Spiking Neurons

Computer Exercises using the NEURON Simulator

SICSA Cognitive Computation Summer School

August 2013, Stirling, U.K.

Author: Dr Bruce Graham, Computing Science & Mathematics, University of Stirling, U.K.

URL: www.cs.stir.ac.uk/~bpg/

Email: b.graham@cs.stir.ac.uk

Computer Exercises using NEURON

Table of Contents

Overview

These exercises make use of the **NEURON** simulator (www.neuron.yale.edu). A number of the exercises are based on examples from the book, "**Principles of Computational Modelling in Neuroscience**" by Sterratt, Graham, Gillies and Willshaw (CUP, 2011). The book will be referred to as **PCNM** in the following text.

Running exercise simulations

The following NEURON code archive (**nrnzip**) files are provided via the "**NEURON exercise code**" link on the web page, **http://www.cs.stir.ac.uk/~bpg/CNschools/COGCOMP2013/:**

FIcurve, EIosc, EIbalance, EIbalnet, STDPtiming, STDPsequence, AM

To run each exercise:

1. open this web page in a browser (and leave this page open between exercises)
2. click on the required NEURON file (as named in the exercise instructions)
3. select open in the browser popup (it should be recognised as a NEURON archive file)
4. select "Yes" (just press return) to run NEURON in the window that appears

Though it will not be necessary, if you are interested, you can extract all the code files for any exercise by putting the nrnzip file (eg **AM.nrnzip)** into suitable "uncompress" software (they are really just standard ZIP files).

Other (optional) tutorial material

If you would like more experience with NEURON before (or after) trying these exercises, then the following tutorial material is available.

NEURON GUI tools: Neuron models and small networks can be constructed in NEURON using in-built GUI tools, with no programming required. Links to relevant tutorials on these tools, available on the NEURON website, are provided on the computer exercises web page.

Learning Hoc: To gain basic experience in constructing NEURON simulations by programming them in the scripting language "hoc", try the series of tutorials by Gillies and Sterratt. The tutorials are available in PDF format on the exercises web page, and all associated hoc code is also provided.

1. Frequency-Input Current (F-I) Firing Curves of a Neuron

The Frequency-Current (F-I) firing curve of a neuron is classified as either Type I, if the frequency of firing increases smoothly from zero once the firing threshold is reached, or as Type II, if the neuron suddenly starts firing with a finite frequency after the threshold.

In this exercise we will see that the presence or absence of particular ion channels in the membrane can determine the type of the F-I curve of a neuron. The code and GUI are already provided for you using example code from the PCMN book.

Running the book code:

Open **FIcurve.nrnzip** to run the code associated with figure 5.9 in PCNM. This demonstrates Type I and Type II firing in a neuron, controlled by the presence or absence of the potassium A-type current.

