

Spiking Neurons

Computer Exercises using the NEURON Simulator

**Bruce Graham, Computing Science &
Mathematics, Faculty of Natural Sciences,
University of Stirling, U.K.**

URL:

<https://github.com/bpgraham/NEURONexercises>

Email: bruce.graham@stir.ac.uk

Table of Contents

Part 1: Electrophysiology of a Neuron

1. Frequency-Input Current (F-I) Firing Curve of a Neuron
 - 1.1. F-I curve of a basic neuron
 - 1.2. F-I curve types I and II
2. Electrical activity in a CA1 Pyramidal Cell

Part 2: Neural Networks and Synaptic Plasticity

3. Simple Excitation-Inhibition (E-I) Oscillator
4. Excitation-Inhibition Balance
 - 4.1. Single I&F Neuron
 - 4.2. Network of I&F Neurons
5. STDP in Action
 - 5.1. Phase precession of spike timing
 - 5.2. Sequence learning
6. Associative Memory in a Network of Spiking Neurons

Overview

These exercises make use of the **NEURON** simulator (neuron.yale.edu). It is not necessary to be familiar with NEURON before attempting these exercises, but previous experience will allow you to progress faster and to try some of the optional, advanced exercises.

The code for these exercises is written in **hoc** code with graphical user interfaces based on **Interviews**. These are the “traditional” tools of NEURON, but it is now also possible to write such simulations using Python and make use of any graphical interface tools available to Python, such as Matplotlib for plotting results. The NEURON documentation available via your NEURON installation will be very useful if you want to learn how to create simulations with hoc or Python for yourself.

A number of these 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.

The NEURON Model Code

Code files

The code for each of the exercises is stored in an appropriately named folder, as detailed in the instructions for each exercise. In each case there is one or more “.hoc” files containing the code that defines the simulation. These may be accompanied by “.ses” files that contain user interface (GUI) definitions and “.mod” files that define various model components, such as ion channel models.

Running the code

Linux

1. In a Linux terminal window, change to the directory (folder) containing the desired HOC file (command “cd”).
2. If the folder contains “.mod” files, first rebuild NEURON to incorporate these model components by typing: “**nrnivmodl**”.
3. Run NEURON by typing “**nrngui filename.hoc**” eg “nrngui Flcurve.hoc”.

Windows

1. In Windows Explorer, change to the folder containing the desired **hoc** file.
2. If the folder contains “.mod” files, first rebuild NEURON to incorporate these model components by running “**mknrndll**” on this folder (you will find this command in the NEURON Start group).
3. Run NEURON by double-clicking the relevant “.hoc” file eg “Flcurve.hoc”.

Part 1: Electrophysiology of a Neuron

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

In the following two exercises you will explore how the firing rate of a neuron changes as a function of injected current in the cell body. This corresponds to a typical electrophysiological experiment that is used to in part characterise the properties of a neuron. The code for the exercises is in the **Fcurve** folder.

Exercise 1.1: F-I curve of a simple neuron (Fcurve.hoc)

A simple compartmental model neuron is stimulated by a constant current injection to the cell body. The cell body 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 (“segments” in NEURON terminology), each of which contains only leak ion channels. The predefined simulation code is available in **Fcurve.hoc** in the **Fcurve** folder.

The exercise is to determine the firing rate response of this neuron to different magnitudes of current injection (which approximates a constant stream of EPSPs due to input from other neurons, which are not modelled). This is the **F-I curve** of our neuron.

