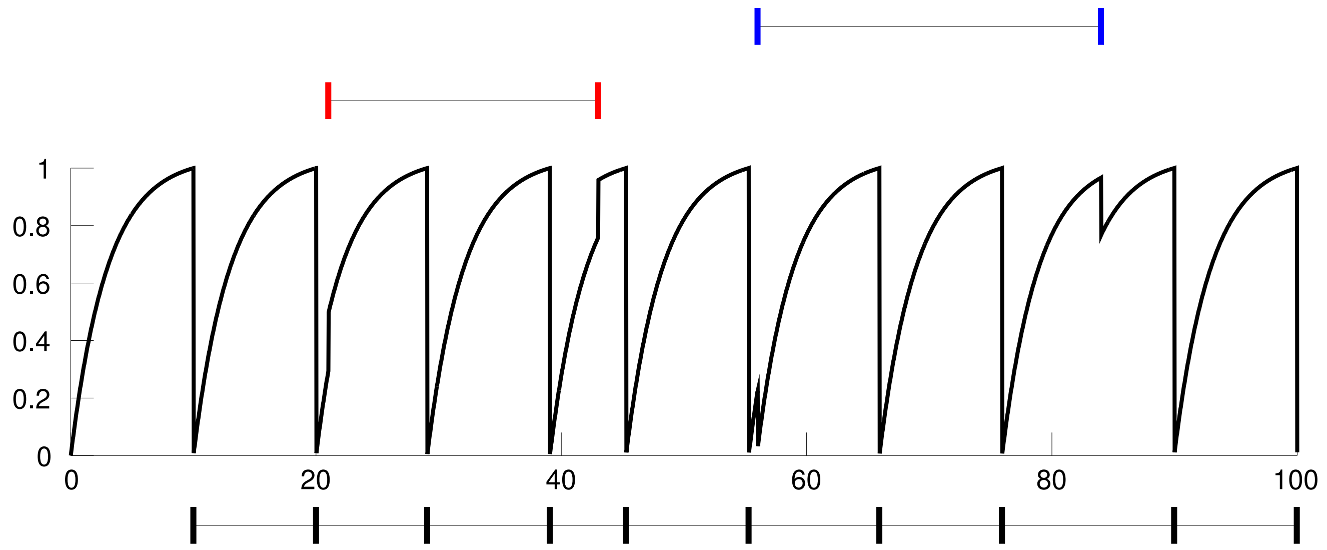
G-Protein synapse -- gsyn.mod

```
NEURON {
        POINT_PROCESS GSyn
        RANGE tau1, tau2, e, i
        RANGE Gtau1, Gtau2, Ginc
        NONSPECIFIC_CURRENT i
        RANGE g
}

PARAMETER {
        tau1=0.1 (ms)
        tau2 = 1 (ms)
        Gtau1 = 20 (ms)
        Gtau2 = 21 (ms)
        Ginc = 1
        e=0     (mV)
}

STATE {
        A (umho)
        B (umho)
}

INITIAL {
        LOCAL tp
        A = 0
        B = 0
        tp = (tau1*tau2)/(tau2 - tau1) * log(tau2/tau1)
        factor = -exp(-tp/tau1) + exp(-tp/tau2)
        factor = 1/factor
        tp = (Gtau1*Gtau2)/(Gtau2 - Gtau1) * log(Gtau2/Gtau1)
        Gfactor = -exp(-tp/Gtau1) + exp(-tp/Gtau2)
        Gfactor = 1/Gfactor
}

BREAKPOINT {
        SOLVE state METHOD cnexp
        g = B - A
        i = g*(v - e)
}

DERIVATIVE state {
        A' = -A/tau1
        B' = -B/tau2
}

NET_RECEIVE(weight (umho), w, G1, G2, t0 (ms)) {
        INITIAL { G1 = 0  G2 = 0  t0 = 0 }
        G1 = G1*exp(-(t-t0)/Gtau1)
        G2 = G2*exp(-(t-t0)/Gtau2)
        G1 = G1 + Ginc*Gfactor
        G2 = G2 + Ginc*Gfactor
        t0 = t
        w = weight*(1 + G2 - G1)
        state_discontinuity(A, A + w*factor)
        state_discontinuity(B, B + w*factor)
}
```