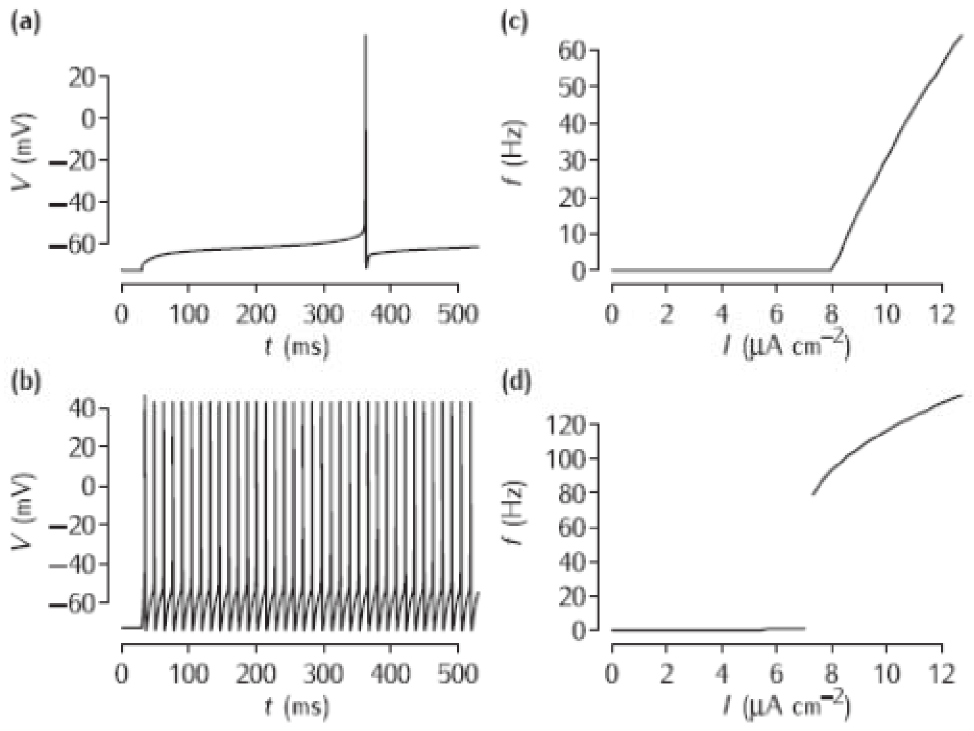


Figure 5.9: (a) First spike in a Type I neuron. (b) Start of spiking in a Type II neuron. (c) f-I curve for Type I. (d) f-I curve for Type II.

Once the simulation file is open, these are the steps to reproduce each panel in the figure:

Panel (a)

1. In the top **Parameters** window, click on the checkbox next to **Type I parameters**
2. Click on **Set Type I threshold current**
3. Click on **Init & Run** in the **Run Control** window
4. The voltage plot corresponding to Panel (a) above is in Graph[0] in the GUI (click on it to bring it to the front if it is partly obscured by other windows). The plot windows below this graph show the sodium, potassium and leak conductances and associated currents that

underpin this voltage response. Note that the potassium currents are in the opposite direction to the sodium current (which is responsible for generating the action potentials).

Panel (b)

1. In the top **Parameters** window, click on the checkbox next to **Type II parameters**
2. Click on **Set Type II threshold current**
3. Click on **Init & Run** in the **Run Control** window
4. The voltage plot should now look like Panel (b)

Panel (c)

1. In the top **Parameters** window, click on the checkbox next to **Type I parameters**
2. Click on **Plot** in the **Grapher** window
3. Simulations for a number of levels of current injection will be run and the firing frequency will be plotted against current in the **Grapher** window. Once the simulations have run, to make sure you can see these results, right-click in this graph area and select **View... -> View = Plot.** This plots a count of the number of action potentials (rather than frequency) in the simulation period (1050 ms) against the stimulation current amplitude. This model cell is slightly different from the one that produce fig. 5.9 above, but it has qualitatively the same firing characteristics.

Panel (d)

1. In the top **Parameters** window, click on the checkbox next to **Type II parameters**
2. Click on **Plot** in the **Grapher** window
3. Simulations for a number of levels of current injection will be run and eventually the firing frequency will be plotted against current in the **Grapher** window. Once again, when the simulations have run, to make sure you can see these results, right-click in the Grapher's graph area (APCount[0]) and select **View... -> View = Plot.**

Question 1: What cell parameters are different between the Type I and Type II examples?

Supplementary Question (for later research): Can you explain why the difference in firing characteristics arises? You can begin to answer this now by generating the Panel (a) results and comparing the two potassium currents (delayed rectifier "csk" and A-type "cska") between action potentials: how do they differ?

2. Simple Excitation-Inhibition (E-I) Oscillator

A network of two neurons forming an E-I oscillator is modelled.

Each neuron contains a cell body that is a cylinder 30um in diameter and 30um long, and contains sodium, potassium and leak ion channels, and so can produce action potentials if stimulated to threshold. A 1000um long and 2um thick dendrite is attached to the cell body and is modelled as 11 compartments, each of which contains only leak ion channels.

The excitatory cell connects to a synapse on the dendrite of the inhibitory cell, and the inhibitory cell connects back to a synapse on the dendrite of the excitatory cell. There is a connection delay between each cell. The excitatory cell also is stimulated by a constant current injection to the cell body.

The exercise is to determine the firing patterns of these two cells for different current injection amplitudes to the excitatory cell, different connection weights (excitatory and inhibitory) and different connection delays.

