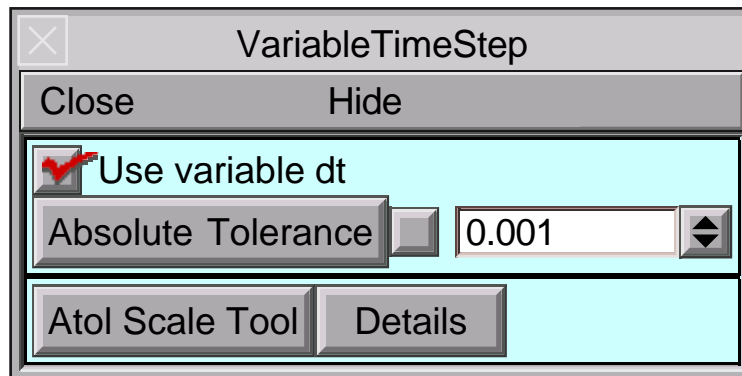


`cvar_active(1)`



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_cadifmp | 1e-06 | 3e-06 | 0 |
| pump_cadifmp | 1e-15 | 1e-13 | 0 |
| pumpca_cadifmp | 1e-15 | 3.6e-15 | 0 |
| 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-NFixedStep
☒ Ccode
☐ Daspk
☐ Local step

DAE and daspk require sparse solver, ccode requires tree solver

☒ Mx=b tree solver
☐ Mx=b sparse solver

☐ 2nd order threshold (for variable step)

RunControl

Init (mV)

Init & Run

Stop

Continue til (ms)

Continue for (ms)

Single Step

t (ms)

Tstop (ms)

dt (ms)

Points plotted/ms

Quiet

Real Time (s)

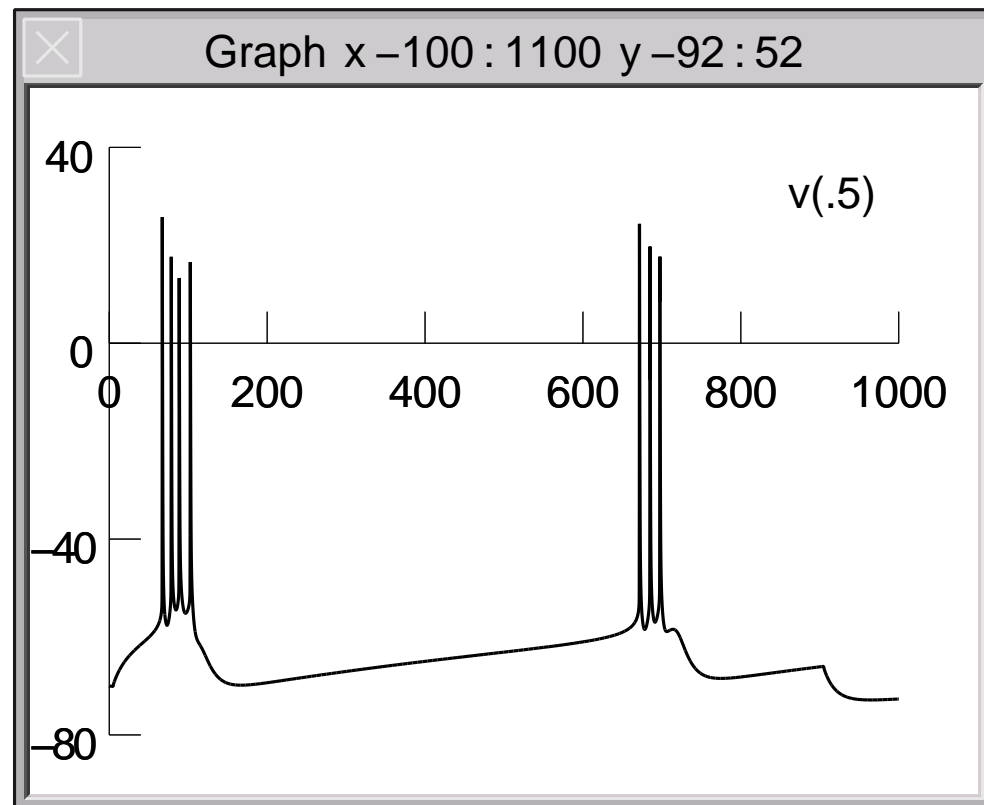


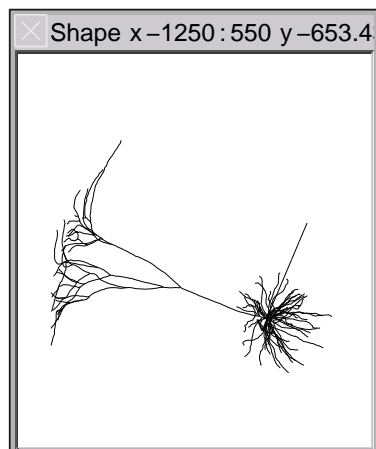
Figure 1

a. L3 Aspiny

b. L4 Stellate

c. L3 Pyramid

d. L5 Pyramid



Insert/Remo

soma

☒ pas

☐ hh

☒ ca

☒ cad

☒ kca

☒ km

☒ kv

☒ na

RunControl

Init (mV) ← ▢ -70 ▢ ▴ ▾

Init & Run

Stop

Continue til (ms) ← ▢ 5 ▢ ▴ ▾

Continue for (ms) ← ▢ 1 ▢ ▴ ▾

Single Step

t (ms) ▢ 1000

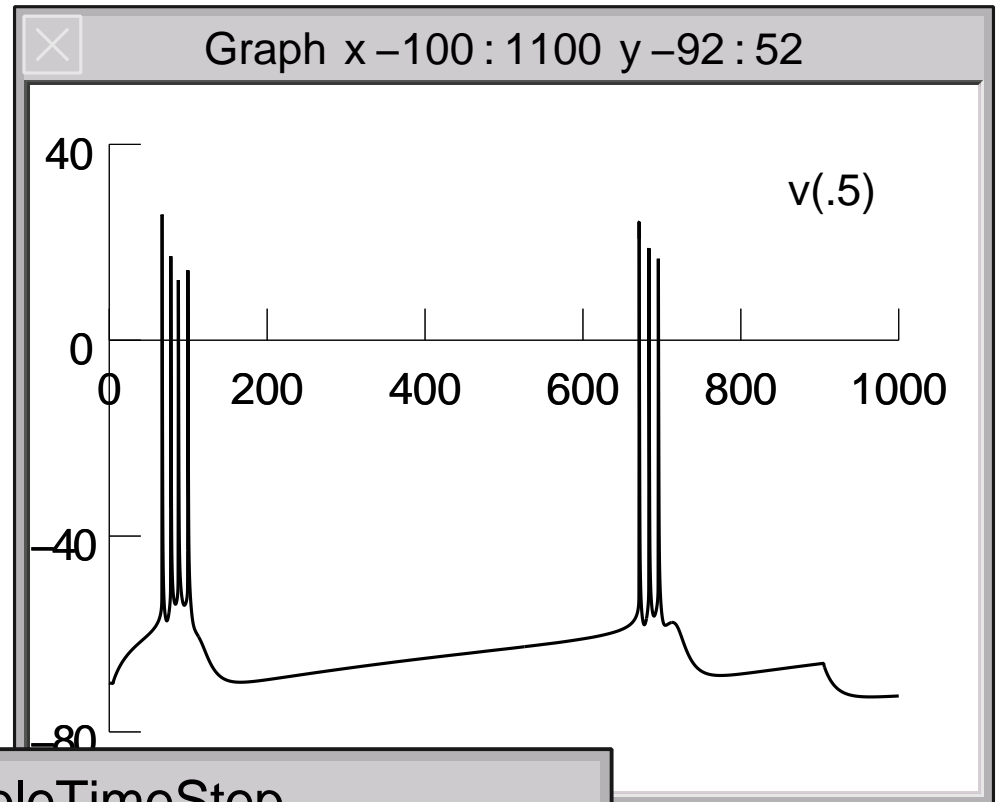
Tstop (ms) ▢ 1000 ▢ ▴ ▾

dt (ms) ☒ ▢ 6.131 ▢ ▴ ▾

Points plotted/ms ▢ 40 ▢ ▴ ▾

☐ Quiet

Real Time (s) ▢ 36

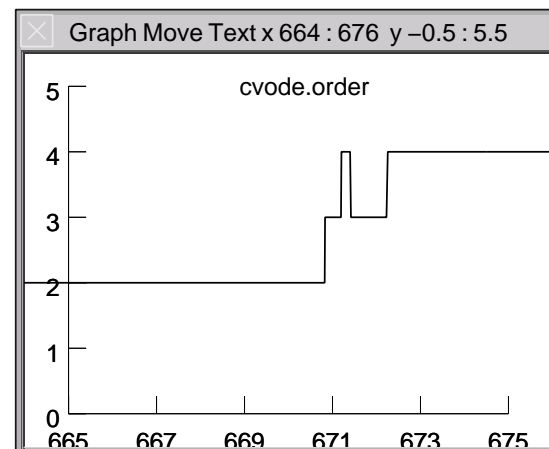
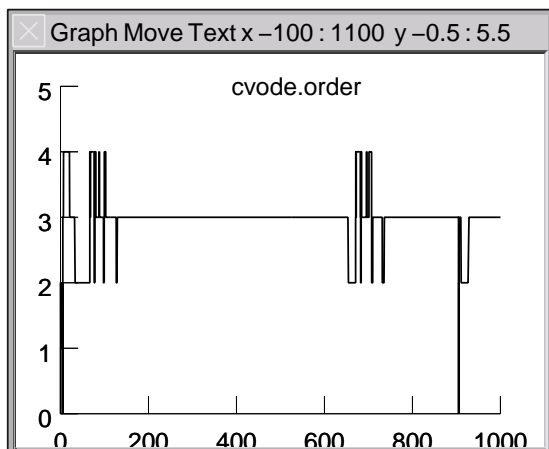
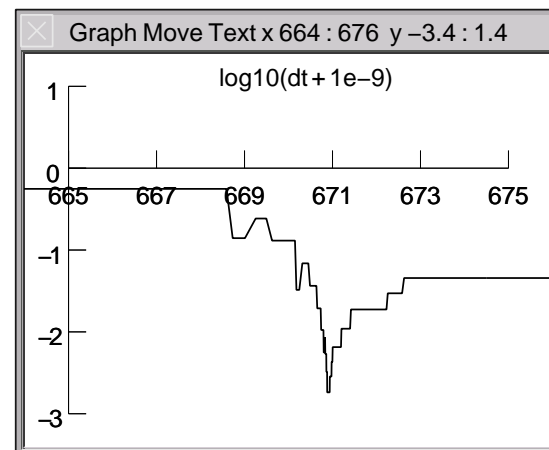
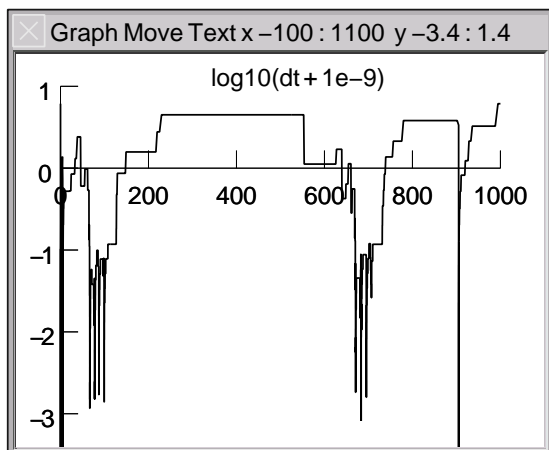
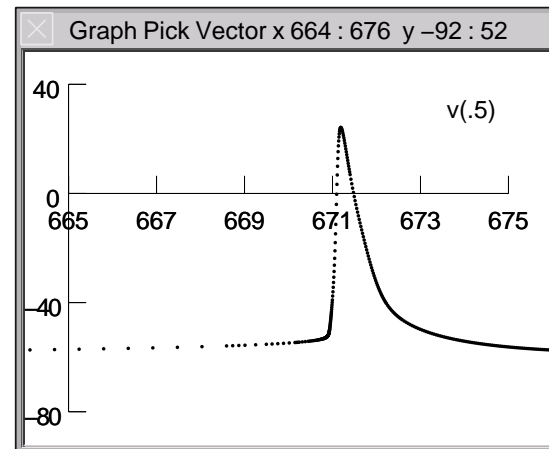
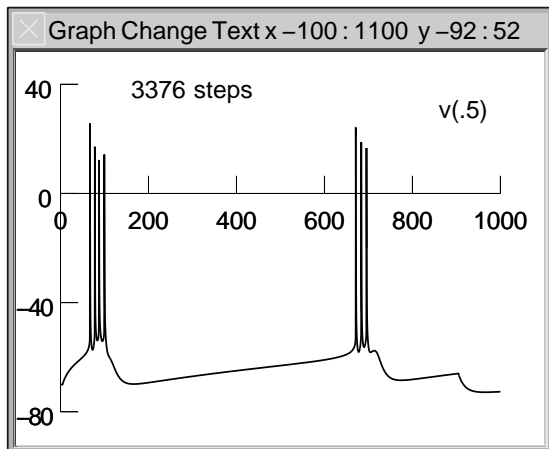


VariableTimeStep

☒ Use variable dt

Absolute Tolerance ▢ 0.001 ▢ ▴ ▾

Atol Scale Tool ▢ Details ▢





Spinal Motor Neuron: McIntyre et al 2002

Simulation of peripheral nervous system (PNS) myelinated axon. This model is described in detail in: McIntyre CC, Richard Grill WM.(2002)

Reference: McIntyre CC, Richardson AG, Grill WM (2002) Modeling the excitability of Mammalian nerve fibers: influence of afterpotentials on the recovery cycle. *J Neurophysiol* **87**:995-1006 [[PubMed](#)]

Citations [Citation Browser](#)

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

Model Type: [Axon](#);

Cell Type(s): [Spinal motor neuron](#);

Channel(s): [I_{Na,p}](#); [I_{Na,t}](#); [I_K](#); [I_{Sodium}](#); [I_{Potassium}](#);

Receptor(s):

Transmitter(s):

Simulation Environment: [Neuron](#);

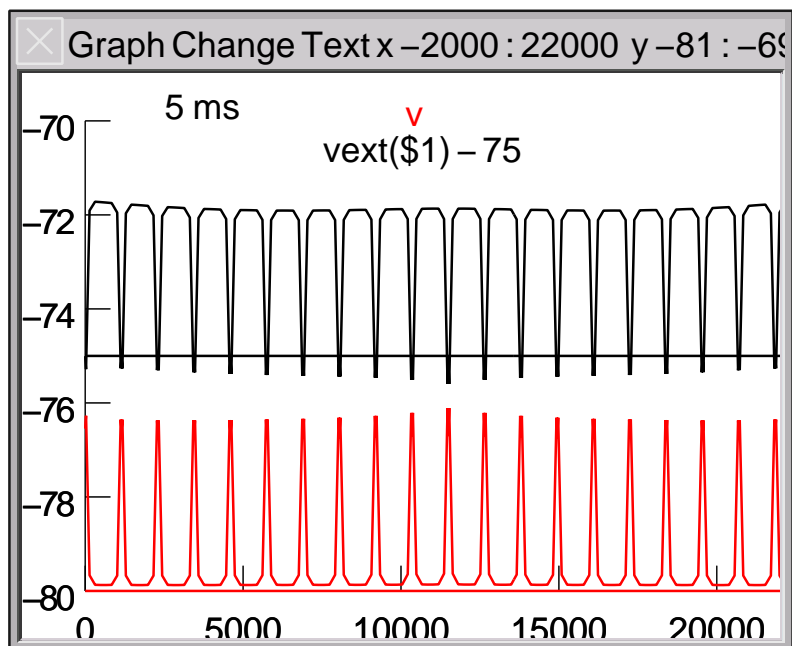
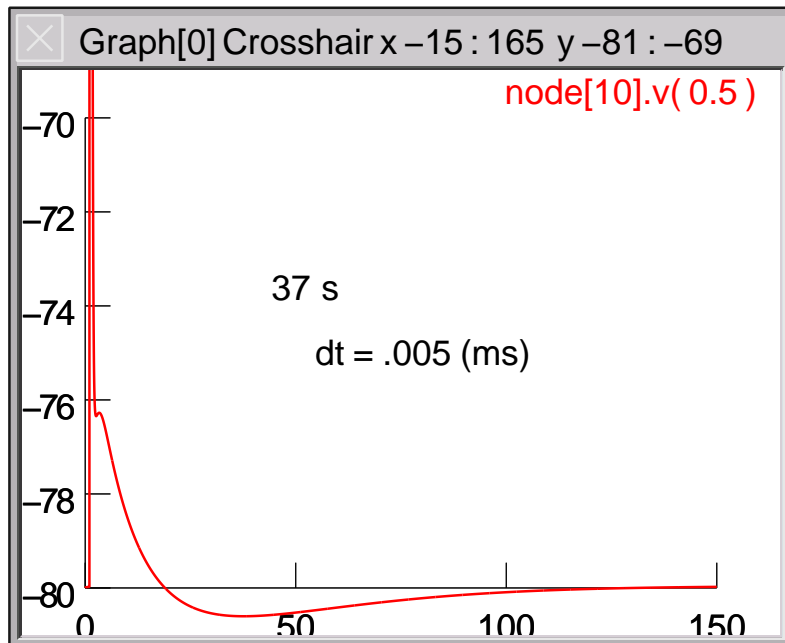
Model Concept(s): [Axonal Action Potentials](#); [Action Potentials](#);