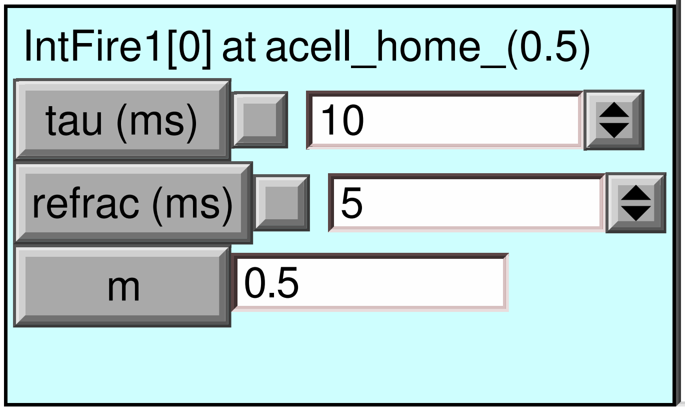




```
NEURON {
  ARTIFICIAL_CELL IntFire
  RANGE tau, m
}

...declarations...

INITIAL { m = 0   t0 = t }

NET_RECEIVE (w) {
  m = m*exp(-(t - t0)/tau)
  t0 = t
  m = m + w
  if (m > 1) {
    net_event(t)
    m = 0
  }
}
```



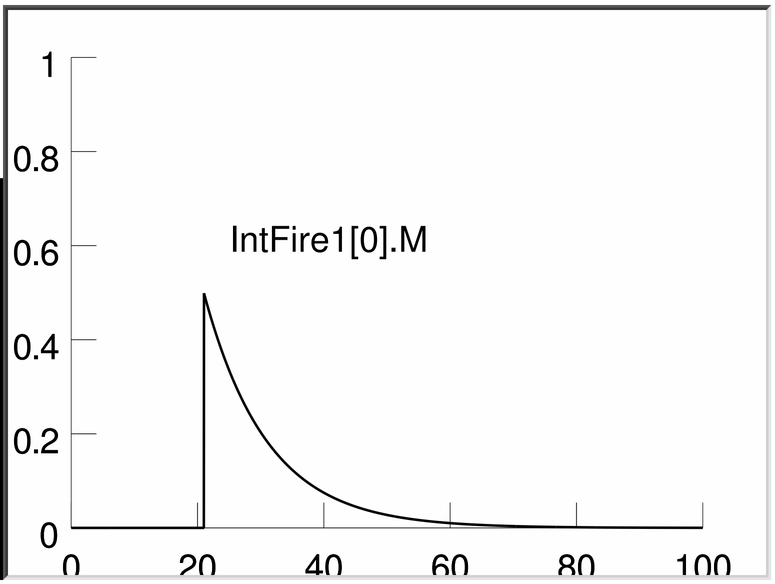
```
: dm/dt = (minf - m)/tau
: input event adds w to m
: when m = 1, or event makes m >= 1, cell fires
: minf is calculated so that the natural
:    interval between spikes is invl

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))
}
```

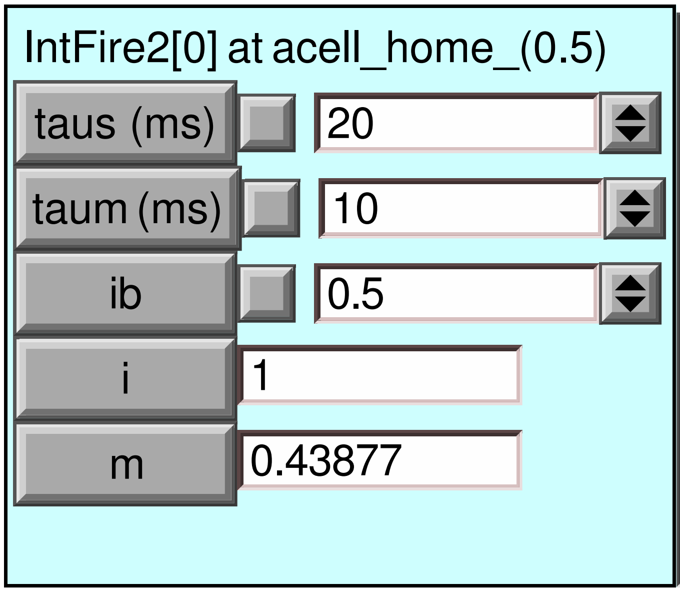

IntFire1[0] at acell_home_(0.5)

tau (ms) 10

refrac (ms) 5

m 0.5




IntFire2[0] at acell_home_(0.5)