Do the following:

1. Open **EIosc.nrnzip** to run the simulation in NEURON.
2. Make sure amp is set to 0.8 nA in the IClamp in a window (this provides a constant current injection to the excitatory cell).
3. In the RunControl window and set Tstop to 1010 (if not already set) and press Init & Run – this will run a simulation of stimulating the excitatory cell with the IClamp electrical current for 1010 msecs.
4. The two separate voltage graphs show the voltage response for the excitatory cell soma (top graph) and for the inhibitory cell soma (bottom graph, red trace). For this network configuration you should observe a characteristic oscillating firing pattern in both cells: repetitions of a burst of action potentials followed by a silent period.
5. Set the E->I weight to 0 and rerun to see what happens when the cells are not connected. Now reset the E->I weight to 0.02 before continuing to step 6.
6. This network behaviour is actually very sensitive to the current injection amplitude to the excitatory cell. Try changing amp in the IClamp window by small amounts, keeping within a range of between 0.6 and 1.0 nA. *Note any different firing patterns that emerge.*
7. Now reset the IClamp amp to 0.8. Try changing the E->I weight in the Weight window in a range between 0.01 and 0.08. *Again, note any different firing patterns.*
8. Reset the E->I weight to 0.02. Try changing the I->E weight in a range from 0.004 to 0.02. *Note what happens now.*
9. Reset the I->E weight to 0.005. Try changing the E-> delay between 0 to 4 msecs and *note what happens.*

Question 2: What is the behaviour of this circuit and how do the various parameters (weights, delays etc) affect this behaviour? Can you explain these effects?

3. Excitation-Inhibition Balance in an I&F Neuron

This exercise runs the code behind figure 8.6 of PCNM that explores the firing characteristics of a single integrate-and-fire neuron being driven by a number of excitatory and inhibitory inputs. This was originally explored by Stein (*Biophysical Journal* 5:173-194, 1965). With a large, balanced number of excitatory and inhibitory inputs (NE=300, NI=150) the cell fires irregularly. With a limited number of only excitatory inputs (NE=18, NI=0) the cell fires at the same mean rate, but much more regularly.

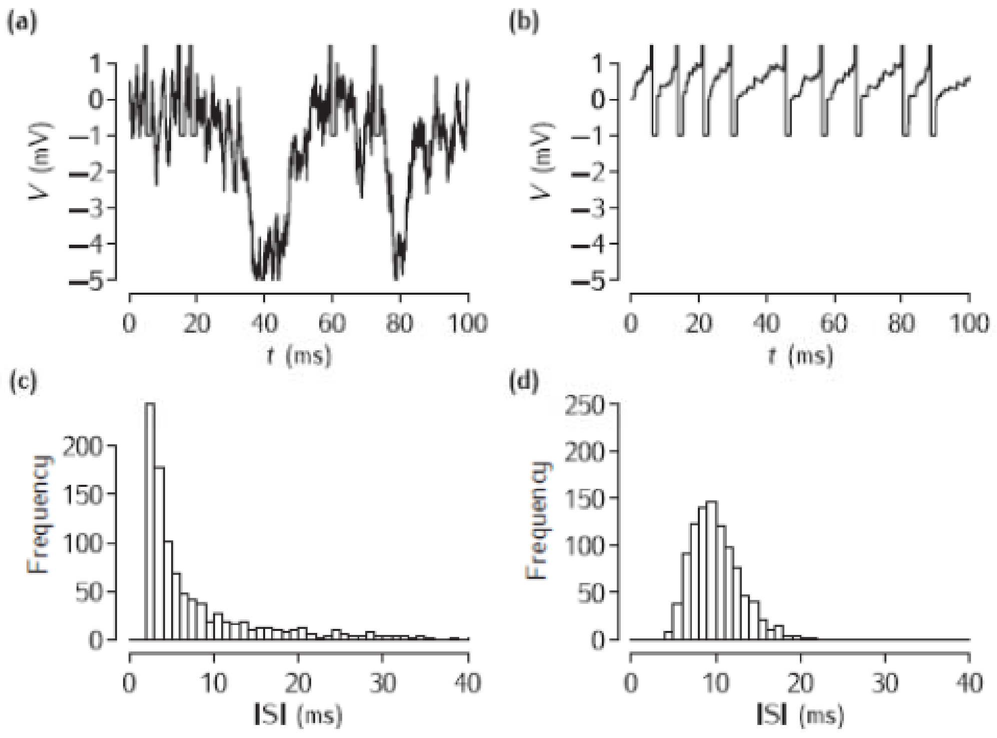


Figure 8.6 (a, c) Large, balanced excitation and inhibition (NE=300, NI=150). (b, d) Small excitation (NE=18, NI=0)

Running the book code:

Open **EIbalance.nrnzip** to run the code associated with figure 8.6 in the book.