Implementer(s): [MacIntyre, CC](#);

Search NeuronDB for information about: [Spinal motor neuron](#); [I_{Na,p}](#); [I_{Na,t}](#); [I_K](#); [I_{Sodium}](#); [I_{Potassium}](#);

| Model files | Download zip file | Auto-launch | Help downloading and running models |
|---|---|-------------|---|
| <div> \ <div> MRGaxon README AXNODE.mod MRGaxon.hoc mosinit.hoc MRGaxon.ses </div> </div> | <p>SIMULATION OF PNS MYELINATED AXON</p> <p>This model is described in detail in:</p> <p>McIntyre CC, Richardson AG, and Grill WM. Modeling the excitability mammalian nerve fibers: influence of afterpotentials on the recover cycle. <i>Journal of Neurophysiology</i> 87:995-1006, 2002.</p> <p>The model is set up to reproduce part of Fig 2A from this paper.</p> <p>This model can not be used with NEURON v5.1 as errors in the extracellular mechanism of v5.1 exist related to xc. The original stimulations were run on v4.3.1. NEURON v5.2 has corrected the limitations in v5.1 and can be used to run this model.</p> <p>Please contact mcintyre@bme.jhu.edu if you have any questions about</p> | | |

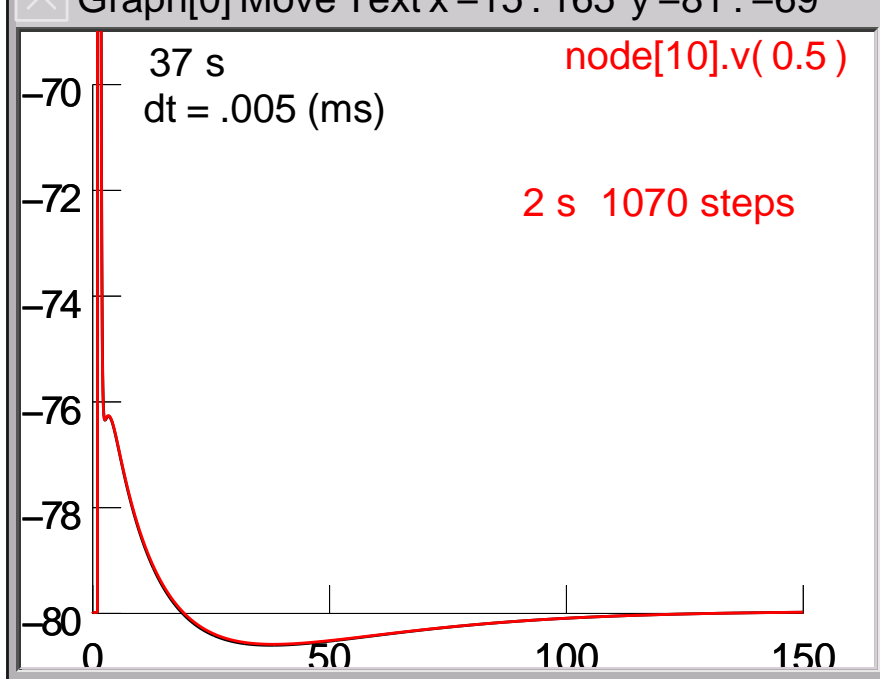
Total site hits since January 1, 2002: **346093**



ModelView[0]

221 sections; 221 segments

- * 1 real cells
 - * root node[0]
 - 221 sections; 221 segments
 - * 1 distinct values of nseg
 - * 5 inserted mechanisms
 - * Ra
 - * capacitance
 - * pas
 - * extracellular
 - * 2 xrxial[0]
 - 160 xrxial[0] = 80684
 - 61 xrxial[0] = 337397
 - xrxial[1] = 1e+09
 - * 2 xg[0]
 - 200 xg[0] = 4.16667e-06
 - 21 xg[0] = 1e+10
 - xg[1] = 1e+09
 - * 2 xc[0]
 - 21 xc[0] = 0
 - 200 xc[0] = 0.000416667
 - xc[1] = 0
 - e_extracellular = 0
 - * axnode
 - * 6 subsets with constant parameters
 - * 1 IClamp

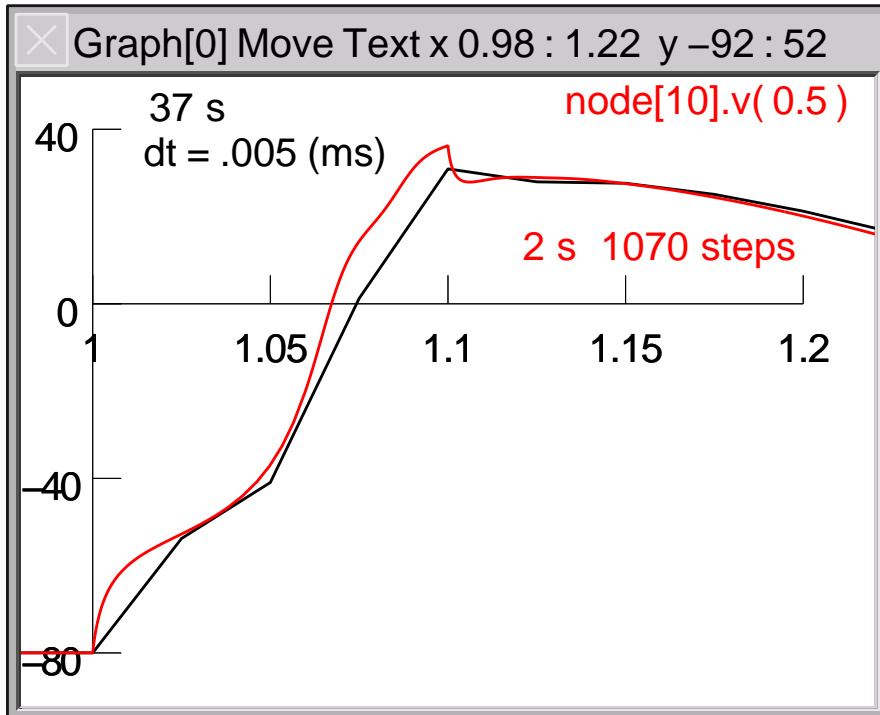


VariableTimeStep

☒ Use variable dt

Absolute Tolerance

Atol Scale Tool Details



Numerical Method Selection

Refresh

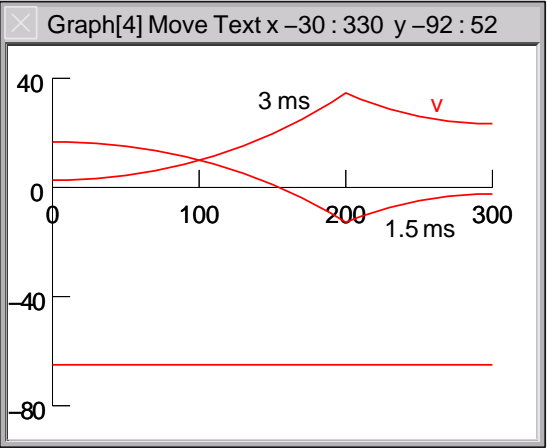
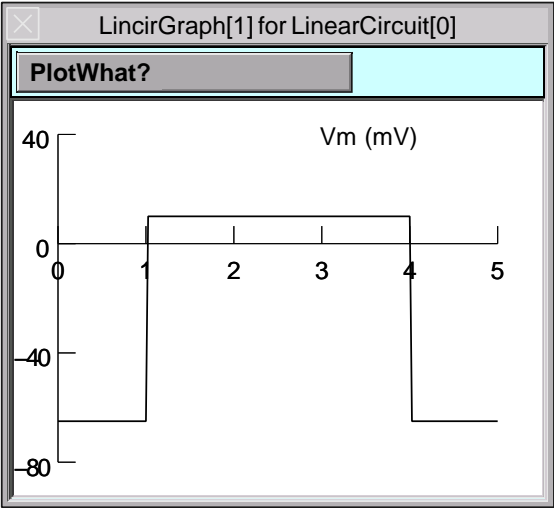
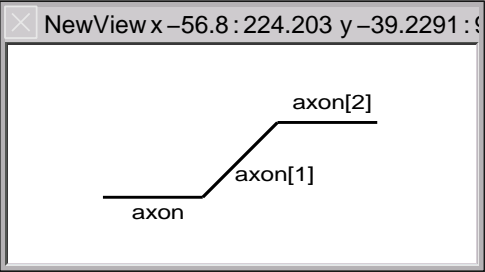
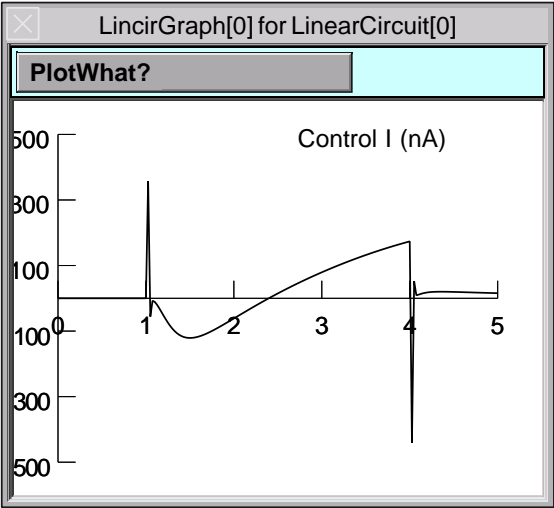
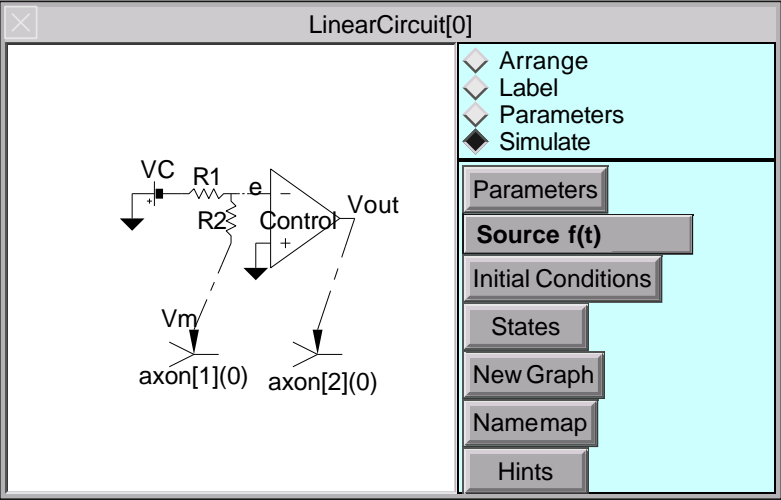
current model type: ODE <*DAE*>
ODE model allows any method
DAE model allows implicit fixed step or daspk

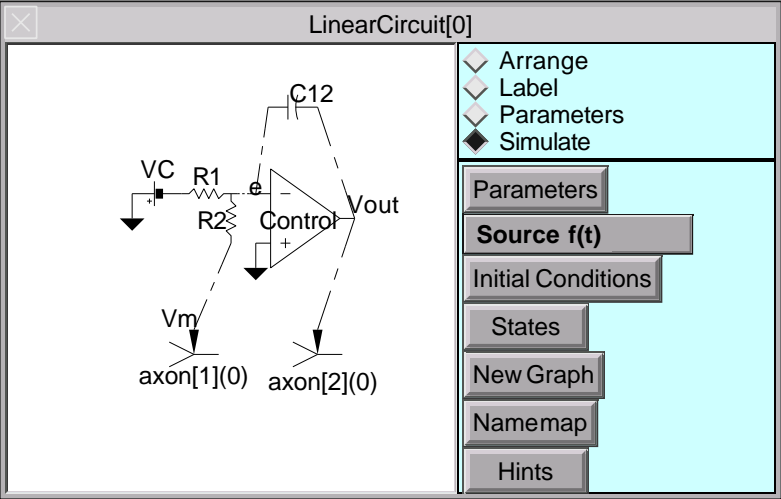
☐ Implicit Fixed Step
☐ C-NFixedStep
☐ Cvode
☒ Daspk
☐ Local step

DAE and daspk require sparse solver, cvode requires tree solver

☐ Mx=b tree solver
☒ Mx=b sparse solver

☐ 2nd order threshold (for variable step)





VariableTimeStep

☒ Use variable dt

Absolute Tolerance

Atol Scale Tool Details

Values for LinearCircuit[0]

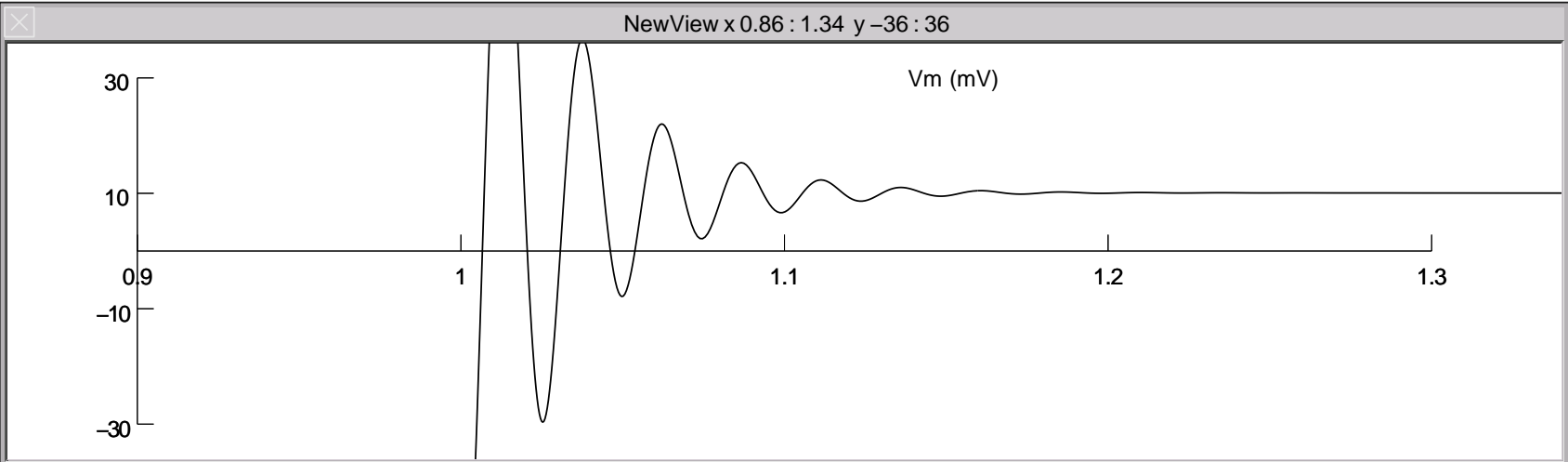
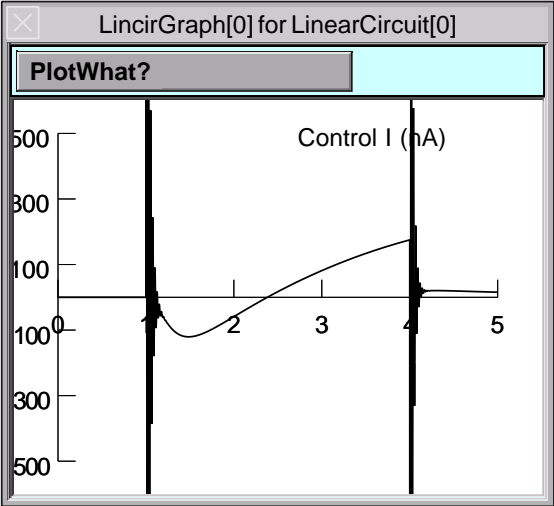
Control Gain