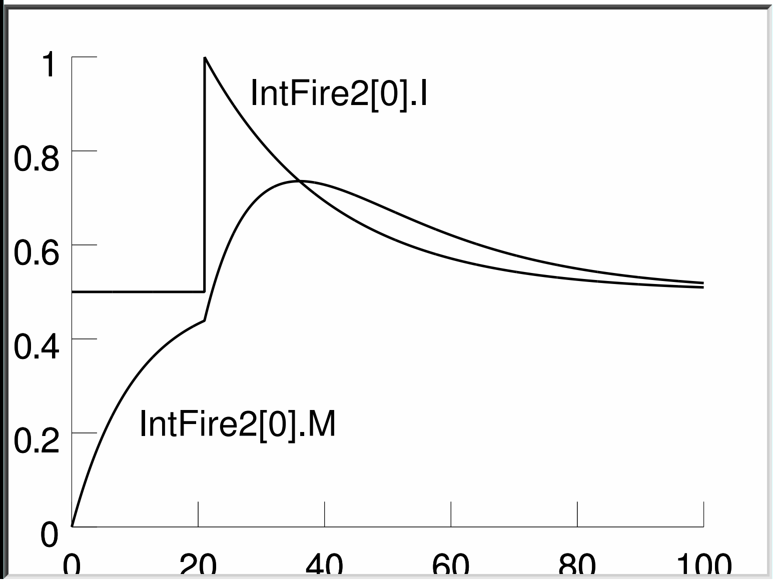
taus (ms) 20

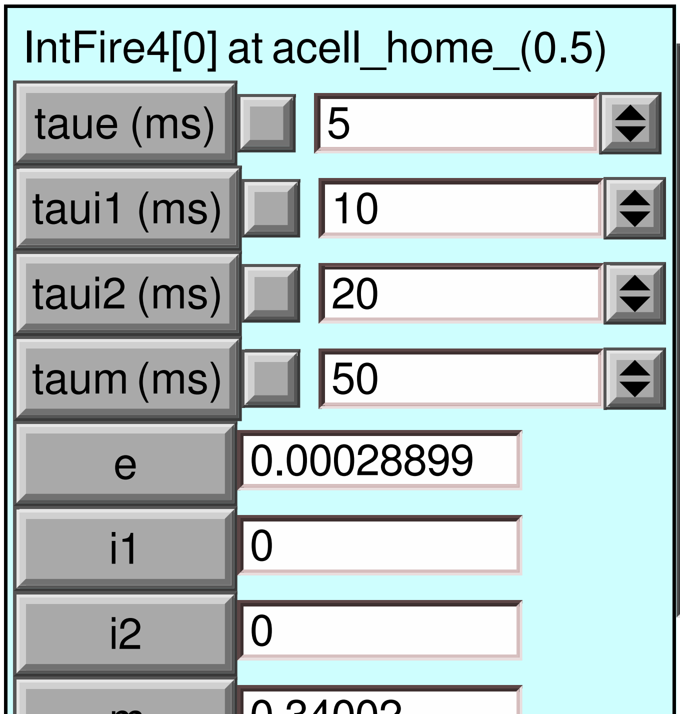
taum (ms) 10

ib 0.5

i 1

m 0.43877




IntFire4[0] at acell_home_(0.5)