To reproduce the data behind panel **(a)** and **(c)**:

1. In the **RunControl** window click on **Init&Run**
2. A trace of the membrane potential appears in the top window (first 100 msecs of simulation only) and at the end of the simulation a histogram of the interspike intervals recorded over the entire simulation (10 secs) appears in the lower window.

To reproduce the data behind panel **(b)** and **(d):**

1. In the **Parameters** window change **NE** to 18 and change **NI** to 0.
2. In the **RunControl** window click on **Init&Run**.

Do the following:

Set the parameters to the large, balanced condition (NE=300, NI=150) and now do:

1. Explore the effect of changing the strength of the excitatory and inhibitory inputs by changing **JE** and **JI** in the **Parameters** window. Make small changes to one or other input strength and note what effect it has on the regularity of firing. For each value of JE that you try, find a value of JI that results in (a) irregular firing, and a value of JI that results in (b) regular firing.
2. Return the strengths to their original values (JE=0.1, JI=-0.2). Now, in the same way, explore changing the number of excitatory and inhibitory inputs.

Question 3: How sensitive is irregular firing to the balance between excitation and inhibition?

4. Excitation-Inhibition Balance in a Network of I&F Neurons

This exercise runs the code behind figure 9.8 of PCNM that explores the firing characteristics of a network of integrate-and-fire neurons, based on the work of Amit & Brunel (*Network: Computation in Neural Systems* 8:373-404, 1997). The network contains randomly connected populations of excitatory and inhibitory neurons, which also receive external random excitatory inputs. In an infinitely large network like this, with balanced excitation and inhibition, the cells should fire essentially randomly and uncorrelated with each other. In finite sized networks, as in the simulation, episodes of strong correlations in firing between groups of neurons appear. These become more evident the smaller the network.

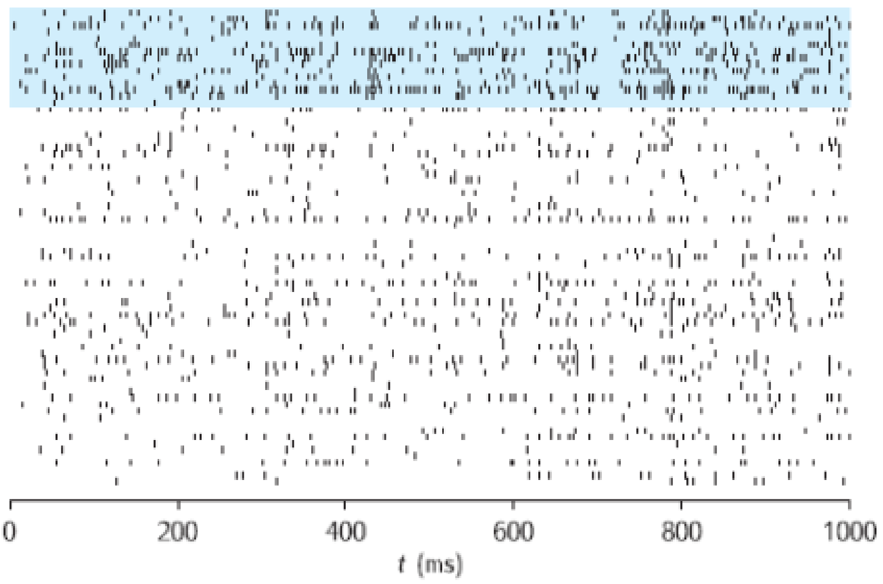


Figure 9.8 Raster plot of activity from 15 inhibitory (shaded at top) and 60 excitatory neurons exhibiting largely irregular firing.

Running the book code:

Open **EIbalnet.nrnzip** to run the code associated with figure 9.8 in the book. This simulation is set up to run a network of half the size (scale=0.5) of the network used to produce the results in the figure.