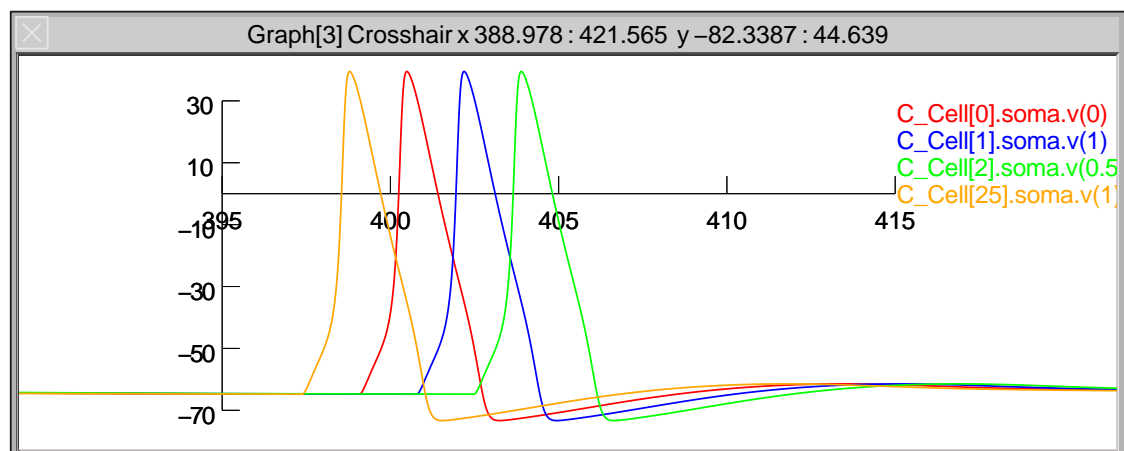
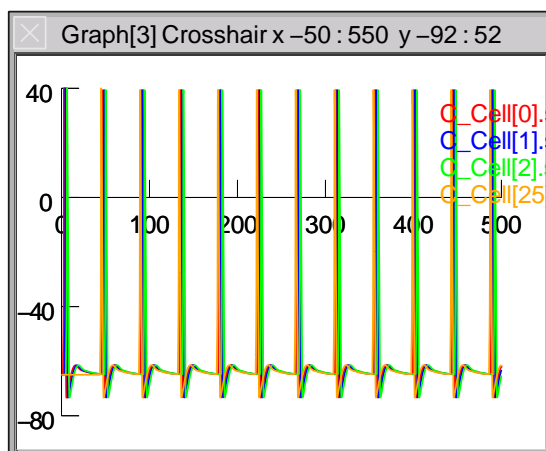
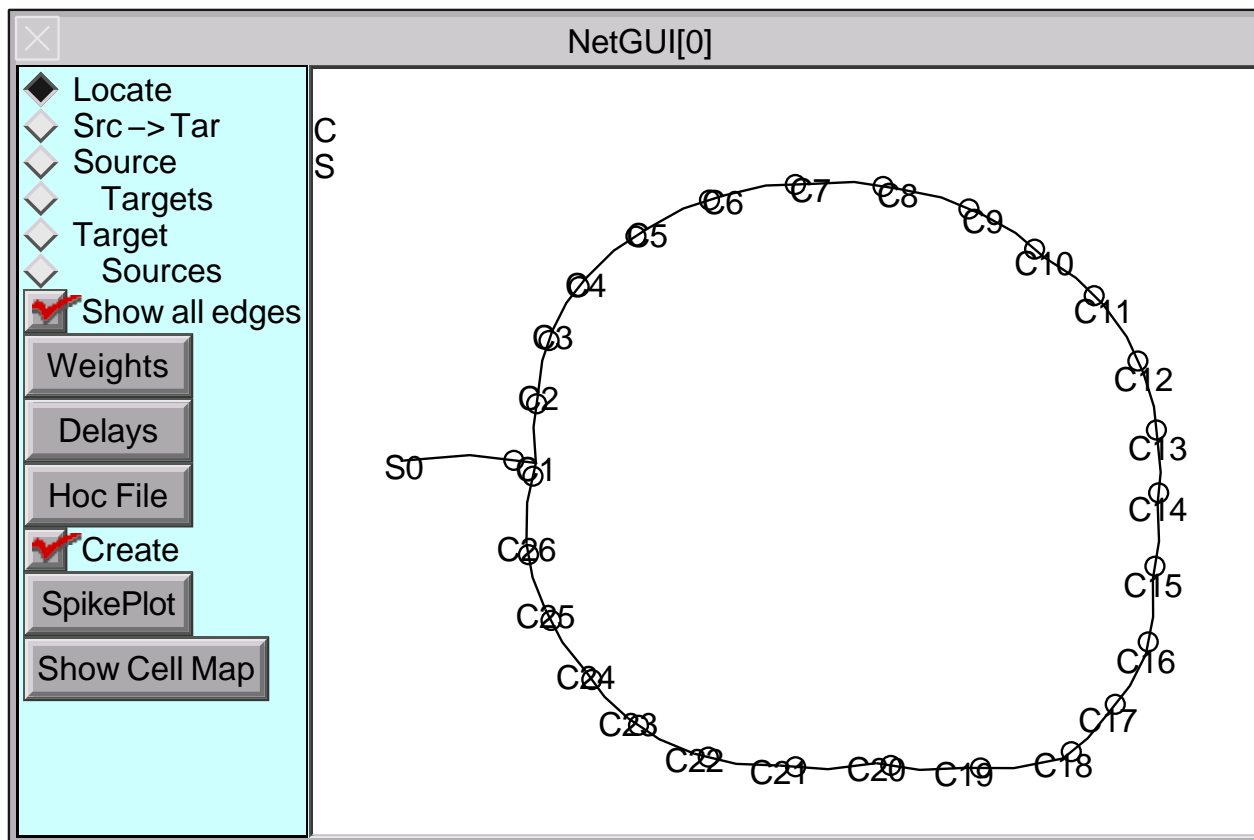
Control Tau (ms)

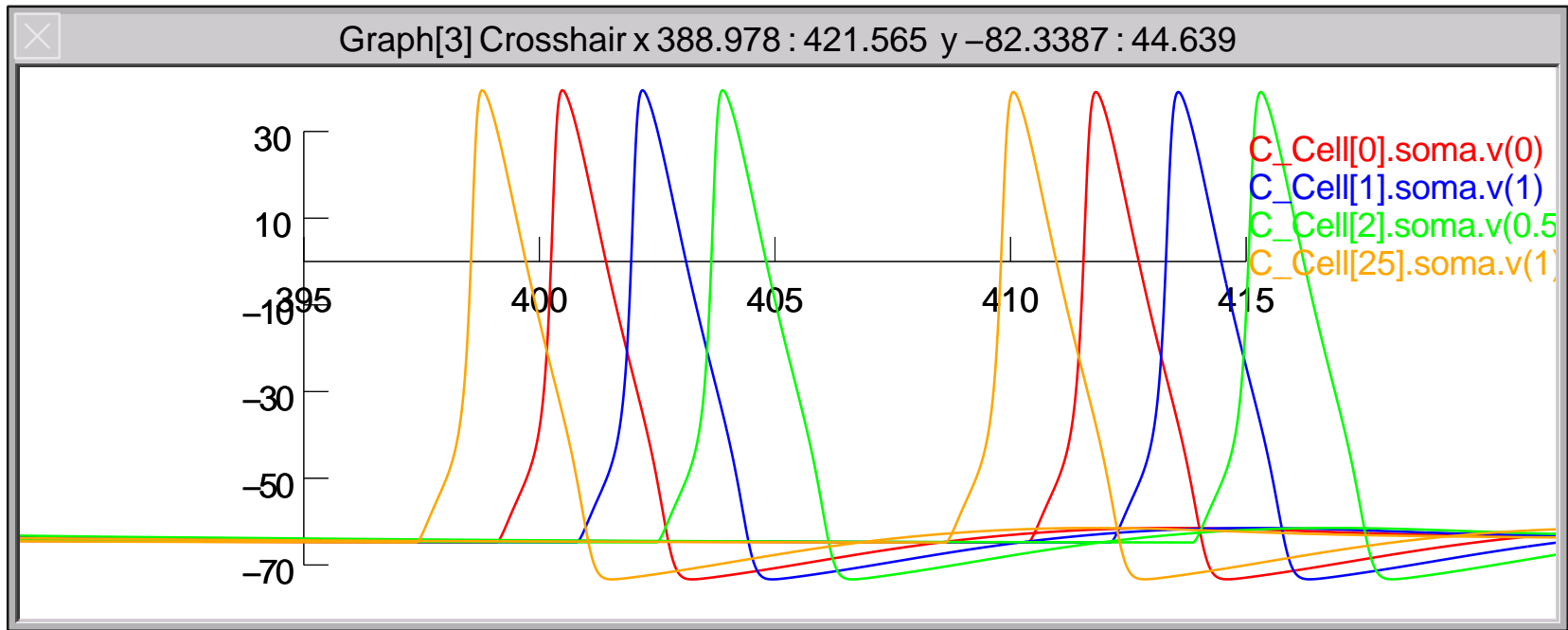
R1 (Mohm)

R2 (Mohm)

C12 (nF) ☒







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-NFixedStep

☐ Cvode

☐ Daspk

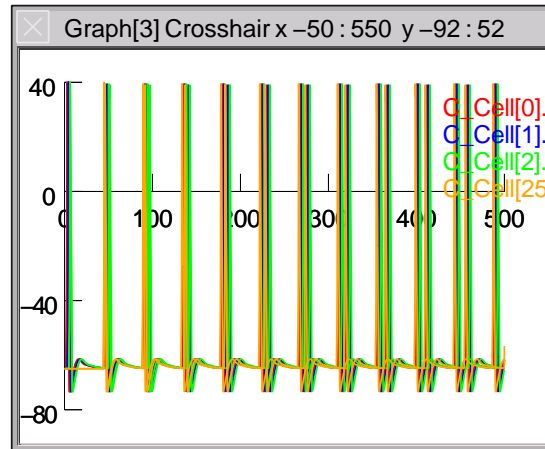
☐ Local step

DAE and daspk require sparse solver, cvode

☒ Mx=b tree solver

☐ Mx=b sparse solver

☐ 2nd order threshold (for variable step)



RunControl

Init (mV)

Init & Run

Stop

Continue til (ms)

Continue for (ms)

Single Step

t (ms)

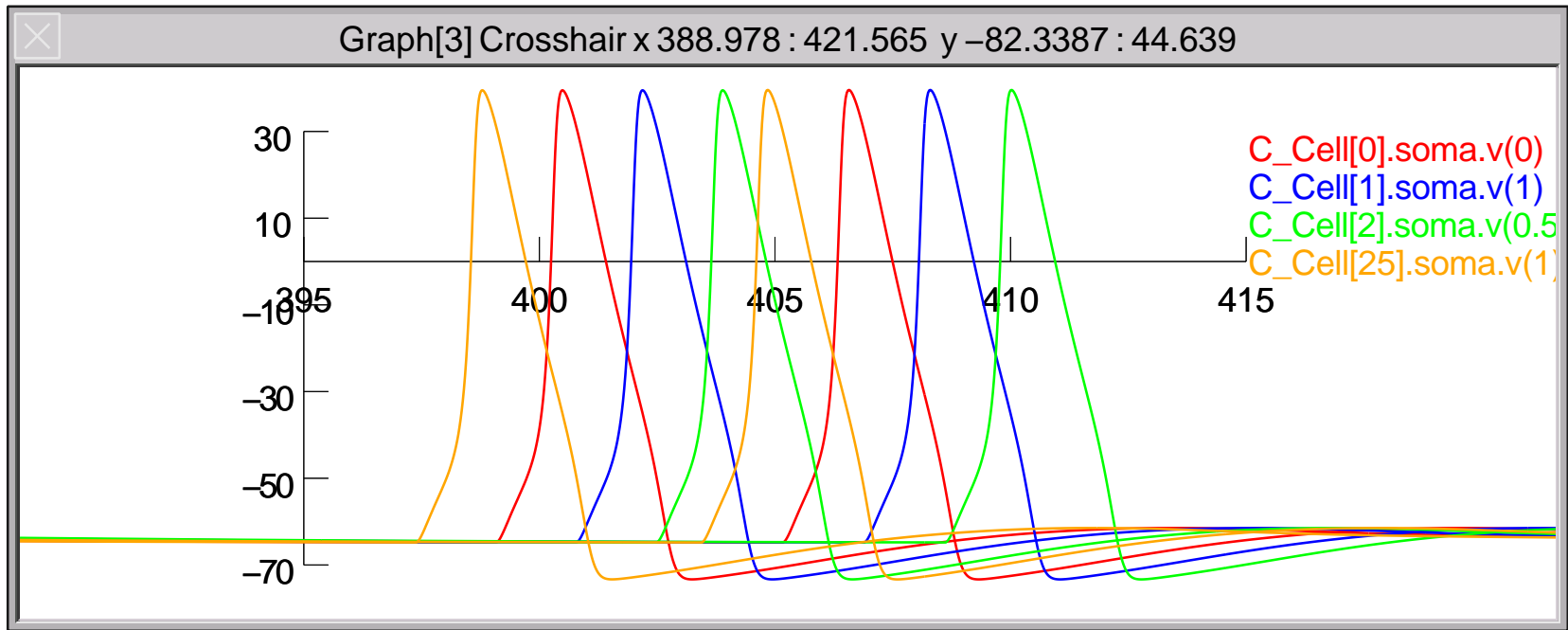
Tstop (ms)

dt (ms)

Points plotted/ms

Quiet ☐

Real Time (s)



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-NFixedStep

☐ Cvode

☐ Daspk

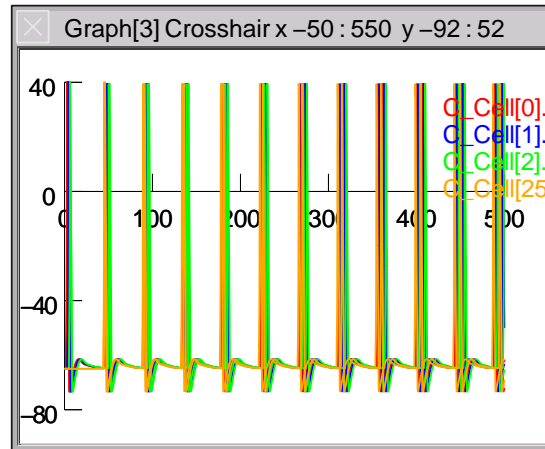
☐ Local step

DAE and daspk require sparse solver, cvode

☒ Mx=b tree solver

☐ Mx=b sparse solver

☐ 2nd order threshold (for variable step)



RunControl

Init (mV)

Init & Run

Stop

Continue til (ms)

Continue for (ms)

SingleStep

t (ms)

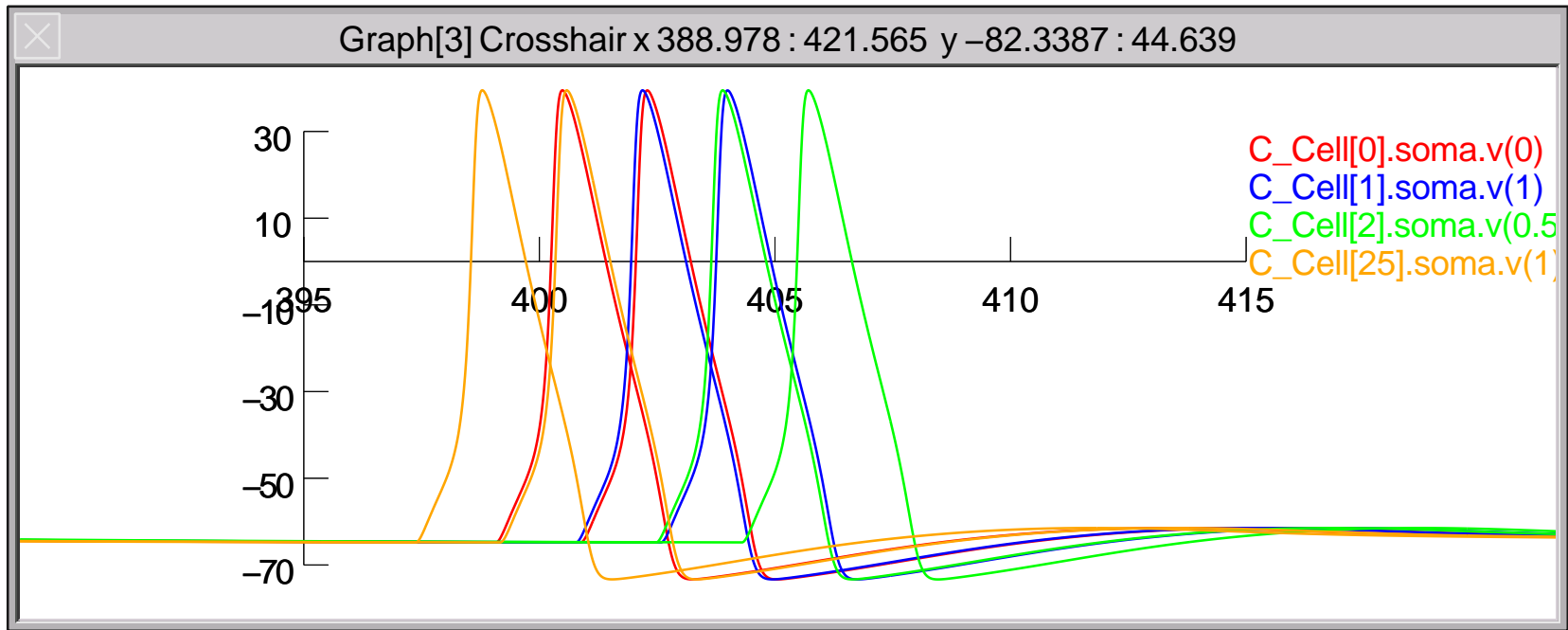
Tstop (ms)

dt (ms)

Points plotted/ms

Quiet ☐

Real Time (s)



Numerical Method Selection

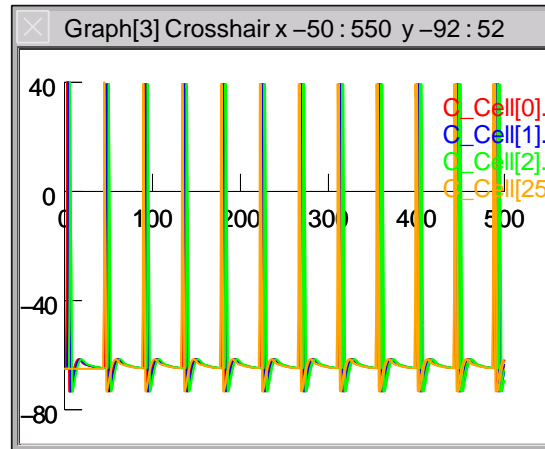
Refresh

current model type: <*ODE*> DAE
ODE model allows any method
DAE model allows implicit fixed step or daspk

