Practical 4: Integrate and fire neurons

Neural Computation 2002-2003. Mark van Rossum

29th January 2004

1 Introduction to matlab

Start matlab with 'matlab' on the command line. Commands to matlab can be entered on its command line (handy for small stuff and plots), or one can create an m-file, which contains a program or procedure. We'll do the latter.

If you are not familiar with MATLAB: Briefly go through the handout and perhaps the introduction section of the on-line manual. The on-line help is very good. It is good to check each command used here to see its capabilities.

Despite the discouragement to write 'for' loops in matlab, the time loop can often only be written as a for loop...

2 Basic integrate and fire

We will construct an integrate-and-fire neuron. Parameters: $V_{rest} = -70$mV, $V_{thr} = -50mV$, $V_{reset} = -70$mV, $\tau_m = 20ms$. Simulation parameters: time-step of 0.1 ms. Total simulation time 1 to 10 s. (You can take $R_m = 1$).

We want to record the spike times and the inter spike intervals of the spike train. Store the spike-train in an array of '1's and '0's, initialise an array
`spiketrain=zeros(1, nsteps);`
where nsteps is the total number of time-steps. This creates an array of nsteps length filled with zeros. Note, `zeros(nsteps)` would create an nsteps×nsteps matrix. Dimension of variables can be checked in 'Workspace' tab of the main window.

For debugging purposes it is good to record the membrane potential in a array as well:
`vmem=vrest*ones(1,nsteps);`

The array which contains the interspike intervals does not have fixed length, as the number of spikes per run can differ. A growing array can be made by initializing
`intervals=[];`
now elements can be added to an array with
`intervals=[intervals new_interval];`

- Given a certain input current, how does the membrane potential change in one time-step (see also lecture notes)?

Based on this line write the simulation loop. Make sure to end all lines within the loop with a";". This prevents output being printed.

- Apply a stimulus which slowly increases with time. Inspect the firing pattern and the membrane potential.[1]

We would like to know how rapid the cell fires in response to a certain current injection. The instantaneous firing frequency is a bit ill-defined for events like spikes. Nevertheless, one trick is to plot $1/(t_i - t_{i-1})$ vs. t_i, where t_i is the time of spike i.

- Measure the F/I curve this way.

3 Noisy integrate and fire neuron

Next, we make the model more realistic by adding noise.

- Add a noise current to the input of the integrate and fire neuron (`itotal=istimulus+sigma*randn`), where `sigma` is the standard deviation of the noise. Plot the distribution of intervals for a few different constant stimulus currents and noise levels. An easy way to plot a histogram is to use `hist(intervals)`. Explain its shape.

- Take a constant stimulus. Plot the autocorrelation of the spike train ('`xcorr(spiketrain)`'). xcorr also calculates the cross correlation, hence its name. But with just one array as argument it calculates the autocorrelation.) Explain its shape (careful, some structure can only be seen after zooming in). What autocorrelation would you expect for a precisely periodically spiking neuron?

- EXTRA: Suppose we would like to write a more general integrate-and fire simulator in which the timestep is user-adjustable. How should one normalise the noise term such that the effect of the noise does not depend on the timestep? (Tricky)

4 Stimulus locking

Using the noisy neuron we research the precision of firing, and the interaction between stimulus and noise. In principle, we can simulate this by running the simulation a few times (with different random seeds) to see how the spike time varies from run to run. Here, however, we run the simulation for a group of independent neurons simultaneously. This is equivalent to repeating one-neuron simulations and it is better suited for matlab.

Save the work you have done sofar. Change your program, so that you have a 'vector' of neurons. For instance, the history of membrane potentials becomes a matrix.

```
ncells=50;
vmem= vrest*zeros(ncells, nsteps);
```

Similarly the stimulus current becomes a vector. Matlab allows to update many matrix-elements at once. The membrane potential update rule becomes
`vmem(:, itime) = ....`
The ':' means 'for all these elements'.

[1] Useful plot commands:

```
plot(x) % plot array x
plot(x,y) % plot values in y-array vs.  x-array
```

Unless specified otherwise, matlab will create one plot window and new plots overwrite old ones:

```
hold on % keep old plot, plot output of next plot command in the same window
figure % create new plot window, and make it current
close all % close all plot windows, good in the start of a program
```

Threshold detection now is easiest to implement with '`find`'
`newspikes = find(vmem(:,itime)>vthr);`

This find command returns a vector with the neurons that crossed the threshold, for example
`newspikes=[2 4]`
This can be used an argument to update the spike trains and reset the appropriate membrane potentials:
```
vmem(newspikes, itime)=vreset;
spiketrain(newspikes,itime)=1;
```

Create a raster plot (like on page 48 of the lecture notes). One way of doing that is
```
hold on
for ic=1:ncells
sp=find(spiketrain(ic,:)>0);
spy=0*sp+ic;
plot(sp,spy);
end
```

In order to see the response to a variying current, we introduce a second random number. So now there are two random currents, a common one (the stimulus), and a noise one:
`i = ones(1,ncells)*istim + sd_noise*randn(1,ncells)+sd_stim*randn;`

First fix the noise level to a decent value. Now try a few different stimuli. Which stimuli give more precise responses?

With `sound(spiketrain(1,:),10000)` you can listen to your spiketrain. (Careful, save first. It might crash matlab on DICE machines). You might need headphones.

5 Adding adaptation

We add adaptation to the model as follows:
1) If the neuron spikes, increase the Ca concentration with 1.
2) At every timestep the Ca concentration decays exponentially back to zero with a time-constant of 50 ms.
3) The Ca activates a KCa current as $I_{Kca} = [Ca].g_{KCa}(V_{KCa}^{rev} - V)$. Take $V_{Kca}^{rev} = V_{rest}$ and $g_{kca} = 2$.

Apply step currents of different strengths. How much and how quickly does the neuron adapt?