Click the Run button in the small Control window. It will take several minutes to set up and run the simulation! Eventually the membrane potential traces of 3 example excitatory neurons (ifn[0].M etc) and one inhibitory neuron (ifn[NE].M, blue line) will start to appear in the Graph windows 0 to 3.

What the plots reveal:

At the end of the simulation, the other Graph windows will show traces of different views of the network spiking activity, as follows:

Graph[4]: Raster plot of the excitatory and inhibitory cells' spiking. Each row is one cell, and each mark represents that cell firing an action potential. The yellow background highlights the inhibitory cells. You might need to right-click on this Graph and select **View... -> View = Plot** to reveal the excitatory cells' spiking activity.

Graph[8]: Instantaneous firing rate of the excitatory cells and inhibitory cells (a count of the number of cells that spiked in each population in small time bins: essentially a summation of the raster plot).

Graph[5]: Instantaneous firing rates of excitatory cells (they may not be visible in Graph[8]).

Graph[6] and Graph[7]: Histograms of the average firing rates (over entire simulation period) of the excitatory and inhibitory cells, respectively. *What do these plots reveal about the differences between the excitatory and inhibitory population activities?*

Graph[9]: autocorrelation of spike intervals in excitatory cells. This gives an indication of the irregularity, or otherwise, of each cell's firing pattern.

Do the following:

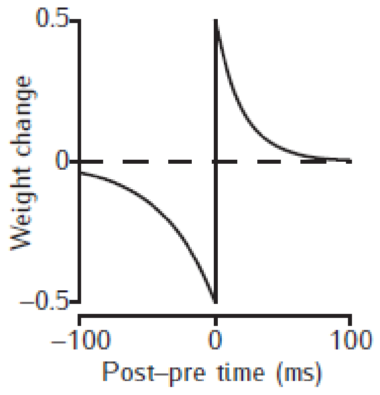
Try several smaller networks i.e. reduce the scale parameter below 0.5. Once a new simulation is finished, you may need to refresh the plot windows by right-clicking and selecting **View... -> View = Plot** to reveal the new data.

Question 5: How do the network firing characteristics change as the size of the network is reduced?

5. STDP in Action

There are two exercises to illustrate the effects of spike-time-dependent plasticity (STDP). They make use of a simple, saturating version of STDP (equation 7.37 in PCNM with leading terms fixed at e_i=e_j=1):

$$\frac{\mathrm{d}w_{ij}}{\mathrm{d}t} = \epsilon_i \epsilon_j \Bigg[(w^{\mathrm{LTP}} - w_{ij}) \sum_k A^{\mathrm{LTP}} \exp(-(t - \tilde{t}^{\mathrm{pre}})/\tau^{\mathrm{LTP}}) \delta(t - t_k^{\mathrm{post}}) - (w_{ij} - w^{\mathrm{LTD}}) \sum_l A^{\mathrm{LTD}} \exp(-(t - \tilde{t}^{\mathrm{post}})/\tau^{\mathrm{LTD}}) \delta(t - t_l^{\mathrm{pre}}) \Bigg].$$



This figure shows schematically the magnitude of the weight change as a function of the time the postsynaptic spike follows a presynaptic input. LTP takes place if the postsynaptic spike occurs after the presynaptic spike, with the magnitude of the weight increase decaying with a time constant of 17 ms (right-hand curve; schematic only). LTD occurs when the presynaptic spike follows the postsynaptic spike, decaying with a time constant of 34 ms (left-hand curve; schematic only).

5a. Phase precession of spike timing

In this exercise, a single neuron receives excitatory input from 10 spike trains through synapses that can undergo STDP. Each spike train consists of regular spikes spaced at 30 msec intervals. However, each spike train is offset by increments of 1 msec, so that the first spike in the 10th spike train occurs 10 msecs after the first spike in the first train. The individual EPSPs from the different spike trains summate, so that initially the cell fires its own action potential near the end of the sequence of spikes from all the inputs. Depending on the STDP parameters, the timing of this postsynaptic spike may evolve over the simulation, relative to the input spike times.