☐ Implicit Fixed Step
☐ C-NFixedStep
☒ Cvode
☐ Daspk
☐ Local step

DAE and daspk require sparse solver, cvode
☒ Mx=b tree solver
☐ Mx=b sparse solver

☐ 2nd order threshold (for variable step)



VariableTimeStep

☒ Use variable dt

Absolute Tolerance

Atol Scale Tool Details

Absolute Tolerance Scale Factors

Analysis Run Rescale Original

*10 /10 Hints

| | | | |
|-----------|--------|---------|---------|
| v | 10 | 73 | 0.013 |
| m_hh | 1 | 0.99 | 0.00028 |
| h_hh | 0.1 | 0.6 | 0.00014 |
| n_hh | 0.1 | 0.77 | 5.5e-05 |
| Exp2Syn.A | 0.0001 | 0.00053 | 1.5e-06 |
| Exp2Syn.B | 0.0001 | 0.00054 | 1.8e-10 |

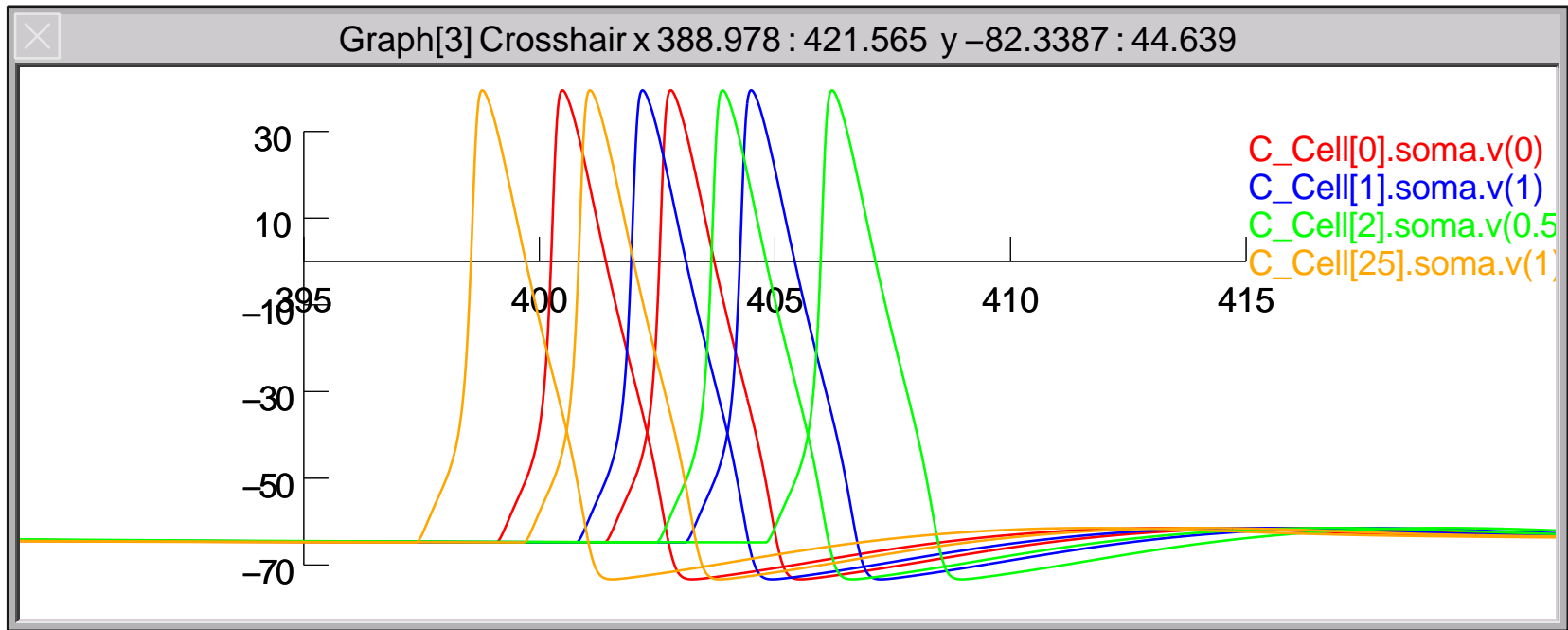
Tstop (ms)

dt (ms) ☒ 0.021762

Points plotted/ms

☐ Quiet

Real Time (s)



Numerical Method Selection

Refresh

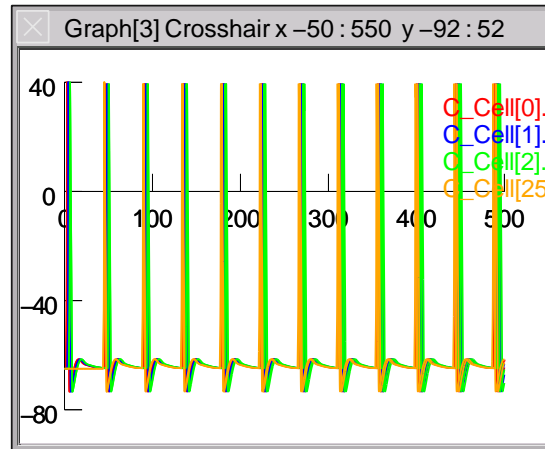
current model type: <*ODE*> DAE
ODE model allows any method
DAE model allows implicit fixed step or daspk

☐ Implicit Fixed Step
☐ C-NFixedStep
☒ Cvode
☐ Daspk
☐ Local step

DAE and daspk require sparse solver, cvode

☒ Mx=b tree solver
☐ Mx=b sparse solver

☐ 2nd order threshold (for variable step)



VariableTimeStep

☒ Use variable dt

Absolute Tolerance ☒ 0.0001

Atol Scale Tool Details

Absolute Tolerance Scale Factors

Analysis Run Rescale Original

*10 /10 Hints

| | | | |
|-----------|--------|---------|---------|
| v | 10 | 73 | 0.013 |
| m_hh | 1 | 0.99 | 0.00028 |
| h_hh | 0.1 | 0.6 | 0.00014 |
| n_hh | 0.1 | 0.77 | 5.5e-05 |
| Exp2Syn.A | 0.0001 | 0.00053 | 1.5e-06 |
| Exp2Syn.B | 0.0001 | 0.00054 | 1.8e-10 |

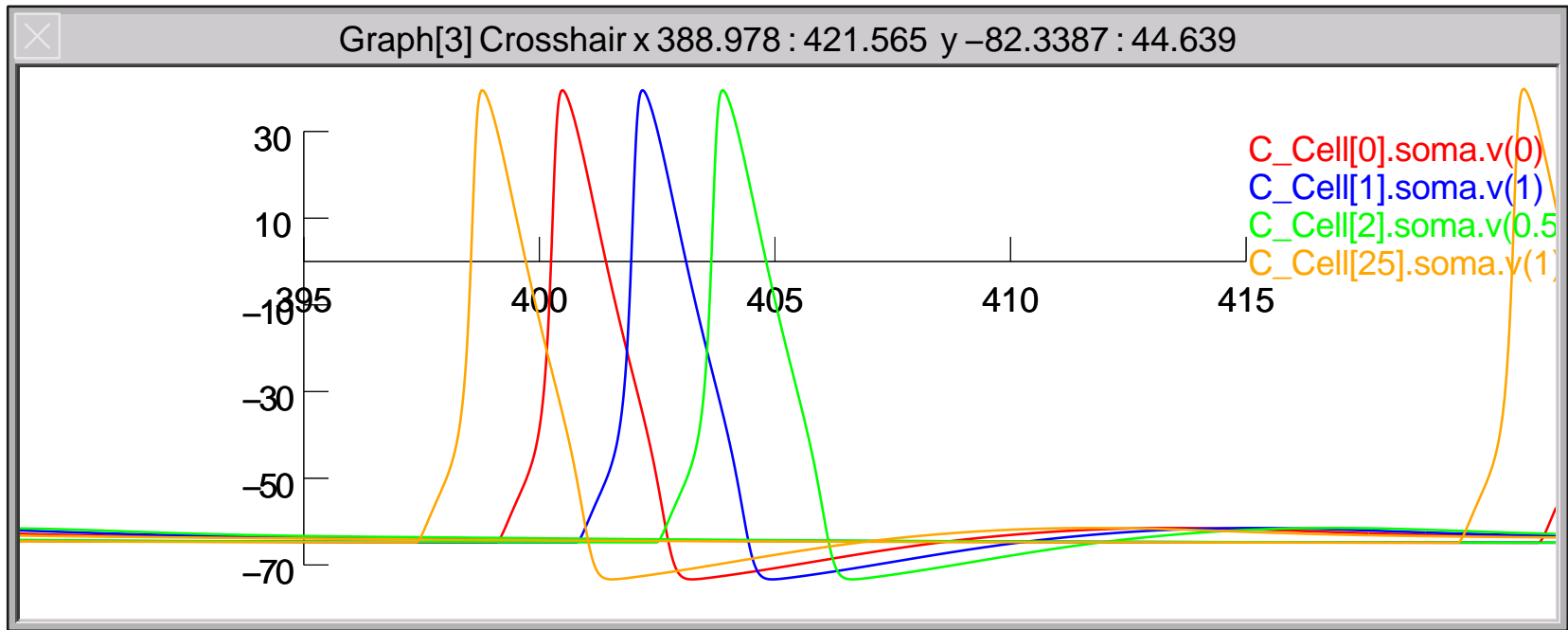
Tstop (ms) 500

dt (ms) ☒ 0.033263

Points plotted/ms 40

Quiet

Real Time (s) 5



Numerical Method Selection

Refresh

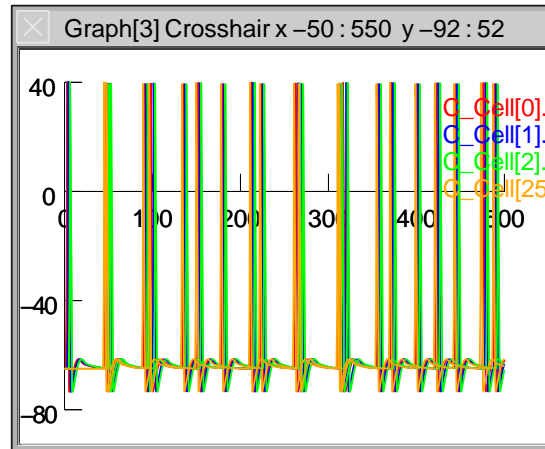
current model type: <*ODE*> DAE
ODE model allows any method
DAE model allows implicit fixed step or daspk

☐ Implicit Fixed Step
☐ C-NFixedStep
☐ Cvode
☐ Daspk
☒ Local step

DAE and daspk require sparse solver, cvode

☒ Mx=b tree solver
☐ Mx=b sparse solver

☐ 2nd order threshold (for variable step)



VariableTimeStep

☒ Use variable dt

Absolute Tolerance

Atol Scale Tool Details

Absolute Tolerance Scale Factors

Analysis Run Rescale Original

*10 /10 Hints

| | | | |
|-----------|--------|---------|---------|
| v | 10 | 73 | 0.013 |
| m_hh | 1 | 0.99 | 0.00028 |
| h_hh | 0.1 | 0.6 | 0.00014 |
| n_hh | 0.1 | 0.77 | 5.5e-05 |
| Exp2Syn.A | 0.0001 | 0.00053 | 1.5e-06 |
| Exp2Syn.B | 0.0001 | 0.00054 | 1.8e-10 |

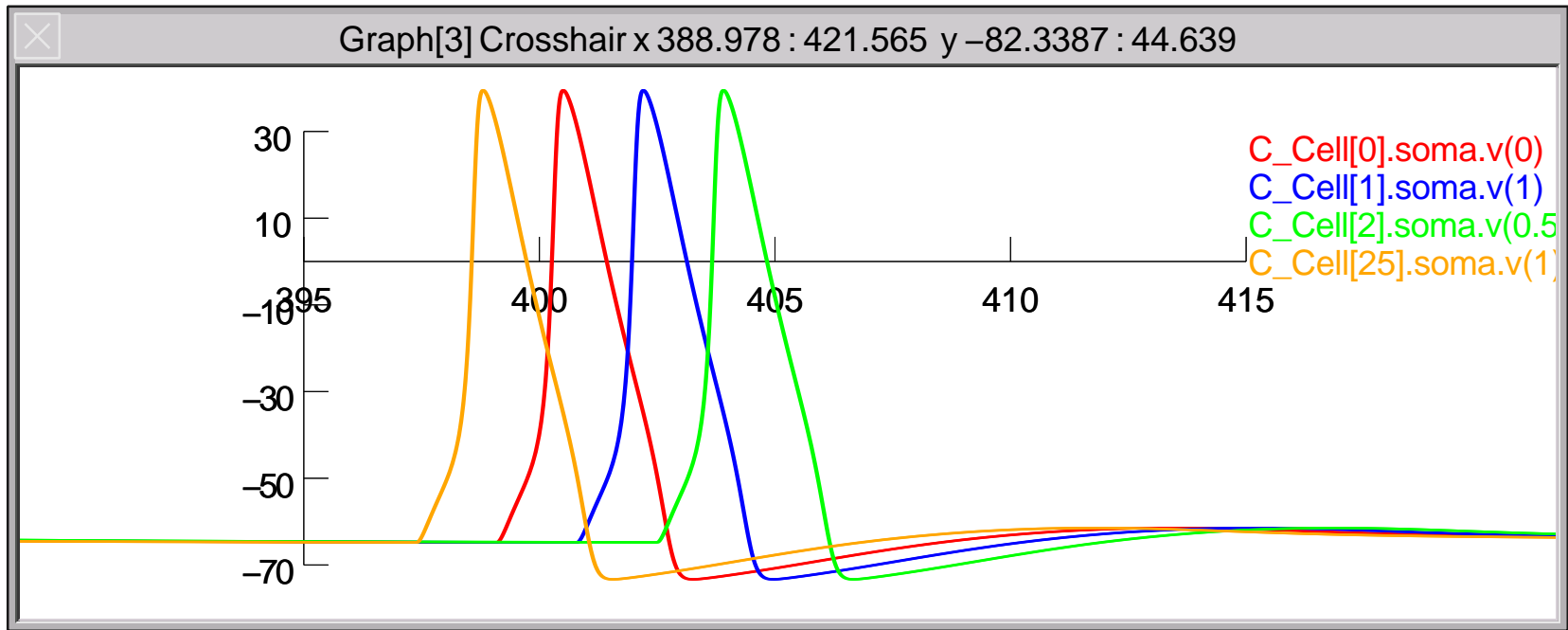
Tstop (ms)

dt (ms) ☒ 0.41618

Points plotted/ms

☐ Quiet

Real Time (s)



Numerical Method Selection

Refresh

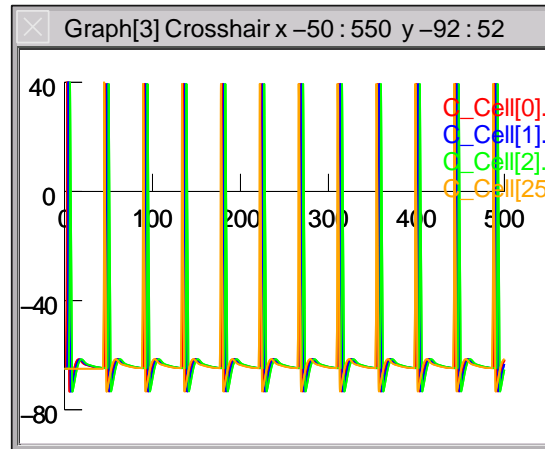
current model type: <*ODE*> DAE
ODE model allows any method
DAE model allows implicit fixed step or daspk

☐ Implicit Fixed Step
☐ C-NFixedStep
☐ Ccode
☐ Daspk
☒ Local step

DAE and daspk require sparse solver, ccode

☒ Mx=b tree solver
☐ Mx=b sparse solver

☒ 2nd order threshold (for variable step)



VariableTimeStep

☒ Use variable dt

Absolute Tolerance

Atol Scale Tool Details

Absolute Tolerance Scale Factors

Analysis Run Rescale Original

*10 /10 Hints

| | | | |
|-----------|--------|---------|---------|
| v | 10 | 73 | 0.013 |
| m_hh | 1 | 0.99 | 0.00028 |
| h_hh | 0.1 | 0.6 | 0.00014 |
| n_hh | 0.1 | 0.77 | 5.5e-05 |
| Exp2Syn.A | 0.0001 | 0.00053 | 1.5e-06 |
| Exp2Syn.B | 0.0001 | 0.00054 | 1.8e-10 |

Tstop (ms)