taue (ms) 5

taui1 (ms) 10

taui2 (ms) 20

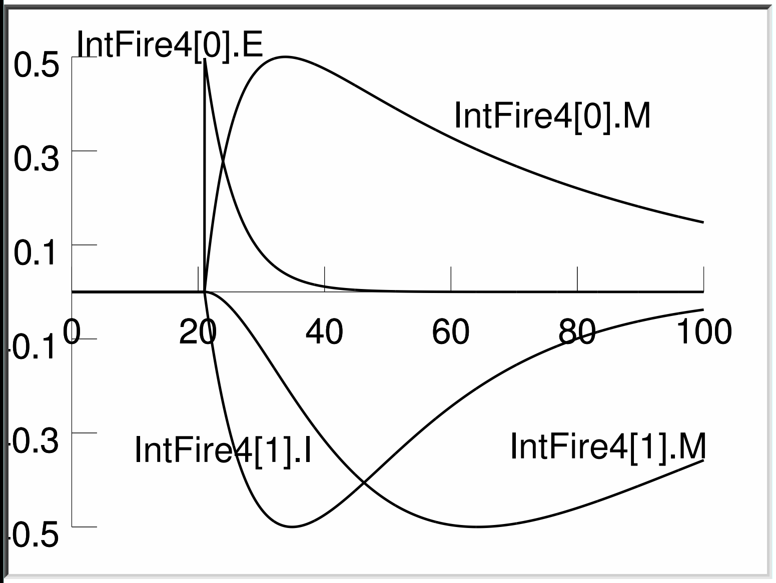
taum (ms) 50

e 0.00028899

i1 0

i2 0

m 0.34002

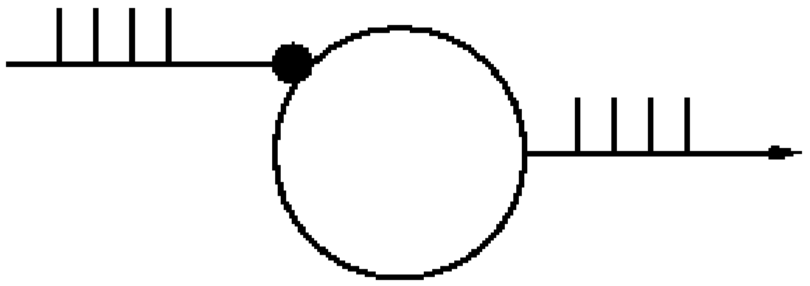


NetCon and NET_RECEIVE

```
NetCon(source, target, threshold, delay, weight)

   Event delivery with axonal delay

   Watch the source for threshold crossing in positive
   direction.
      &soma.v(.5)
      PresynapticObject.x
      Or PresynapticObject has a NET_RECEIVE block and calls
        net_event(t1) (discrete event simulation)

      All NetCon objects with same source use same threshold
      detector.

      If threshold occurs at time, t1, insert
      NetCon objects with that source into delivery queue
      for delivery at time t2 = t1 + NetCon.delay

   Balanced binary tree queue implementation.
   No loss of events. Works for delay=0.

   Event delivered to target NET_RECEIVE block at time t2.
      Same synaptic target equations used by many NetCon s.

      All declared NET_RECEIVE arguments are call by
      reference with separate storage in each NetCon.
         Calculations for different streams can be done once
         per event instead of once per dt.

      A target can send itself an event with
      net_send(delay, flag) and move it to a new time with
      net_move(t)

   With variable step methods, and 1 or more events at time
   t, fadvance() returns at time t before the events are
   delivered and at time t after the events are delivered.
```

Networks with Artificial Cells

Model



Artificial Integrate and Fire cell stimulated by a burst
of action potentials.

Simulation

The strategy is to

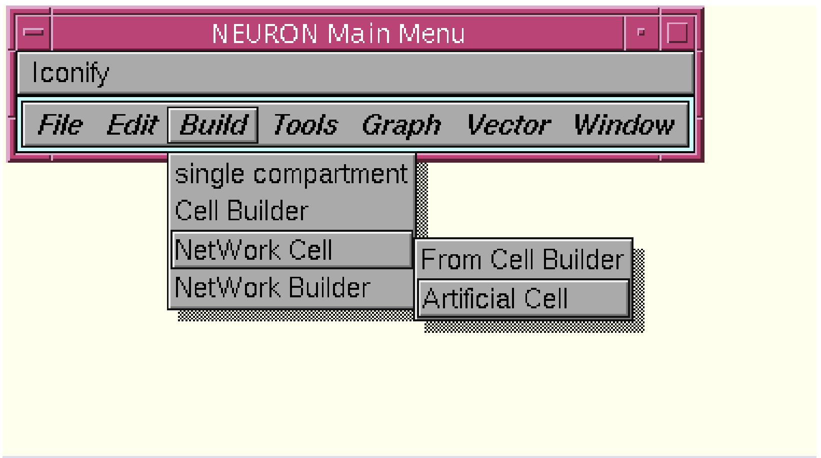
1) define the types of cells (and stimulators),