Do the following:

1. Run **Fcurve.hoc** in NEURON (as per Linux or Windows instructions, above).
2. Find the PointProcessManager window containing APCount[0]. Click Show and select Parameters to see the data associated with this point process, which will count the number of action potentials the neuron fires.
3. In the IClamp[0] window (which determines the injected current) set the amp to 0.1 nA. (**NOTE:** you must press the return key after entering the number for NEURON to register this change!)
4. Press Init & Run in RunControl – this will run a simulation of stimulating the cell with the IClamp electrical current for 100msecs. The membrane voltage in the cell body over this time is plotted in the voltage Graph[0] window.
5. Note the action potential count (n) in the PointProcessManager window (it may be zero!).
6. Increase the IClamp current amplitude (amp) by a small amount eg 0.1 nA, and repeat steps 4 and 5. Keep incrementing amp by, say 0.1 each time, then by larger amounts when you get bored! There is no need to go beyond an amp of 3.0.

Question 1.1: What is the smallest current amp that causes the cell to fire repeated action potentials and how many does it fire in 100msecs? Ans:

7. Now to plot the complete F-I curve, just press **Plot** in the Grapher window to run a number of simulations automatically. You can alter the number of current steps (Steps) and the length of simulation (Tstop in RunControl) to get a smoother curve, if you wish. To see the curve clearly, right-click on the plot of APCount[0].n and select View...->View=Plot.

Question 1.2: What type (I or II) of F-I curve does this neuron have? (Looking at the figure below in Exercise 1.2 will help with answering this.) **Ans:**

- When you have finished, close this simulation by selecting **File->Quit** in the NEURON Main Menu window. (NOTE: any time you run a NEURON simulation it is important that you close it properly before running it again, otherwise you will end up with multiple instances running at once.)

Exercise 1.2: Type I and Type II F-I Curves

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 the example code from the PCMN book.

Running the code:

To run the code associated with figure 5.9 in PCNM, open **Ftypes.hoc** in NEURON (after first rebuilding NEURON in the Fcurve folder to compile the .mod files). This demonstrates Type I and Type II firing in a neuron, controlled by the presence or absence of the potassium A-type current.

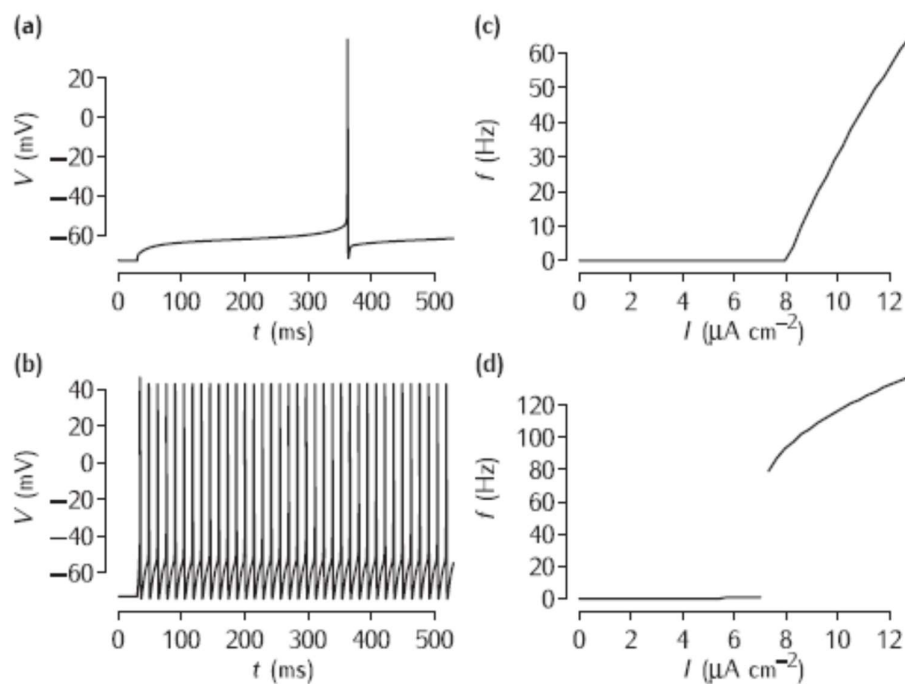


Figure 5.9: (a) First spike in a Type I neuron. (b) Start of spiking in a Type II neuron. (c) f-I curve for Type I. (d) f-I curve for Type II.