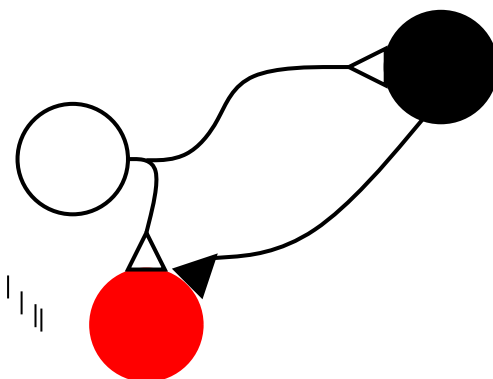
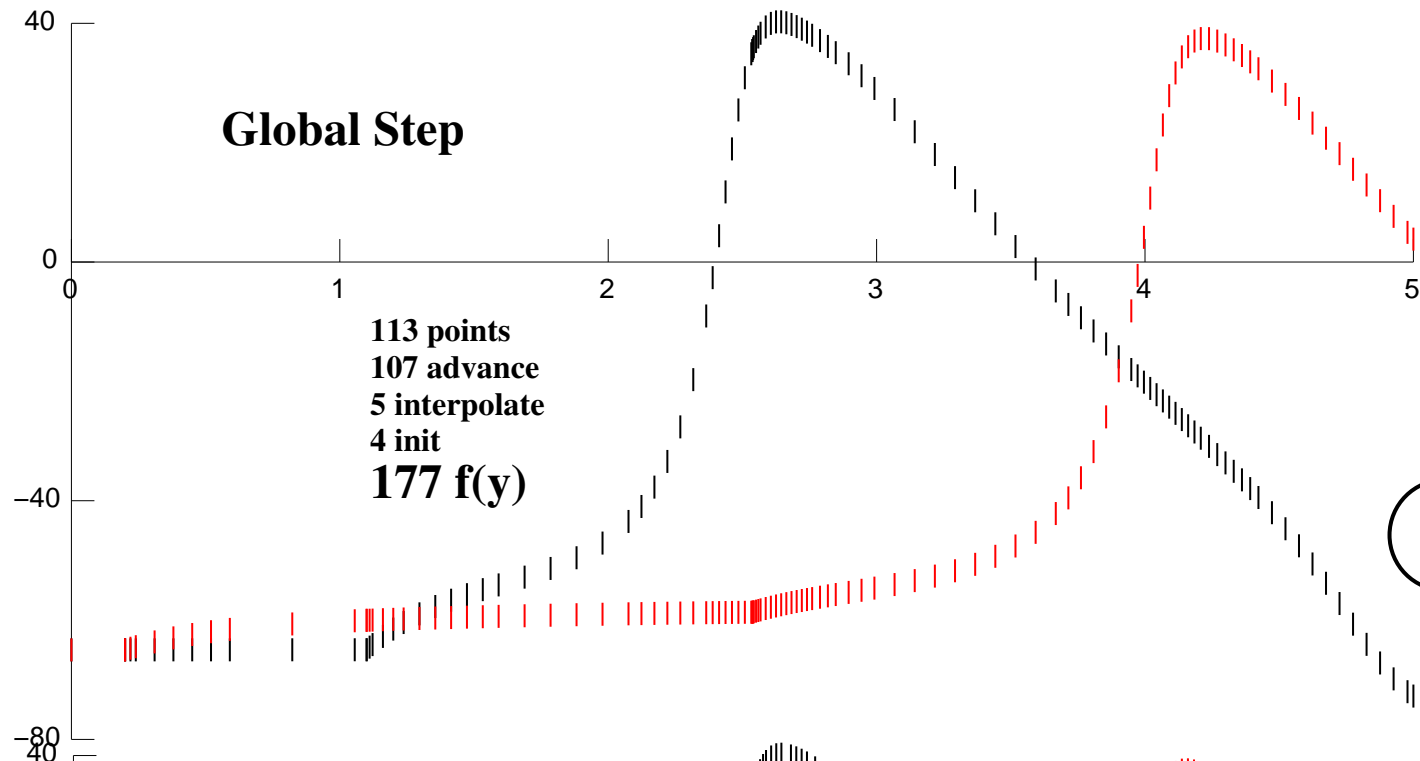
dt (ms) ☒ 0.42314

Points plotted/ms

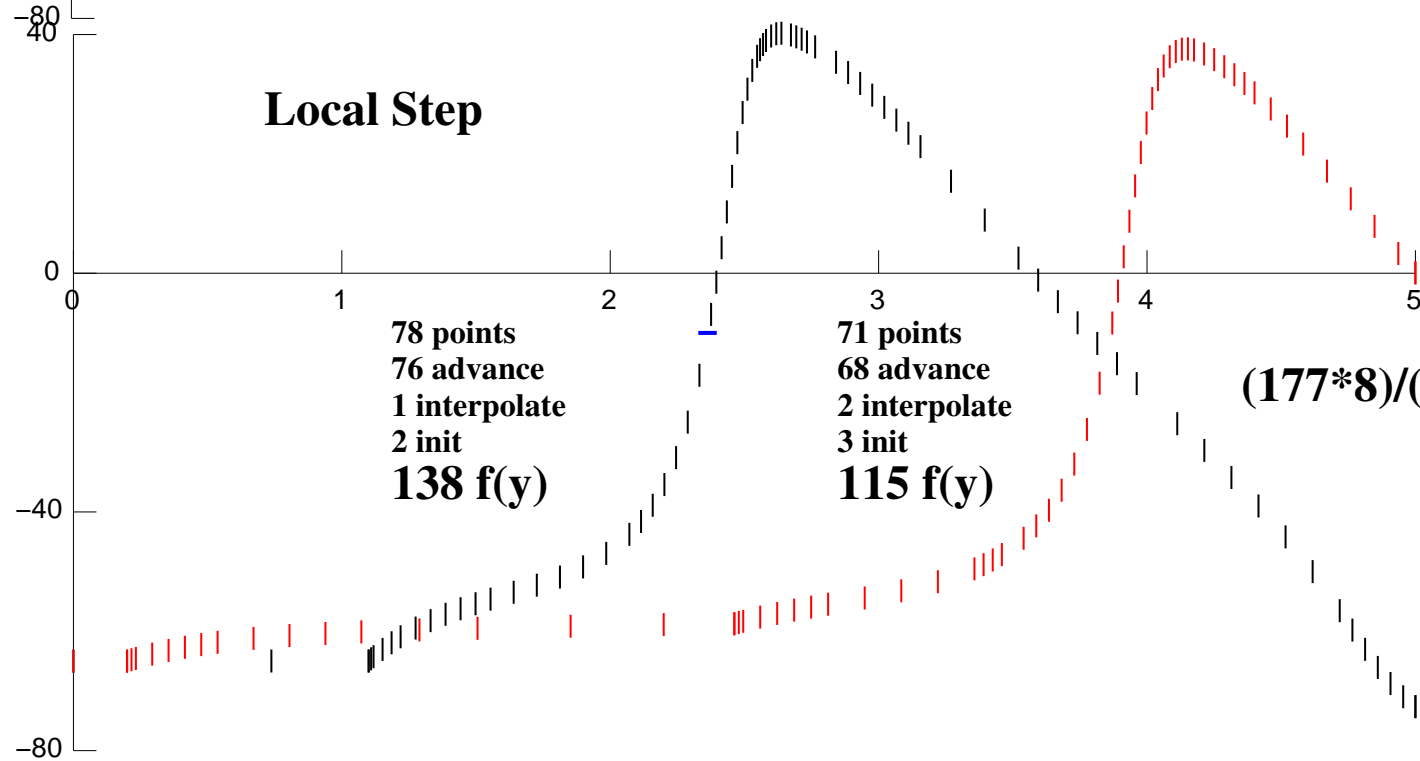
☐ Quiet

Real Time (s)