Do the following:

1. Run this simulation by opening **STDPtiming.nrnzip**.
2. Press **Init&Run** to see the effect described above. *Note any change in the timing of the postsynaptic spike relative to the inputs over the course of the simulation*.
3. Plot the weights (press the **Plot weights** button in the **Weights & Delays** window and the other Graph window will be updated; this window may be obscured by other windows, so click on it to bring it to the front) and *note the pattern of weights across the inputs 0 to 9*. In this simulation, all the weights start at 0.012 and can saturate at a maximum of 0.12 (determined by the weight multiplier) and a minimum of 0.0012 (determined by the weight divisor). Only LTP, and not LTD, is possible, with a potentiation rate of 0.05.
4. Try setting the depression rate to 0.2 and rerun the simulation and replot the weights. *Note any difference you see in the voltage trace and the weight distribution.*

5. Try changing both the LTP and LTD rates individually and together to see if the "phase precession" effect on the postsynaptic spike is quicker or slower to emerge. You might need to run the simulation for more than 200msecs.

Question 5a: How and why does the timing of the postsynaptic spike change, and why does a pattern emerge of weights across the different inputs?

5b. Sequence learning

In this exercise, 10 neurons are connected all-to-all by excitatory synapses that can undergo STDP. Each cell also receives a single external input that is strong enough to make it fire its own AP. These external inputs arrive with a delay of 10 msecs from one neuron to the next, so that the cells in the network fire in sequence at 10 msec intervals. This sequence is repeated, with a delay between repetitions. The connections between cells are set to a weak value (0.001) so that they do not cause the receiving cell to fire. Nonetheless, STDP will change the strength of these synapses depending on the spike times of the cells. The aim of the exercise is to see what pattern of connection weights develops over time as the cells spike in sequence.

Do the following:

1. Run this simulation by opening **STDPsequence.nrnzip**.
2. Press **Init&Run** to run a simulation for 1000 msecs.
3. The voltage Graph shows membrane potential traces from the soma of all 10 cells.
4. The Shape window contains a color matrix representation of all the connection weights. Initially they are all the same. Once your simulation has finished, click the Update plot button, then resize the Shape window a little to get it to actually refresh the colours. Each column shows the weights of the connections a neuron makes to all others in the network, working from the bottom up. The hotter the colour, the stronger the connection. *Note any pattern you can see in the weights that have developed across all the neurons.*
5. The initial simulation contains only LTP. Set the depression rate to 0.2 as well and rerun the simulation (note that updating any parameter in the "Weights & Delays" window results in all synaptic weights being returned to the base value). Update the Shape window as above. *Can you see any difference in the pattern of weights now?*
6. Try different LTP (Potentiation) and LTD (Depression) rates.
7. Try also changing the delay between successive inputs. *How do you think this might interact with the time windows of LTP and LTD (see the STDP figure above)?*

Question 5b: What patterns of weights emerge? Can you explain why? And how do the patterns vary with different rates of LTP and LTD and with different delays between spikes?

6. Associative Memory in a Network of Spiking Neurons

This exercise runs the code behind figure 9.10 in PCMN that explores autoassociative recall of stored patterns in an associative memory neural network. The network consists of 100 two-compartment Pinsky-Rinzel neurons connected by all-to-all excitatory and inhibitory connections. Stored patterns consist of a random selection of 10 active neurons from the network population of 100 excitatory neurons. Fifty patterns have been stored in the network by binary Hebbian learning.