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

Panel (a)

- In the top **Parameters** window, click on the checkbox next to **Type I parameters**
- Click on **Set Type I threshold current**
- 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. The plot windows below this graph show sodium, potassium and leak conductances and current.

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.
4. Once the simulations have run, to make sure you can see these results, right-click in this graph area and select **View... -> View = Plot**

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 again the firing frequency will be plotted against current in the **Grapher** window.
4. Once the simulations have run, to make sure you can see these results, right-click in this graph area and select **View... -> View = Plot**

Question 1.3: What cell parameters are different between the Type I and Type II examples?

Ans:

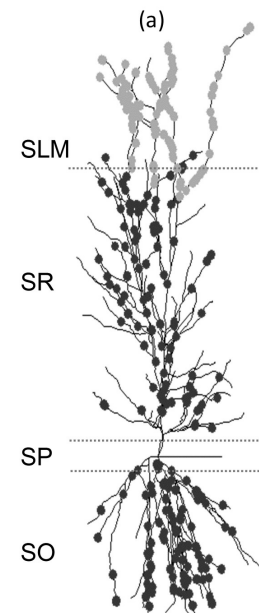
Question 1.4 (supplementary question for later research): Can you explain why the difference in firing characteristics arises?

Ans:

2. Electrical Activity in a Hippocampal CA1 Pyramidal Cell Model

In these exercises you will investigate electrical activity due to current injection and synaptic input in a detailed compartmental model of a hippocampal CA1 pyramidal cell. You will investigate the ability of synaptic input to generate simple spiking, bursting outputs and calcium spikes in the dendrites. You will also study the backpropagation of sodium spikes from the soma into the dendrites, a feature that is important for spike-timing-dependent plasticity (STDP).

The cell model is based on the morphology of an actual PC from a rat (see figure a), with membrane properties extended from the model of *Migliore et al, J Neurophys 94:4145-4155, 2005*. The morphology is divided into 337 electrical compartments (not counting spines). Spatially distributed ion channels consist of the following families: fast sodium (Na), delayed rectifier potassium (K), A-type K, h-current, HVA (putatively R-type) calcium (Ca) and Ca-activated, mAHP K. Na and KDR are distributed throughout the cell, but Na has a higher density in the axon and a lower density in the dendrites; it also has slower recovery from inactivation in the dendrites. K_A and h channels both increase in density with distance from the soma, with a saturating density beyond 350 μ m (the SLM region). Voltage-gated calcium channels (VGCCs: HVA R-type Ca) are distributed throughout the dendrites and spines with a fixed density. This is a restricted set of all the ion channels identified in CA1 PC membrane, but is sufficient to capture key features of the electrical excitability of the cell. Two distinct features are the ease of spread of signals throughout the dendrites (the electrical compactness) and the ability to generate nonlinear, threshold-gated signals, namely sodium and calcium spikes in the dendrites. Both features can be under the control of inhibition and neuromodulation in a single neuron so that different cell excitability may be obtained in different behavioural conditions. The model has been used in *Graham, Saudargiene & Cobb (2014) Neural Computation 26(10):2194-2222*, to study calcium transients in spine heads due to patterns of synaptic input.



Exercise 2.1: Back-propagating action potentials

In this exercise you will use electrical current injection in the cell body (soma) to initiate an action potential (AP) and then measure the AP amplitude as it travels back into the dendrites (back-propagation; note that a small length of axon is modelled along which the AP forward propagates to potentially reach synapses on other neurons, but we will not look at this).