Global Step



Local Step



$$(177 \cdot 8) / (138 \cdot 4 + 115 \cdot 4) = 1.4$$

One integrator instance per cell

$$\forall i, j : \quad \mathbf{ta}_i \leq \mathbf{tb}_j$$

ta

t

tb



advance



interpolate



init

1



2



1

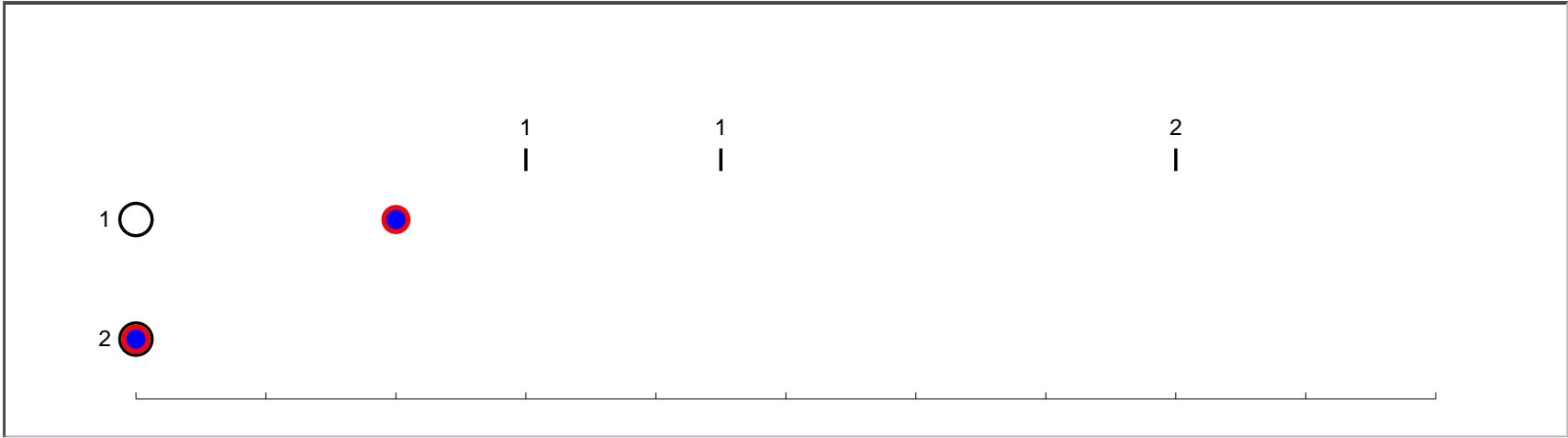


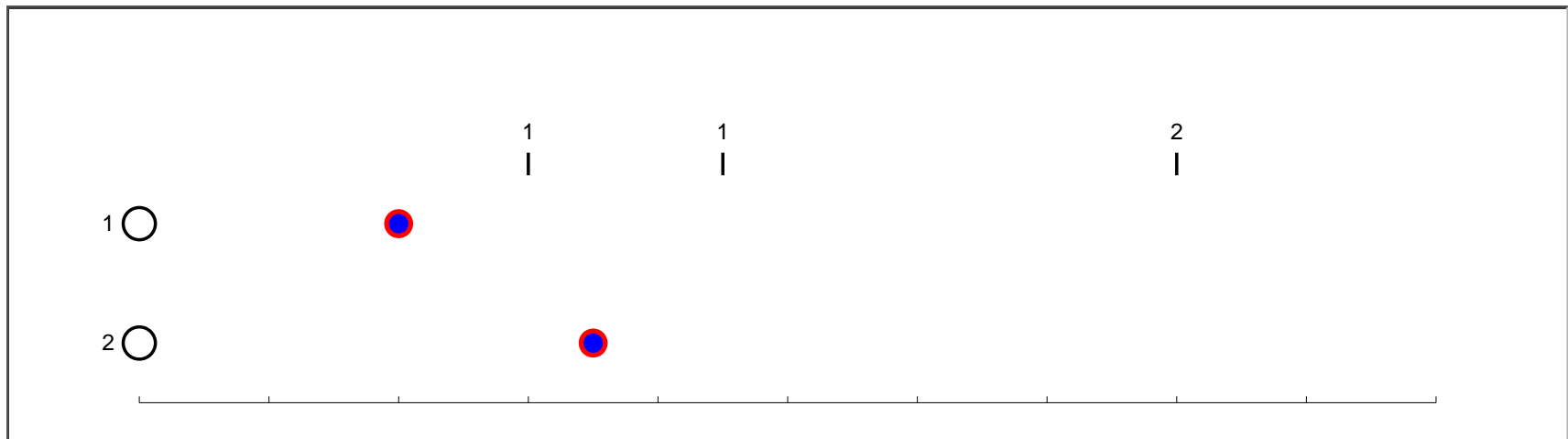
1

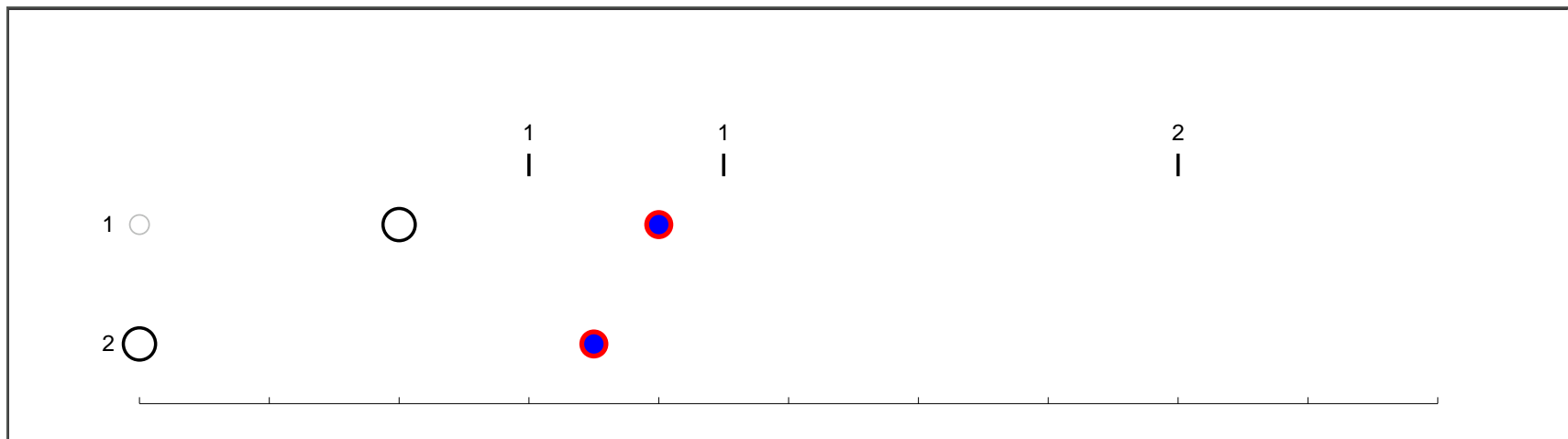


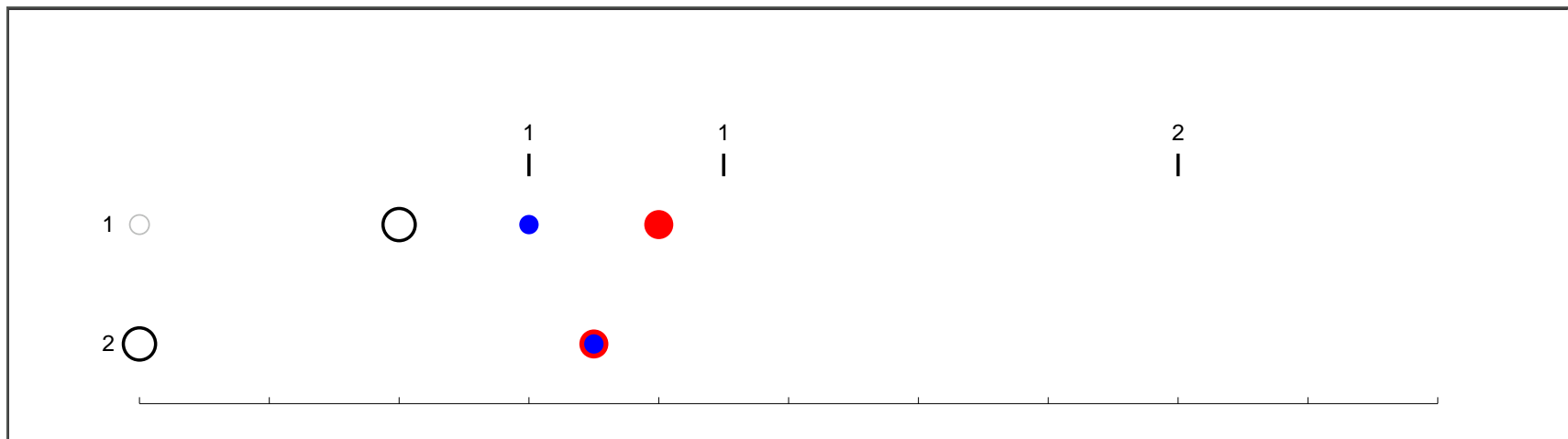
2

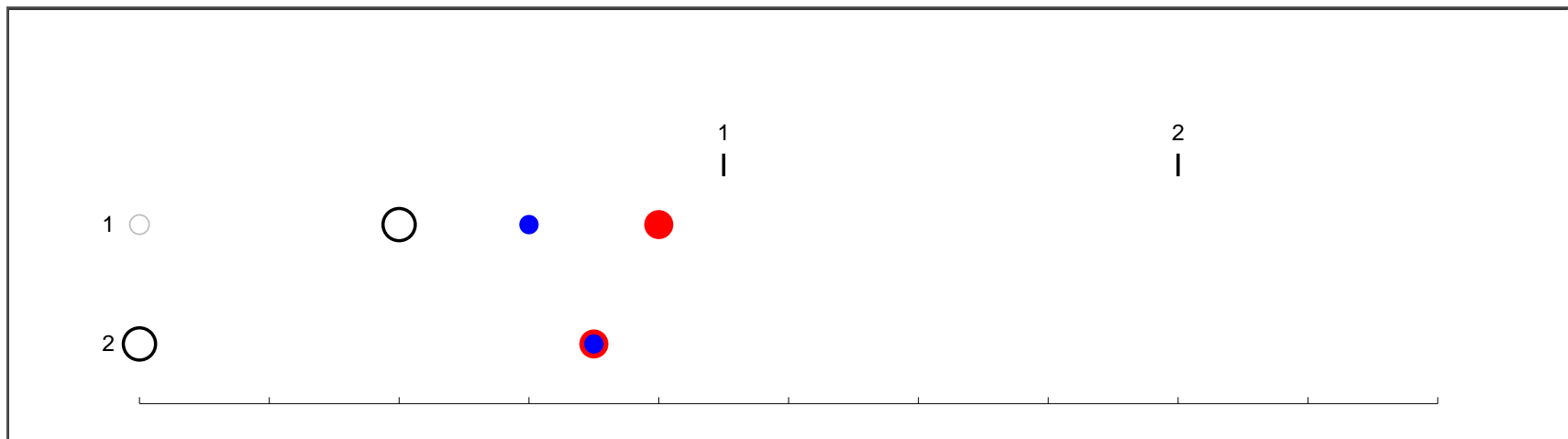


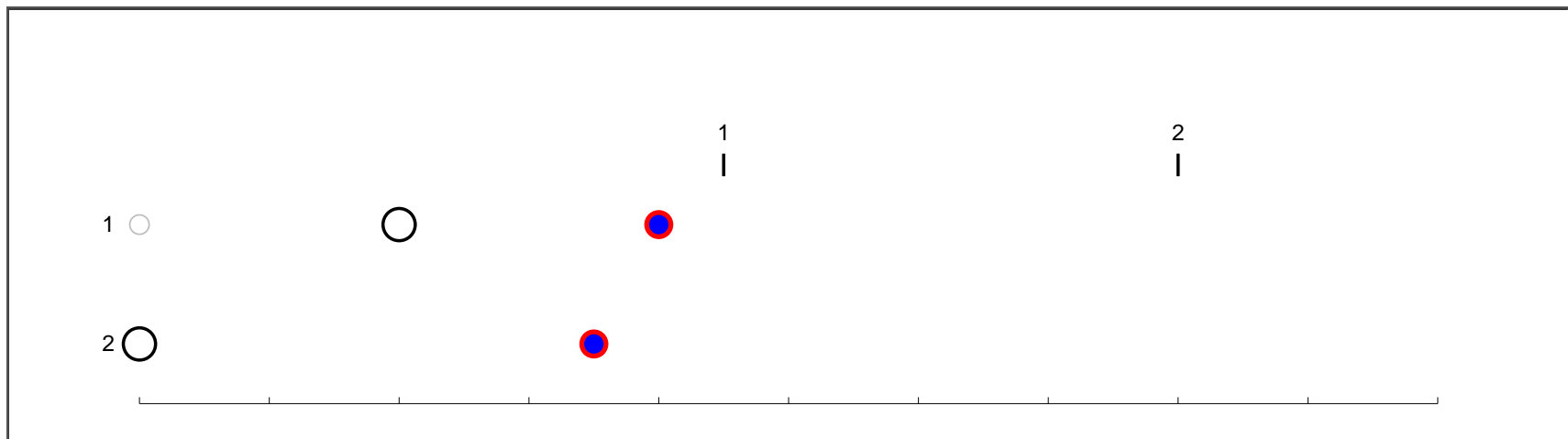


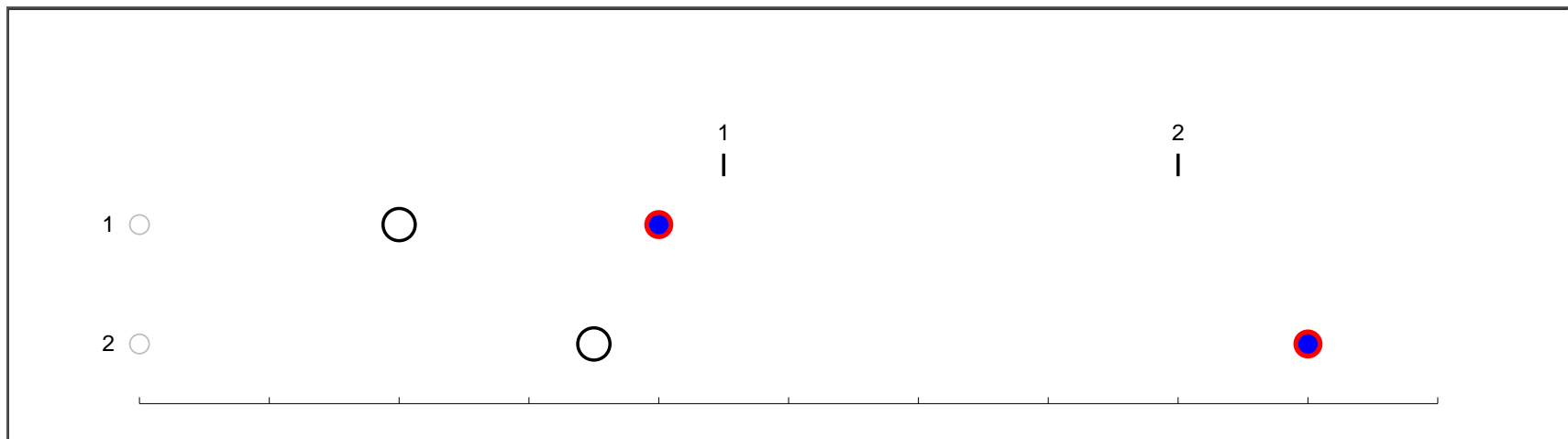


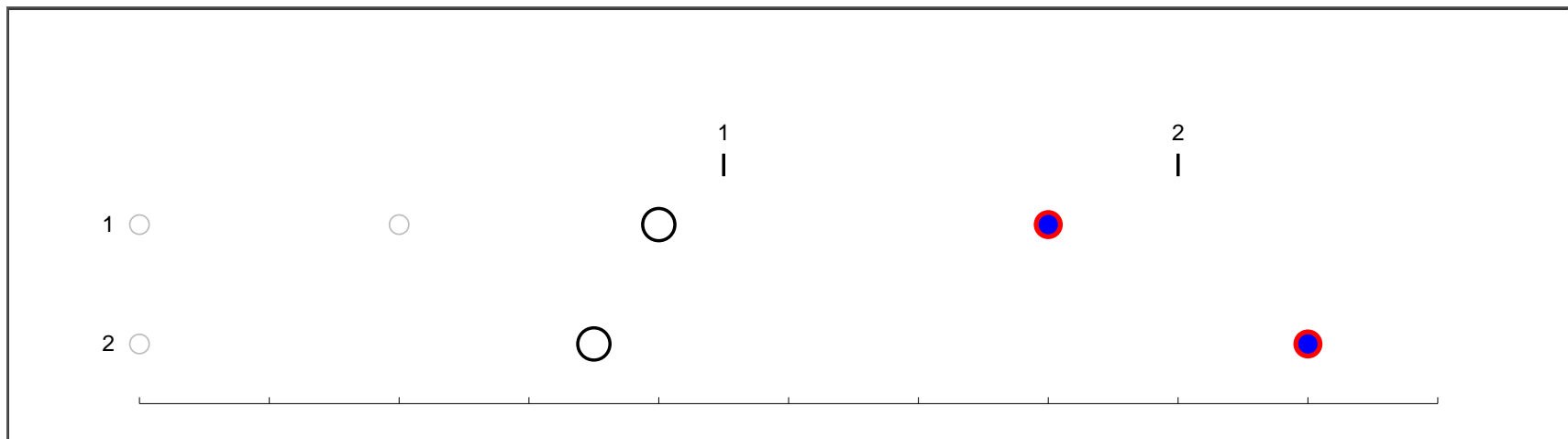


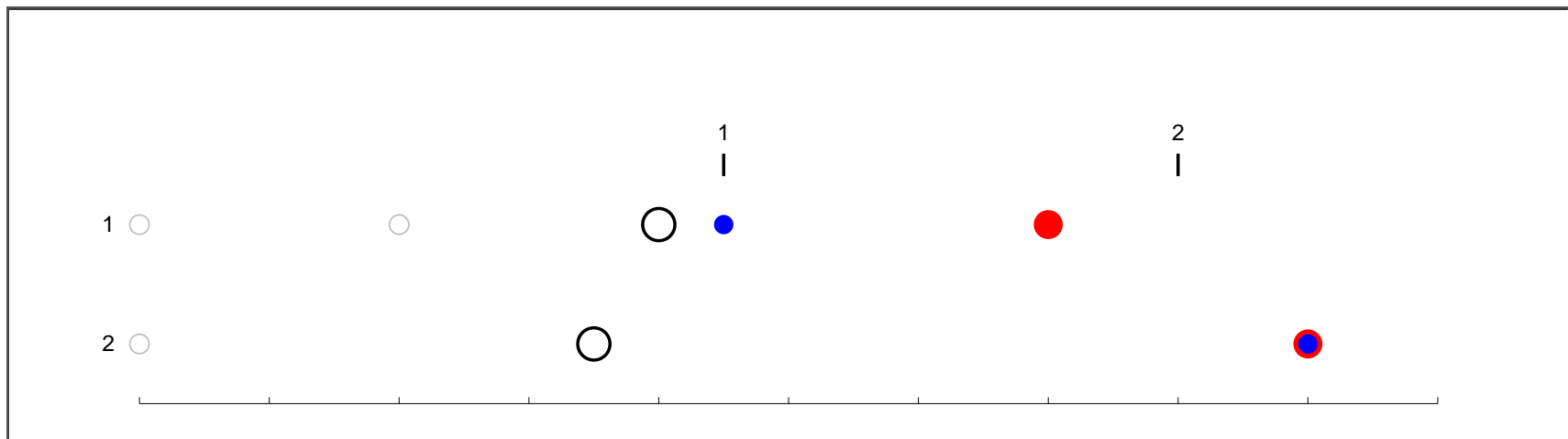


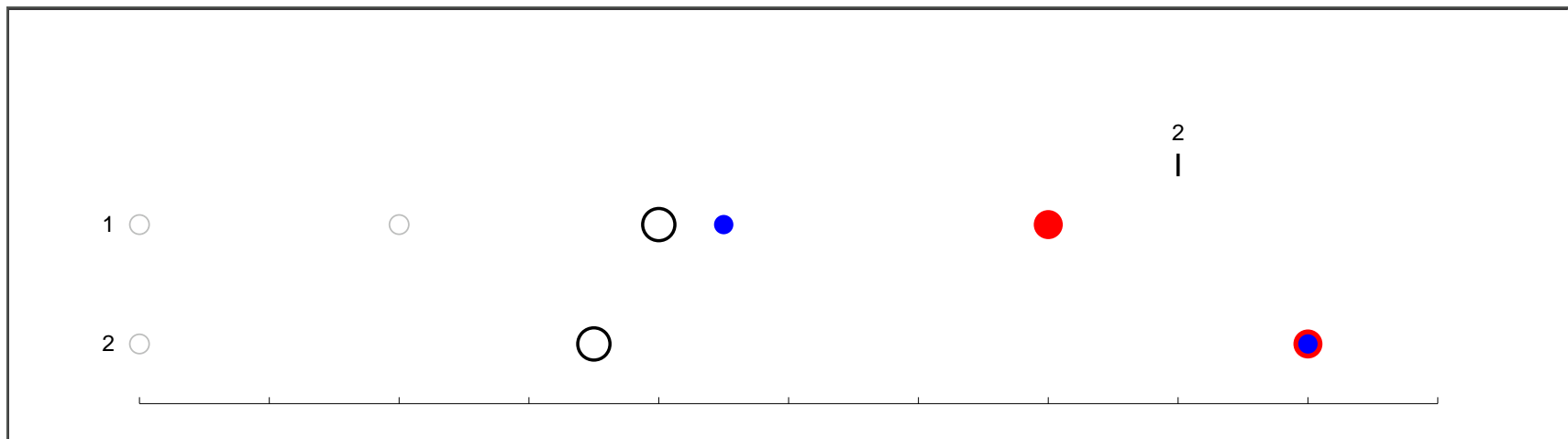


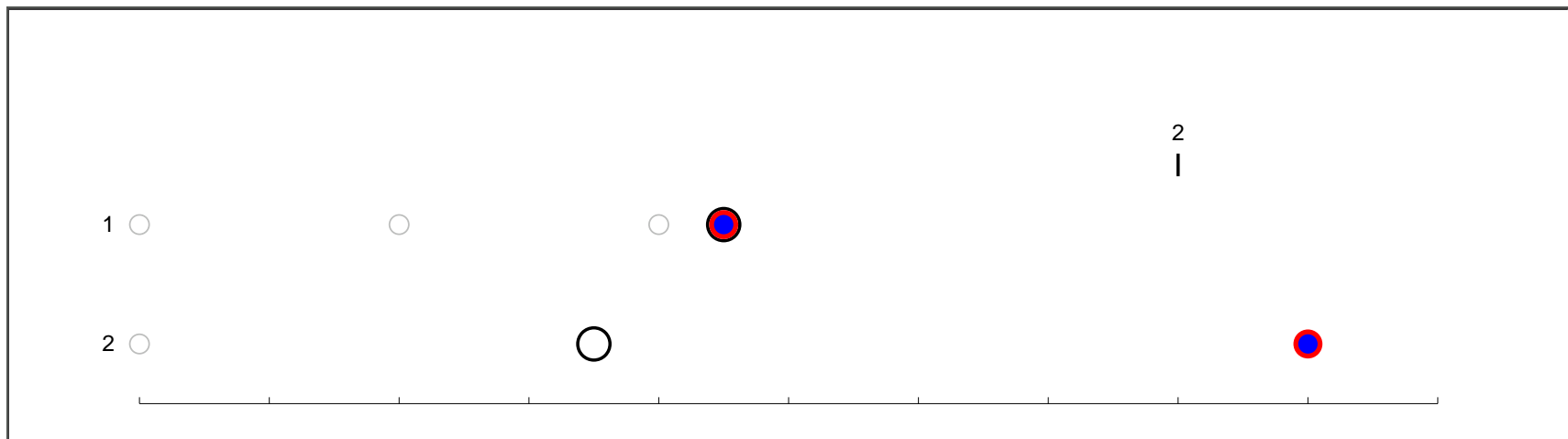


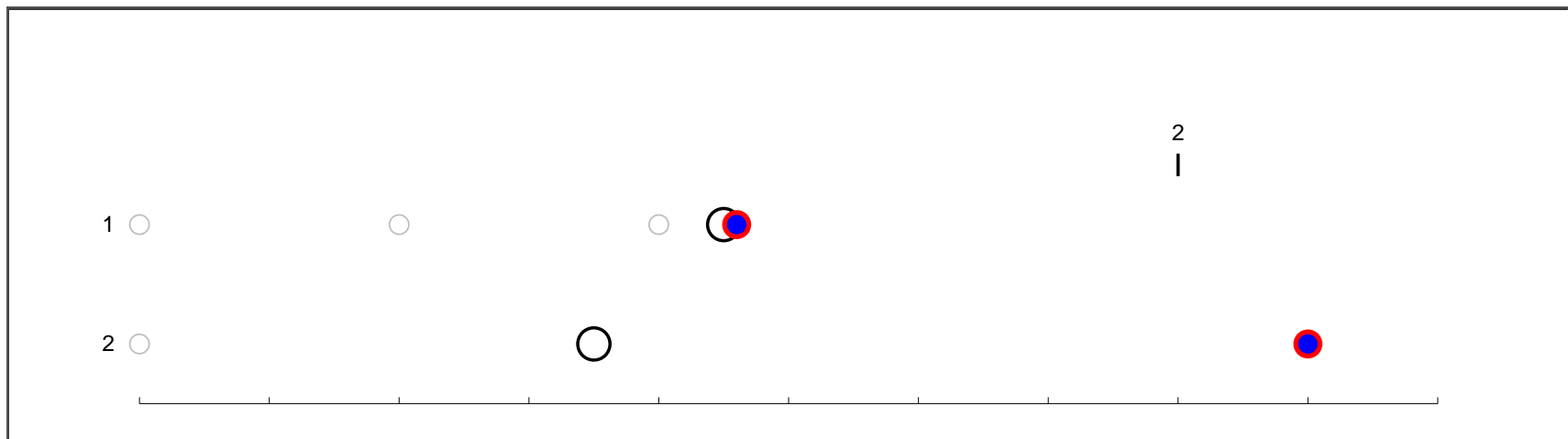


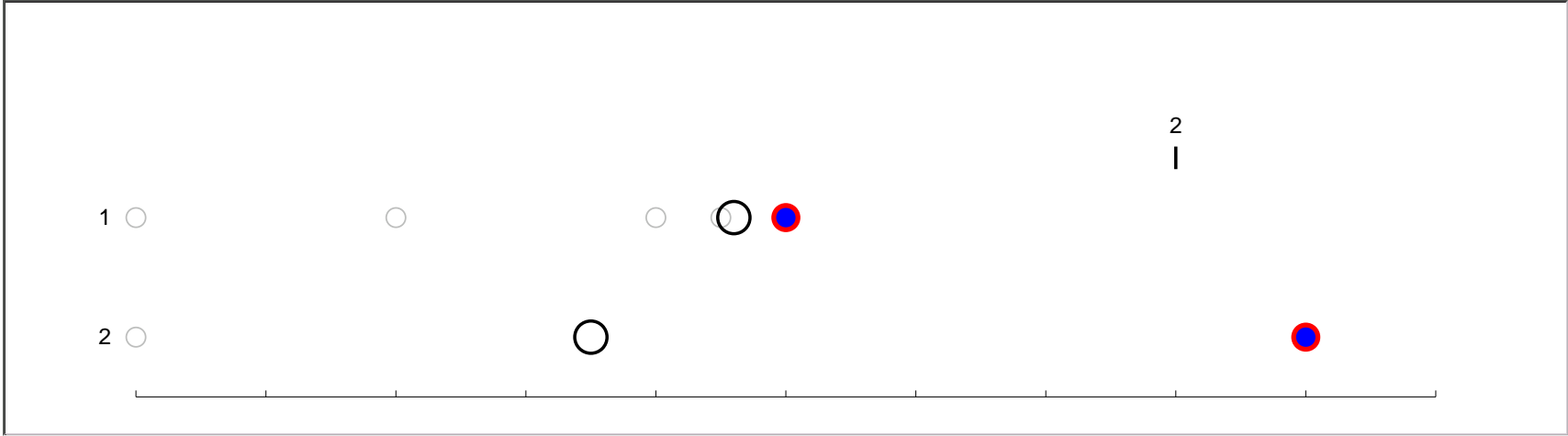


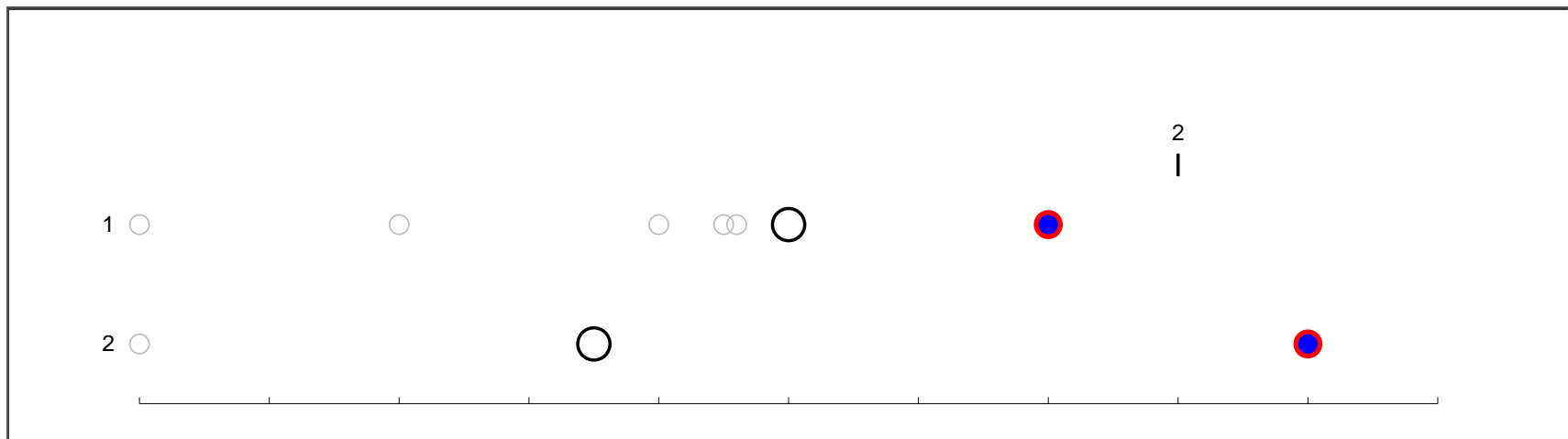


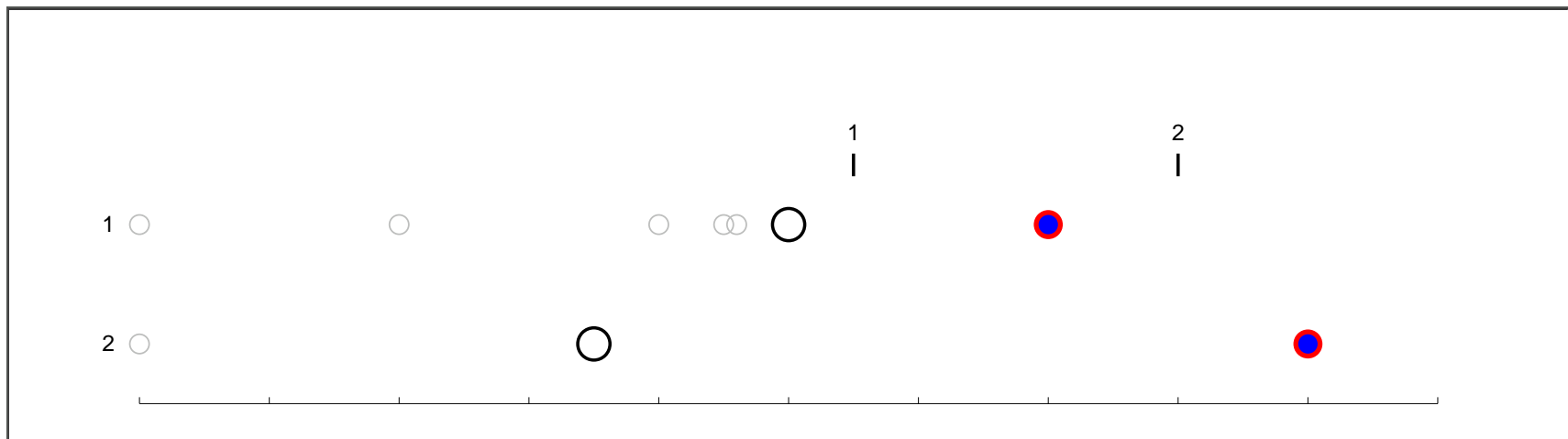


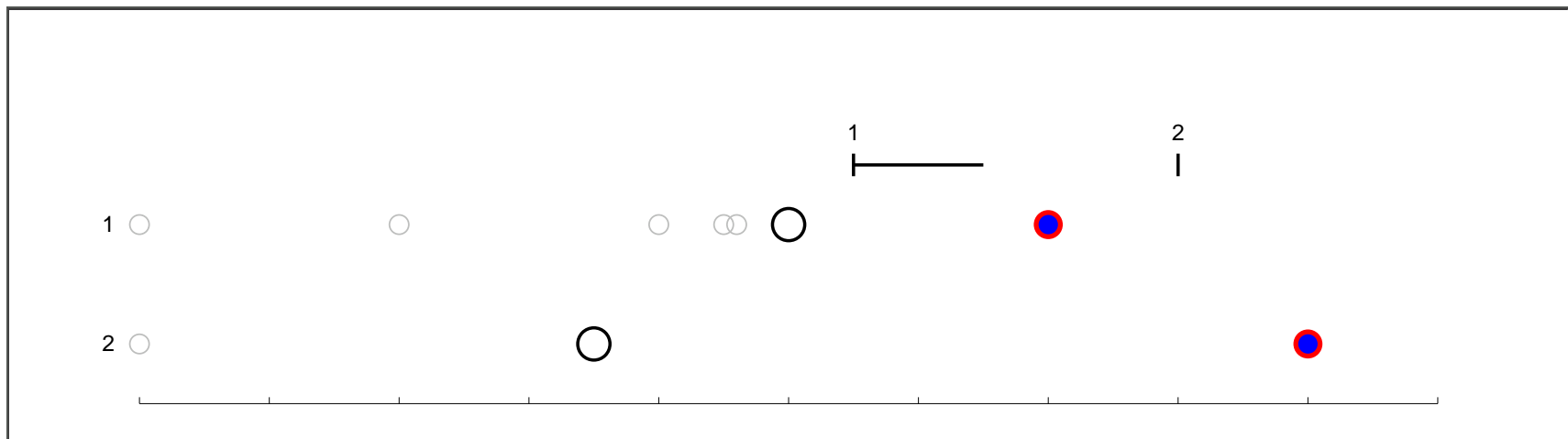


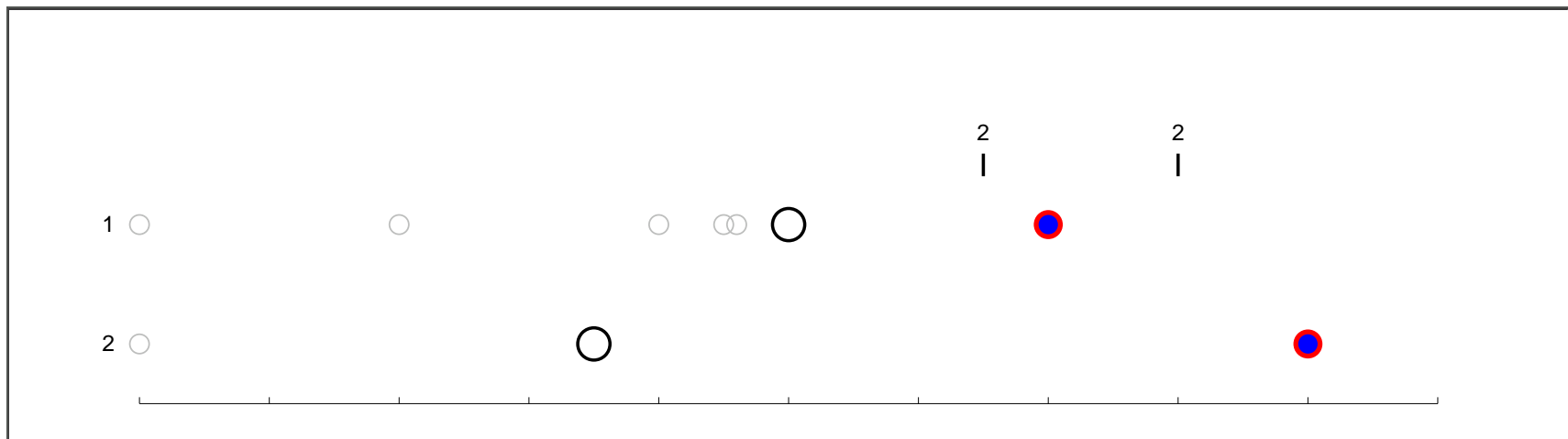


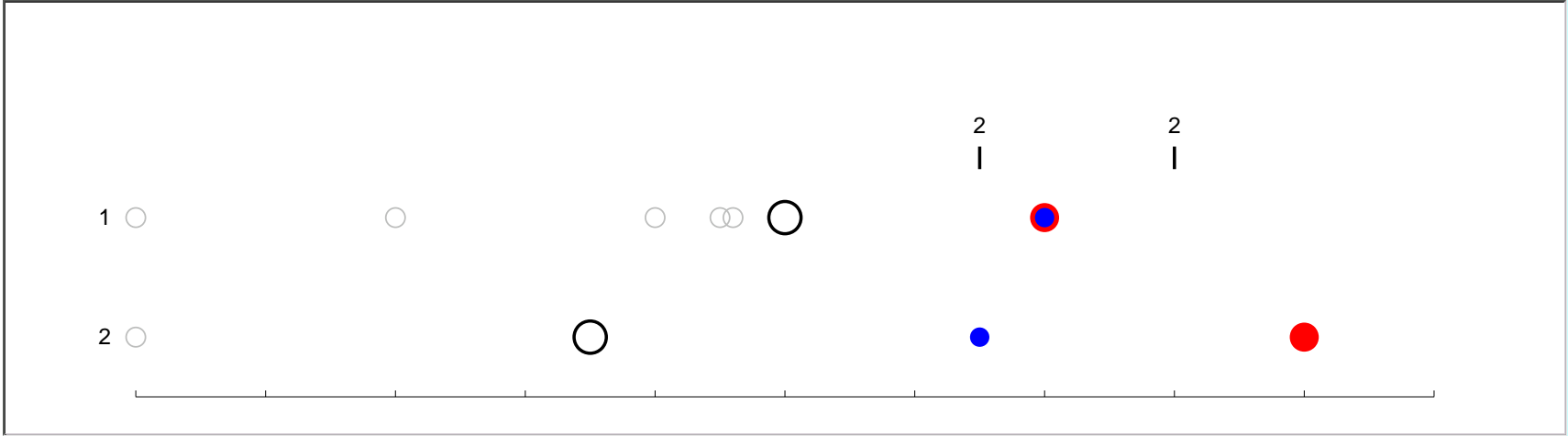


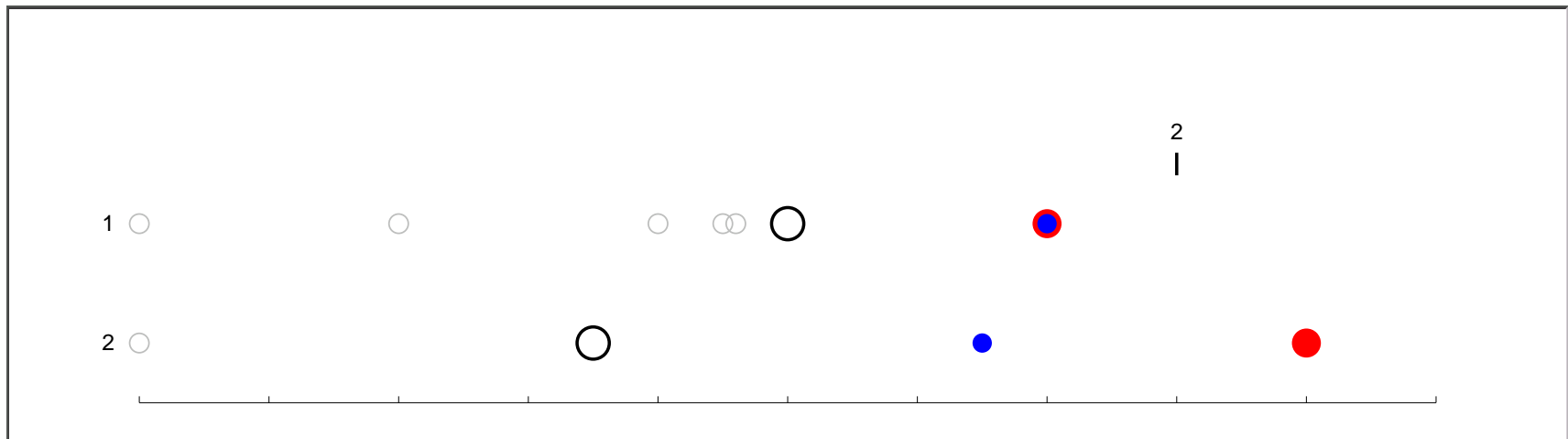


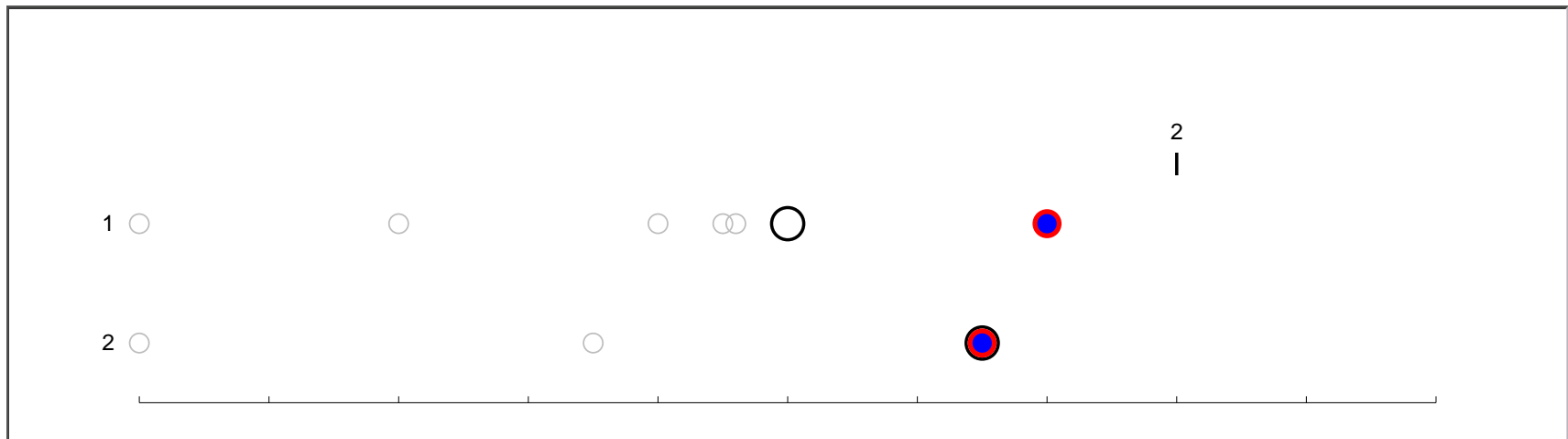













```
STATE { o }
```

```
BREAKPOINT {  
    SOLVE state  
    ik = gbar*o*(v - ek)  
}
```

```
LOCAL fac
```

```
PROCEDURE state() {  
    rate(v)  
    o = o + fac*(oinf - o)  
}
```

```
PROCEDURE rate(v (mV)) {  
    LOCAL a  
    a = alp(v)  
    tau = 1/(a + bet(v))  
    oinf = a*tau  