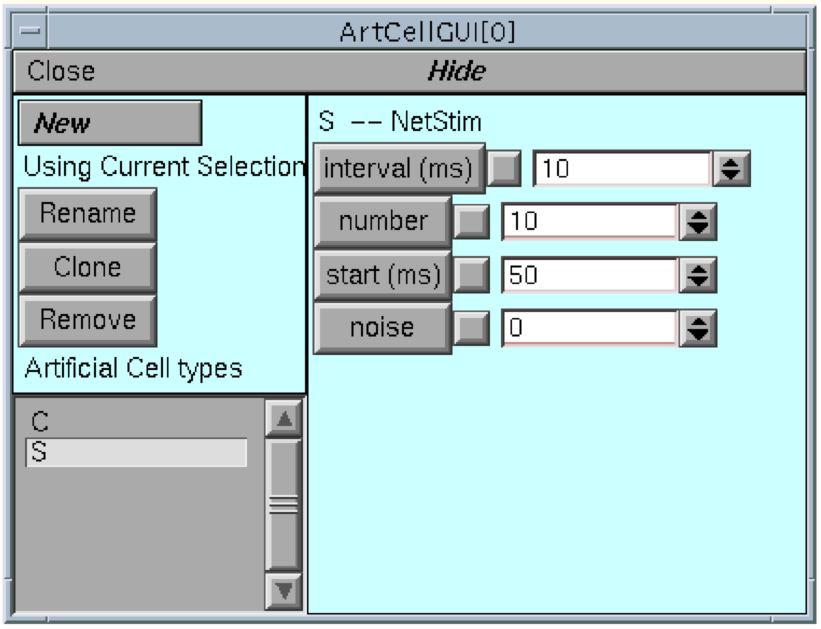
2) create each cell in the network,

3) connect the cells together,

4) specify parameters such as delays and connection weights,

5) run a simulation and plot the input and output spike trains.





