Neural Computation
Practical 2: Connecting together stimuli and neurons

David C. Sterratt

7th February 2002

1 Aims

This practical has two aims:

- To show you how to connect spiking stimuli to neurons and neurons to each other
- To investigate temporal summation

2 Connecting neurons together — NetCon

We have not yet connected neurons together. The modern form of connecting neurons together is in an event based simulation model. Events are usually things like action potentials, as it is the events (rather than, for example, the specific voltage levels) that we need to pass or communicate between neurons. By using an event based model, we can dramatically reduce the amount of inter-neuron communication. This is important in simulation time optimisation and in simulations on parallel machines.

To link neurons together we use a `NetCon` object. The `NetCon` object is associated with a particular *source* of events (usually a soma, where action potentials produce the binary events) and has a particular *target* (usually a synapse). However, it can also have other sources (for example an artificially-generated spike train) or targets (for example a spike recording device).

In our first program using `NetCon`, we will use a source of spikes called a `NetStim`. First of we must create a section for the `NetStim` point process to live on. (Conceptually, it does not necessarily live there; however Neuron demands that it is associated with a section.) To make our simulation more interesting later on, we will construct the simple two-compartment neuron used in the second assignment.

```
load_proc("nrnmainmenu")
nrnmainmenu()

create somaA
somaA insert  hh
somaA {
  diam = 22
  L = 37
  Ra = 100
}
create dendA
dendA insert pas
dendA {
  diam = 2
  L = 500
  nseg = 10
  g_pas = 0.0001
}
connect somaA(1), dendA(0)
```

```
access dendA
```

Now we can create our `NetStim`:

```
objref spikesource
spikesource = new NetStim(0.5) // Location of NetStim is arbitrary
spikesource.interval = 10      // ms (mean) time between spikes
spikesource.number = 100      // (average) number of spikes
spikesource.start = 0         // ms (mean) start time of first spike
spikesource.noise = 0         // range 0 to 1. Fractional randomness
                              // 0 deterministic, 1 intervals have decaying
                              // exponential distribution
```

Note that the text after the `//` is a comment; NEURON ignores this text.

Now we need to place a synapse on our `dendA` compartment. In the second assignment we used an `AlphaSynapse`. This allowed us to create *one* pulse of postsynaptic conductance change at a specified time. We would like to change the conductance at the times specified by the `NetStim` input to the synapse. To do this we need to use a different type of synapse. One type of synapse that works is the `Exp2Syn` object:

```
objref synapse
synapse = new Exp2Syn(0.95) // Inserts ExpSyn 0.95 of way down dendA
synapse.tau1 = 1            // ms rise time
synapse.tau2 = 2            // ms decay time
synapse.e = 0               // mV reversal potential
```

This is a double-exponential synapse. Its conductance varies according to

$$g_{\mathrm{s}} = \overline{g}_{\mathrm{s}} \frac{1}{N} (\mathrm{e}^{-t/\tau_2} - \mathrm{e}^{-t/\tau_1})$$

where $\overline{g}_{\mathrm{s}}$ is a weight (see later) and N is a normalisation factor that ensures that the peak of the curve is 1. If τ_1 and τ_2 are the similar, the function is almost the same as the alpha function with the same time constant. The values here are suitable for an excitatory AMPA synapse.

Now we need to connect the `NetStim` and the `Exp2Syn`. We do this using a `NetCon`:

```
objref connection
thresh = 10
delay  = 0.1
weight = 0.002
connection = new NetCon(spikesource, synapse, thresh, delay, weight)
```

This links the source `spikesource` to the target `synapse`. The `delay` parameter (the fourth parameter) specifies the delay in milliseconds of the connection. The `weight` connection specifies the strength of the connection. The `thresh` parameter is important when the source is a real neuron rather than an artificial one (see section 4).

3 Temporal summation

Now run the simulation for 50 ms and observe the membrane potential in the soma and the distal end of the dendrite. You should see regular EPSPs 10 ms apart in the dendrites and attenuated EPSPs in the soma. You can look at the synaptic conductances by opening a **Current Graph**, right-clicking and selecting **Plot What?** then **Show→Object refs** the clicking on `synapse` and selecting `g` from the middle column.

These conductances are responding to the events generated by the `NetStim`. You should be able to convince yourself that they are created at intervals of 10 ms starting at 0 ms, as specified by the NetStim.

Now change the delay of the connection from the `NetStim` to the synapse:

```
connection.delay = 5
```

You should see that the spikes are delayed by 5 ms, though the interval between inputs is the same.

In general, a single synaptic input is not enough to cause a neuron to fire. Many spikes arriving within a short period of time are usually needed. Reduce the interval between the input spikes by typing

```
spikesource.interval = 0.5
```

You should now see that a train of action potentials is fired by the soma. Try intervals between 0.5 ms and 10 ms to see the effects.

Try increasing the synaptic strength by typing

```
connection.weight=0.04
```

See what happens to the number of action potentials produced for a certain input spike interval.

4 Connecting neurons together

We'll now make a second neuron, identical to the first

```
create somaB
somaB insert  hh
somaB {
  diam = 22
  L = 37
  Ra = 100
}
create dendB
dendB insert pas
dendB {
  diam = 2
  L = 500
  nseg = 10
  g_pas = 0.0001
}
connect somaB(1), dendB(0)
```

The principle for connecting neurons is similar for connecting spike sources to neurons. To connect `somaB` to `dendA` we first need to make a synapse onto `dendA`:

```
objref synapseAB
dendA {
  synapseAB= new Exp2Syn(0.95) // Inserts ExpSyn 0.95 of way down dend
  synapseAB.tau1 = 1            // ms rise time
  synapseAB.tau2 = 2            // ms decay time
  synapseAB.e = 0               // mV reversal potential
}
```

Then we'll need connect the synapse to `somaB`. Here it's a bit different to the spike source above; the threshold at which we count the neuron as having fired is important. The threshold of 10 mV stored in the variable `threshold` should do. We also need to specify *what quantity* we are measuring from the soma and where we are measuring it from. The `&somaB.v(0.5)` part of the commands below tells us that we are measuring the membrane potential halfway down the `somaB` section. The `&` tells us we are dealing with an object reference — don't worry about this.

```
objref connectionAB
connectionAB = new NetCon(&somaB.v(0.5), synapseAB, thresh, delay, weight)
```

Now we do the same for the synapse from somaA to dendB.

```
objref synapseBA
dendB {
  synapseBA= new Exp2Syn(0.95) // Inserts ExpSyn 0.95 of way down dend
  synapseBA.tau1 = 1             // ms rise time
  synapseBA.tau2 = 2             // ms decay time
  synapseBA.e = 0                // mV reversal potential
}
objref connectionBA
somaA connectionBA = new NetCon(&v(0.5), synapseBA, thresh, delay, weight)
```

Note that we can use a slightly different syntax when creating the `NetCon`.

Now you can play with this pair of reciprocally-connected neurons with input to one. You could try varying the delay or weights between the neurons, the amount of input and change the properties of the synapses (e.g. make them inhibitory). Have fun!