    fac = (1 - exp(-dt/tau))  
}
```

```
STATE { o }
```

```
BREAKPOINT {  
    SOLVE state METHOD cnexp  
    ik = gbar*o*(v - ek)  
}
```

```
DERIVATIVE state {  
    rate(v)  
    o' = (oinf - o)/tau  
}
```

```
PROCEDURE rate(v (mV)) {  
    LOCAL a  
    a = alp(v)  
    tau = 1/(a + bet(v))  
    oinf = a*tau  
}
```

```
BREAKPOINT {  
    if (t >= del) {  
        i = f(t-del)  
    }else{  
        i = 0  
    }  
}
```

← **at_time(del)**

(deprecated)

```
BREAKPOINT {  
    if (t >= del) {  
        i = f(t-del)  
    }else{  
        i = 0  
    }  
}
```

```
INITIAL {  
    on = 0  
    net_send(del, 1)  
}
```

```
BREAKPOINT {  
    if (on == 1) {  
        i = f(t-del)  
    }else{  
        i = 0  
    }  
}
```

```
NET_RECEIVE(w) {  
    if (flag == 1) {  
        on = 1  
    }  
}
```

TITLE minimal model of GABA_A receptors

COMMENT

Minimal kinetic model for GABA_A receptors

=====

Model of Destexhe, Mainen & Sejnowski, 1994:

(closed) + T \leftrightarrow (open)

The simplest kinetics are considered for the binding of transmitter (T) to open postsynaptic receptors. The corresponding equations are in similar form as the Hodgkin–Huxley model:

$$dr/dt = \alpha * [T] * (1-r) - \beta * r$$

$$I = g_{\max} * [\text{open}] * (V - E_{\text{rev}})$$

where [T] is the transmitter concentration and r is the fraction of receptors in the open form.