1. Run the simulation by opening **run_PC.hoc** in the **CA1PC** folder in NEURON (**first rebuild NEURON in this folder**). A multitude of windows will appear. Three are “Shape” windows that display the cell morphology, with one of these showing example synapses with blue, red and yellow coloured dots and another with 3 red dots showing the recording locations of the membrane voltage plots (Graph[0], Graph[1] and Graph[2]). These Shape windows might be hidden by other windows so click on them to bring to the front and then move them to somewhere convenient.
2. We will only use a current injection input, so firstly set “Number” of synapses to 0 (instead of 500) for each dendritic layer (Synapses GUI: SR, SO, SLM; note that you must press the

Return key after setting the number to 0 for this change to be applied) to prevent synaptic input.

3. To generate a somatic AP, find the PointProcessManager window that contains IClamp[0]. If necessary, click Show and select Parameters. Then set IClamp[0] amp to 1.5 nA, to inject a small current into the soma after a delay of 50 msec.
4. Right-click in the third "Shape plot" window (the one with no coloured dots) and select "Shape plot" (the very last menu option; you may need to move the window to actually see it). This will provide a visualisation of the membrane voltage throughout the entire cell when we run a simulation.
5. Click "Init & Run" in RunControl to run a simulation for 200 msec. Watch the "Shape plot" window to see a colour coding of membrane voltage (you can make this window larger to see it more clearly). Voltage traces will also appear in the three Graph windows.
6. In the Plots GUI, click "Dendrites" under "Max voltage plots" to see the peak membrane voltage in the dendrites as a function of distance from the soma.
7. Reduce the density of K_A channels in the dendrites to 0.01 mS/cm^2 in the "Ion Channels" GUI, rerun the simulation and replot the peak dendritic voltage to see what happens now.

Question 2.1: What was the effect of K_A on the ability of a somatic AP to back-propagate into the dendrites?

Ans:

Exercise 2.2: Synaptic input to SR

CA1 pyramidal cells get excitatory synaptic inputs from hippocampal CA3 pyramidal cells to their apical and basal dendrites in regions stratum radiatum (SR) and stratum oriens (SO). Here we will study the effects of such synaptic input to the SR apical dendrites. Do the following:

1. Set IClamp[0] amp to 0 to remove the somatic current injection; and also return the density of K_A channels in the dendrites to its original value of 0.03 mS/cm^2 .
2. Set the number of synaptic inputs to stratum radiatum (SR) to 10. Inputs to these synapses activate postsynaptic AMPA and NMDA receptors which both contribute (with different time courses) to generating an excitatory postsynaptic current (EPSC), which depolarises the pyramidal cell and may lead to postsynaptic spike generation. Initially, the selected number of synapses will all receive one presynaptic stimulus at exactly 50 msec.
3. Click "Init & Run" to run the simulation. Watch the third Shape window, which you set to be a "Shape plot" in exercise 2.1, to see the voltage changes throughout the cell. Then study the voltage traces in the three Graph windows: the top trace is from the cell body (soma), the middle trace is from SR, and the bottom trace is from stratum lacunosum-moleculare (SLM).
4. You will see that 10 active synapses do not have a large effect on the pyramidal cell in this simulation. However, measure the amplitude of the EPSP in SR (Graph[1]) and the soma (Graph[0]) and **note these values in the table below**: you can do this by clicking-and-holding with the left mouse button on the voltage traces in these windows; the cursor will change to a cross and will follow the voltage trace as you move the mouse; the x and y coordinates are shown in the top window bar and indicate the time and voltage, respectively; put the cursor

on the peak voltage response, read off the y value and subtract the resting value of -65 mV from this to get the amplitude.

Synapses:	10		50		100	
	Soma	SR	Soma	SR	Soma	SR
High K_A (0.03):						
Low K_A (0.01):						

5. Increase the number of activated SR synapses to 50 and rerun the simulation. Again, watch the “Shape plot” and when the simulation has finished, note the voltage traces in the Graph plots and extract and record the EPSC amplitude in the soma and in SR in the table.
6. Repeat this for 100 synapses.

Question 2.2: Approximately, what is the relationship between the number of stimulated synapses and the EPSC amplitude?