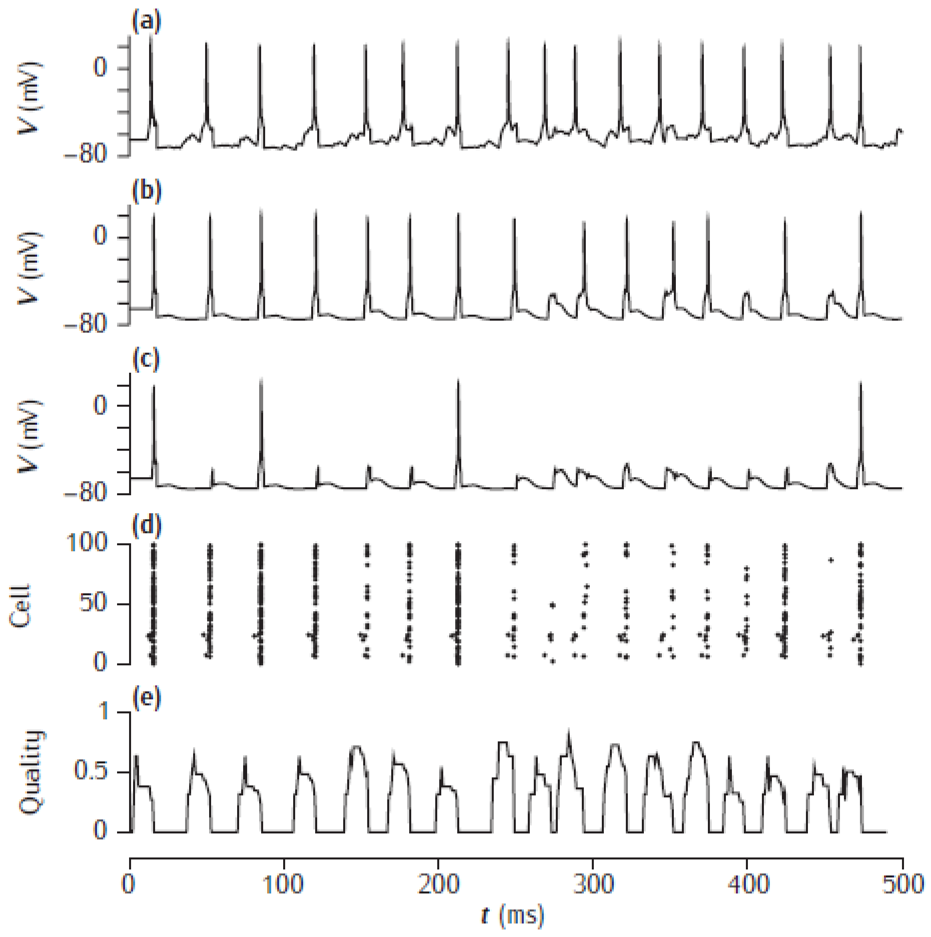


Figure 9.10 (a) Example cue cell. (b) Pattern cell. (c) Non-pattern cell. (d) Raster plot of all cells. (e) Quality of recall.

Running the book code:

Open **AM.nrnzip** to run the code associated with figure 9.10 in the book. This runs a simulation in which 4 of the cells in one of the stored patterns are made active by external 500Hz excitatory input. Hopefully the network activity will largely be the recall of the entire stored pattern. Actually, recall is not perfect: all the pattern cells do fire regularly (eg top two voltage plots), but there is also occasional activity of neurons who do not belong to the pattern (eg third voltage plot). Click **Spike Plot** to see a raster plot of the entire network activity.

Do the following:

1. Try reducing the inhibitory weight by small amounts. You should start to see more spurious (non-pattern) cells firing, but they will tend to fire later than the pattern cells. Additionally, the pattern cells will start to fire in bursts.

2. Return the inhibitory weight to its original value, and try changing the inhibitory delay instead. What happens?
3. Try changing the excitatory weights and delays as well.

Question 7: How sensitive does pattern recall seem to be to these network parameters?

(ADVANCED) Changing the simulation:

You can extract all the code files for this simulation if you put **AM.nrnzip** into suitable "uncompress" software. You can then edit **AAM-SW.hoc** into your text editor to make changes to the simulation (suggested exercises below). To run your new simulation you will need to run **mknrndll** on the folder containing all your files. Then you can run the simulation by double-clicking on **mosinit.hoc**.

Advanced exercise:

Try changing the cue pattern (CPATT) and/or the number of cue cells (CFRAC; the default value of 0.3 results in 4 from 10 cells being selected for the cue) by editing these values in **AAM-SW.hoc** and running the new simulation by double-clicking on **mosinit.hoc**. The default voltage plots may no longer be for pattern and non-pattern cells as before – you can examine the stored patterns by loading **pattsN100S10P50.dat** into your text editor and trying to find indices of pattern cells to plot in your voltage graphs.

Very advanced exercise:

If you want a real challenge, implement inhibition in this network with an explicit population of inhibitory interneurons (rather than have direct inhibitory connections between the excitatory cells). Any simple model spiking cell eg simpcell.hoc or an integrate-and-fire neuron model, would be suitable to use for your inhibitory neurons.