If the time course of transmitter occurs as a pulse of fixed duration, then this first-order model can be solved analytically, leading to a very fast mechanism for simulating synaptic currents, since no differential equation must be solved (see Destexhe, Mainen & Sejnowski, 1994).

```

PROCEDURE release() { LOCAL q
    :will crash if user hasn't set pre with the connect statement

    q = ((t – lastrelease) – Cdur) : time since last release ended

```

: ready for another release?

```

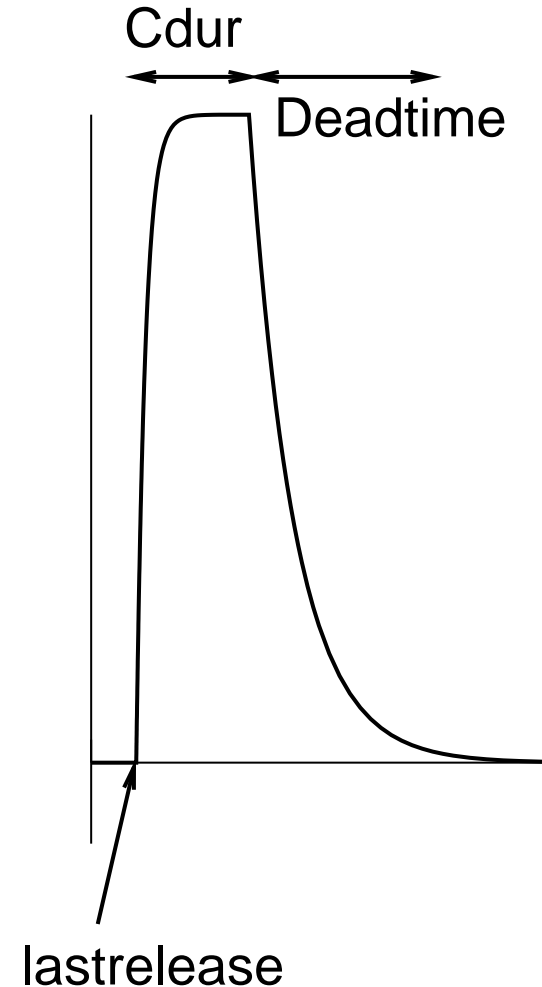
if (q > Deadttime) {
    if (pre > Prethresh) {      : spike occurred?
        C = Cmax                : start new release
        R0 = R
        lastrelease = t
    }
} else if (q < 0) {            : still releasing?
    : do nothing
} else if (C == Cmax) {       : in dead time after release
    R1 = R
    C = 0.
}

```

```

if (C > 0) {                  : transmitter being released?
    R = Rinf + (R0 – Rinf) * exp(– (t – lastrelease) / Rtau)
} else {                      : no release occurring
    R = R1 * exp(– Beta * (t – (lastrelease + Cdur)))
}
}

```



...

$$dr/dt = \alpha * [T] * (1-r) - \beta * r$$

where [T] is the transmitter concentration and r is the fraction of receptors in the open form.

...

```
INITIAL {  
    t0 = 0  
    r = 0  
}
```

```
DERIVATIVE state {  
    r' = (rinf - r)/rtau  
}
```

```
NET_RECEIVE(w) {  
    if (flag == 0) { : external spike, transmitter on  
        rinf = alpha*T/(alpha*T + beta)  
        rtau = 1/(alpha*T + beta)  
        net_send(Cdur, 1)  
    } else if (flag == 1) { :transmitter off  
        rinf = 0  
        rtau = 1/beta  
    }  
}
```

STATE {Ron Roff}

INITIAL {

Ron = 0 Roff = 0

Rinf = Alpha / (Alpha + Beta)

Rtau = 1 / (Alpha + Beta)

Rdelta = Rinf*(1 - exp(-Cdur/Rtau))

synon = 0

}

BREAKPOINT {

SOLVE release METHOD cnexp

g = (Ron + Roff)*1(umho)

i = g*(v - Erev)

}

DERIVATIVE release {

Ron' = (synon*Rinf - Ron)/Rtau

Roff' = -Beta*Roff

}

NET_RECEIVE(weight) {

if (flag == 0) { : spike - T on

synon = synon + weight

net_send(Cdur, 1)

}else{ : transmitter off

synon = synon - weight

Ron = Ron - weight*Rdelta

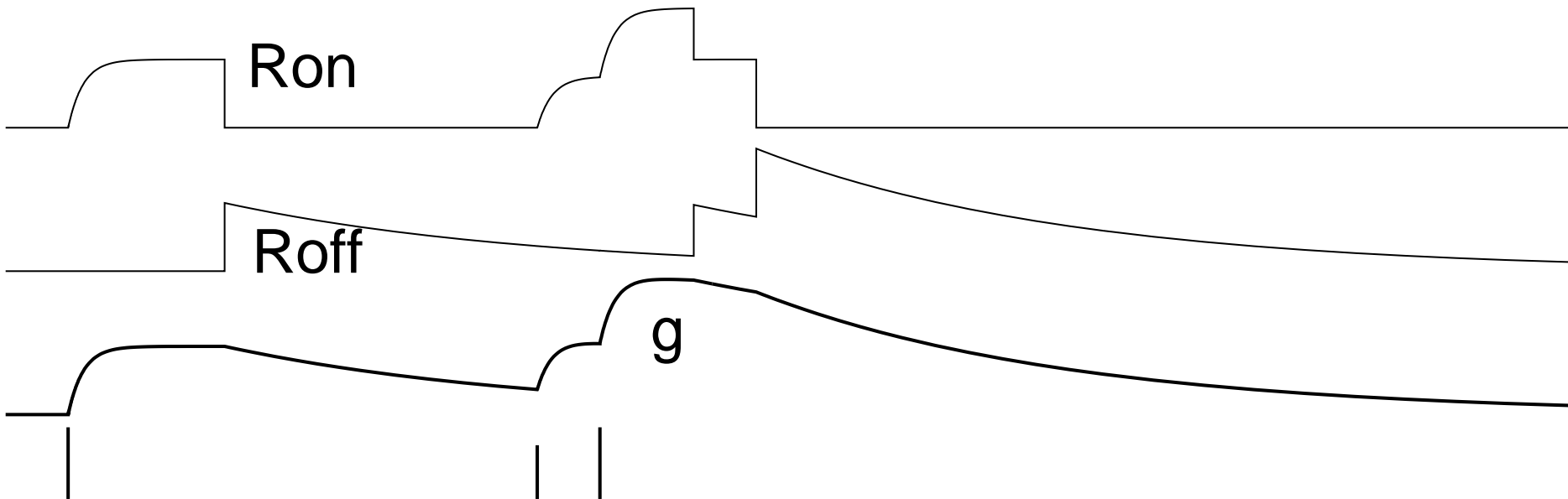
Roff = Roff + weight*Rdelta

}

Ron

Roff

g



```
setpointer gabaa[i], cell[j].axon.v(1)  
gabaa[i].Prethresh = -10
```



```
cell[j].axon { nc = new NetCon(&v(1), gabaa[i]) }  
nc.threshold = -10  
nc.delay = 0
```



```

proc advance() {
  fadvance()
  if (t == t1) { p() }
}

```



```

fih = new FInitializeHandler("ev()")
proc ev() {
  ccode.event(t1, "p()")
}

```

```

proc advance() {
  fadvance()
  if (soma.v(.5) > 10) { p() }
}

```



```

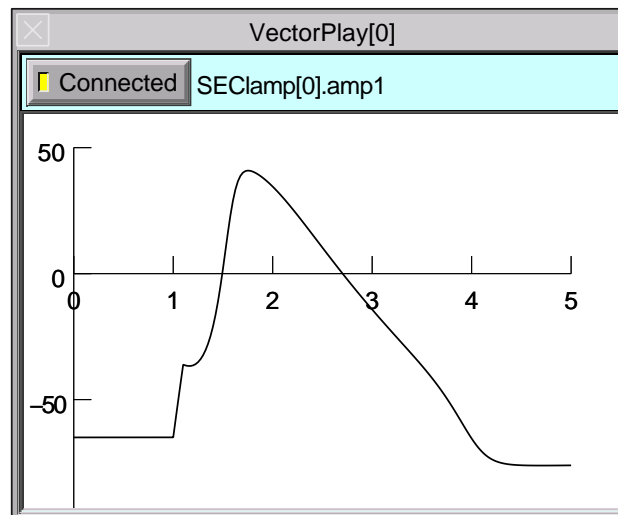
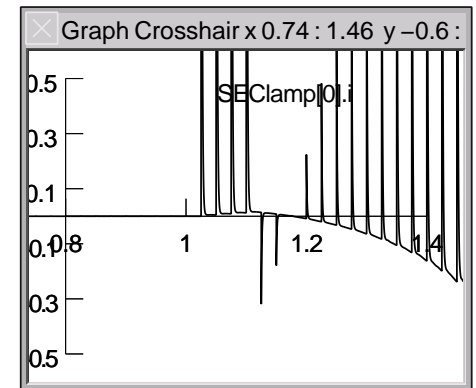
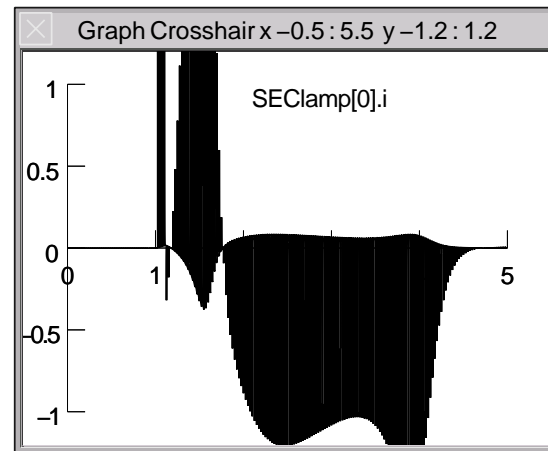
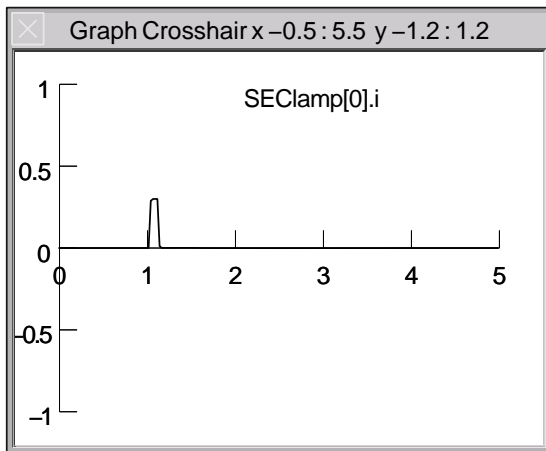
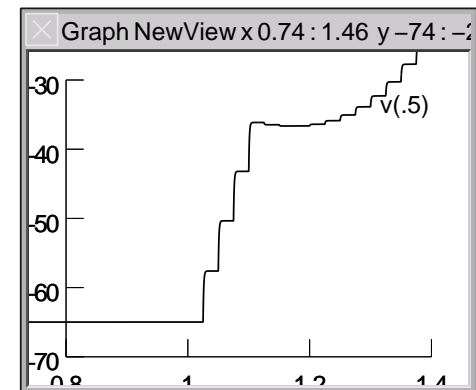
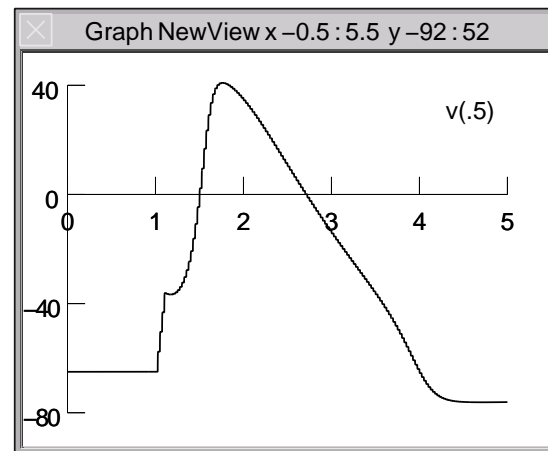
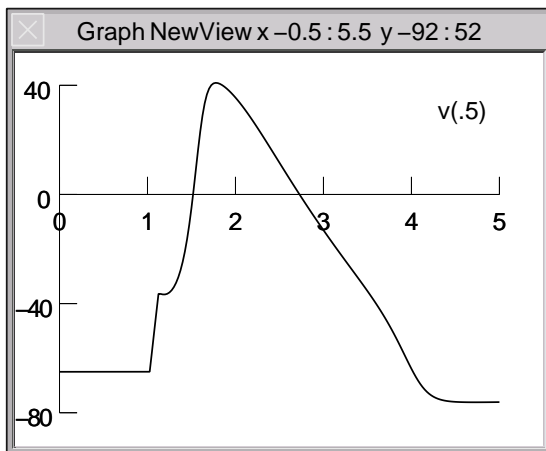
soma { nc = new NetCon(&v(.5), nil)}
nc.threshold = 10
nc.record("p()")

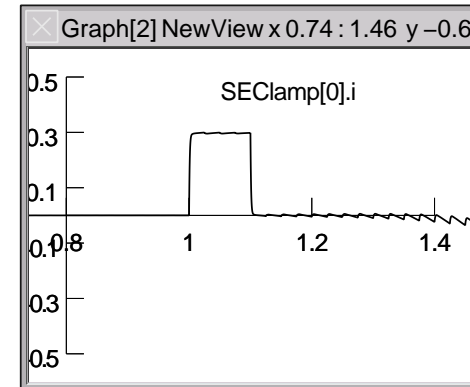
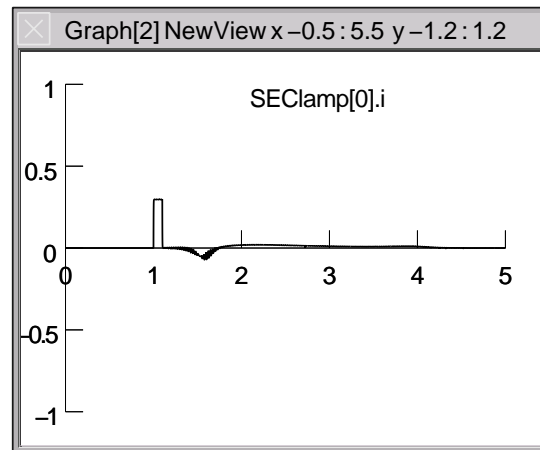
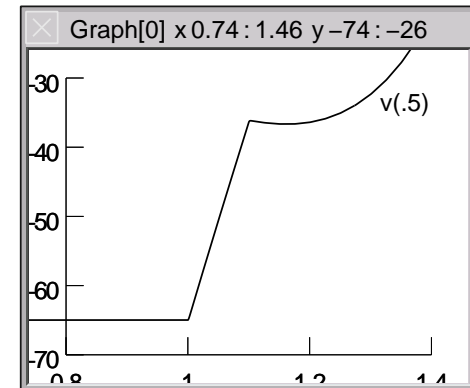
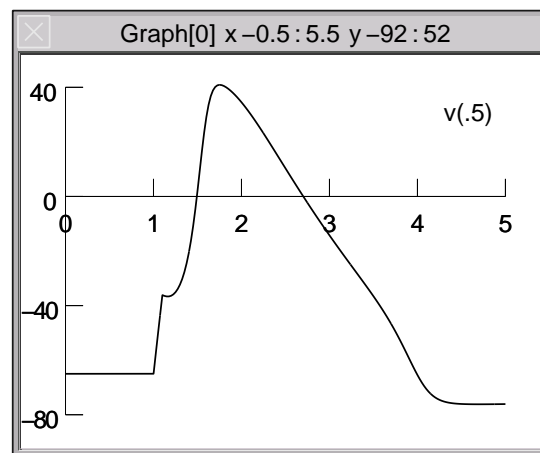
```

```

proc p() {
  // if ANY parameters or states
  // change then be sure to
  ccode.re_init()
}

```





```
soma vvec.play(&SEClamp[0].amp1, tvec, 1)
```