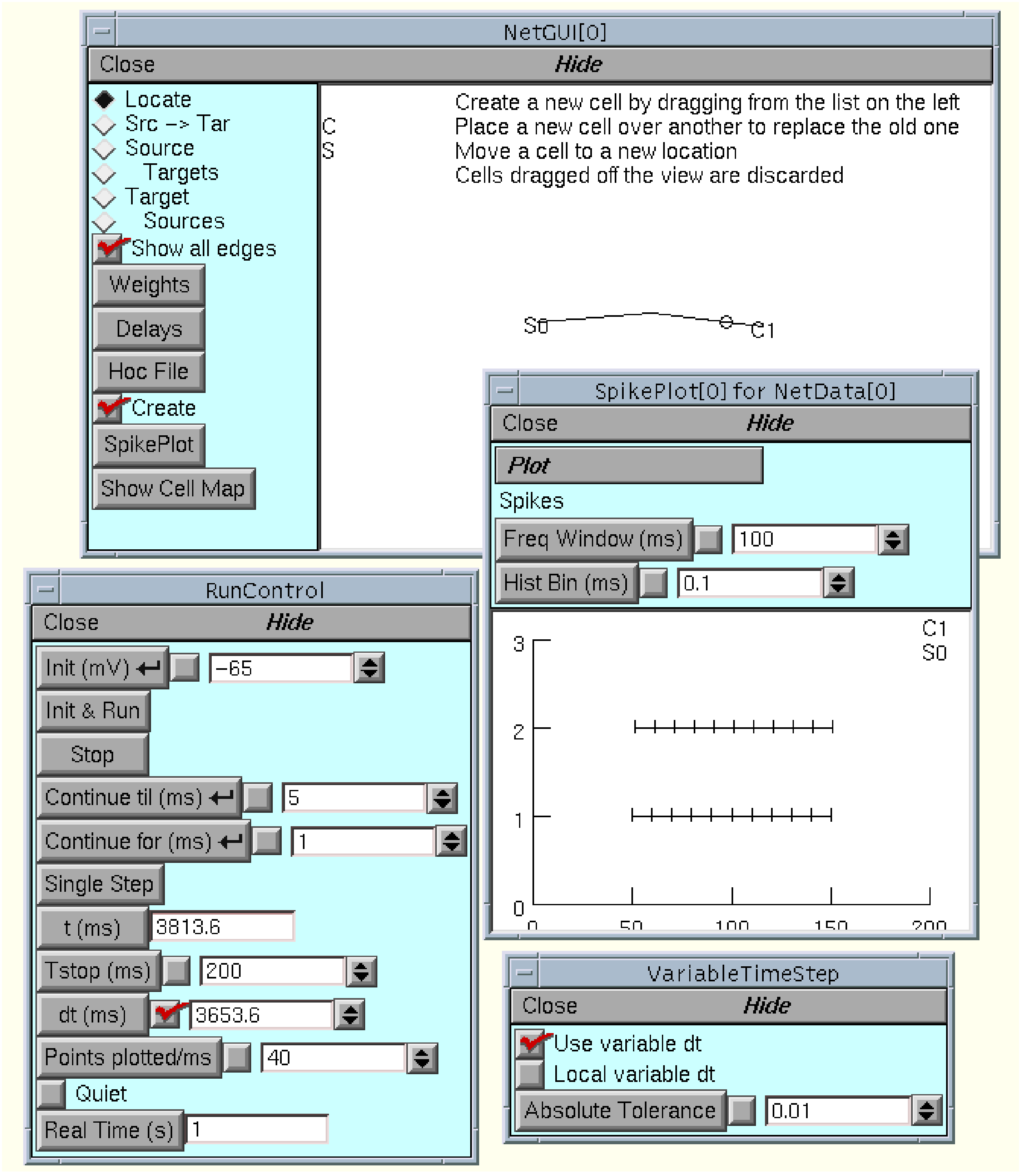
S -- NetStim

interval (ms) 10

number 10

start (ms) 50

noise 0



Create a new cell by dragging from the list on the left
Place a new cell over another to replace the old one
Move a cell to a new location
Cells dragged off the view are discarded

✓ Show all edges
✓ Create

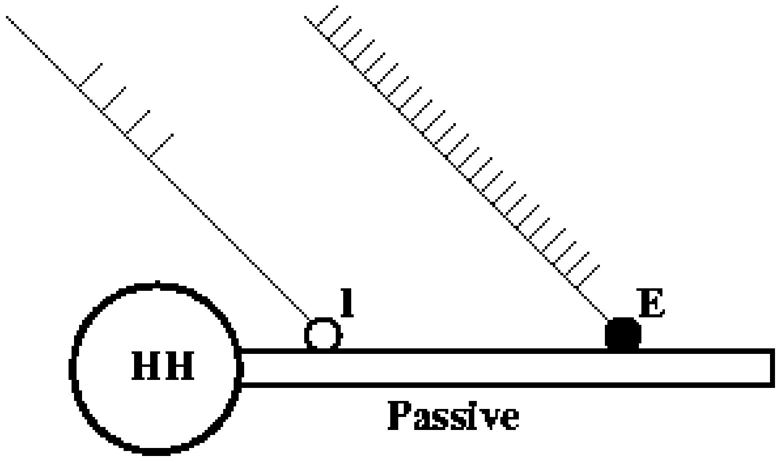
Freq Window (ms) 100
Hist Bin (ms) 0.1

Init (mV) -65
Continue til (ms) 5
Continue for (ms) 1
t (ms) 3813.6
Tstop (ms) 200
dt (ms) ✓ 3653.6
Points plotted/ms 40
Quiet
Real Time (s) 1

✓ Use variable dt
Local variable dt
Absolute Tolerance 0.01

Network ready cells from the CellBuilder

Model



Ball-Stick model cell with distal excitation
and proximal inhibition.

Simulation

The strategy is to

1) Use a CellBuilder window to create a cell type
with specific morphology and membrane properties.

2) Define synapse types with a SynTypeGUI.

3) Define a network ready cell type with a NetReadyCellGUI.

4) Use a NetGUI to construct the network

5) Run a simulation and plot the input and output spike trains.