Ans:

7. Now run the simulation with 150 activated synapses. This summed input is large enough to generate an action potential in the pyramidal cell.

Question 2.3: Where in the cell is this action potential first generated?

Ans:

8. Repeat with 200 synapses.

Question 2.4: Now where in the cell is this action potential first generated?

Ans:

9. Repeat with 400 synapses.

Question 2.5: What is the cell’s response now?

Ans:

10. Set the density of VGCCs (gCa-R) in the dendrites to 0 mS/cm² and rerun the simulation with 400 synapses.

Question 2.6: What characteristics of the VGCCs can you deduce from these simulations?

Ans:

11. Return the density of VGCCs (gCa-R) in the dendrites to 0.03 mS/cm² and now reduce the density of K_A channels in the dendrites to 0.01 mS/cm². Set the number of SR synapses to 10 and run the simulation, noting the EPSC amplitudes in the table.
12. Repeat this with 50 active synapses, again noting the EPSC amplitudes in the table.

Question 2.7: How do these EPSC amplitudes compare with when the K_A density was high?

Ans:

13. Now run with 100 active synapses and note what happens.

Question 2.8: What happens now? Design and run a simulation to test your conclusion.

Ans:

Exercise 2.3 (supplementary): Synaptic input to SLM

The very distal apical dendrites (the SLM region) of this pyramidal cell get excitatory inputs from the entorhinal cortex (EC). Given the distance of these inputs from the cell body, we might suspect that it will be difficult for them to cause a spiking output from this cell. On the basis of what you learnt about the SR inputs in exercise 2.2, design and carry out a set of simulations to determine under what cell configurations (in terms of the cell's membrane ion channel properties) inputs to SLM may cause this cell to fire an output action potential (ie one generated at the soma).

Question 2.9: What are your conclusions?

Ans:

Further exercises (supplementary):

There are many more things about synaptic integration and cell output you can test with this simulation. Feel free to carry out further simulations to explore:

- Inputs from CA3 to stratum oriens (SO)
- Multiple stimuli to each synapse at different frequencies (ie inter-stimulus intervals)
- Asynchronous inputs to groups of synapses

Part 2: Neural Networks and Synaptic Plasticity

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

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

Each neuron (defined in **simpcell.hoc**) 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 predefined network simulation available in **Elosc.hoc** in the **Eloscillator** folder. 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 the file **Elosc.hoc** (in the **Eloscillator** folder) in NEURON.
2. A small window will show the connection weight values between the cells (you might have to move other windows eg RunControl to see this window). Make sure the initial values are E->I weight: 0.02; E-> delay 2; I->E weight: 0.005; I->delay 2.
3. In the IClamp[0] window make sure amp is set to 0.8 nA.
4. In the RunControl window make sure Tstop is set to 1000. Now press "Init & Run" to run a simulation that involves stimulating the excitatory cell with the IClamp electrical current continuously for 1000msecs.
5. The two voltage plots show the spiking activity in the E and I cells, respectively. For this network configuration you should observe a characteristic oscillating firing pattern in both cells: bursts of action potential firing, followed by silent periods.
6. 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.
7. 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.

Question 3.1: What different firing patterns emerge?

Ans:

8. 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.

Question 3.2: Again, what different firing patterns emerge?

Ans:

9. Reset the E->I weight to 0.02. Try changing the I->E weight in a range from 0.004 to 0.02.

Question 3.3: What happens now to the firing patterns?

Ans:

10. Reset the I->E weight to 0.005. Try changing the E->I delay between 0 to 4 msec.

Question 3.4: Again, what happens to the firing patterns?

Ans:

Question 3.5 (summary of results): What is the general behaviour of this circuit and how do the various parameters (weights, delays etc) affect this behaviour?

Ans:

4. Excitation-Inhibition Balance

The balance between excitatory and inhibitory inputs impinging on neurons strongly dictates their firing patterns and can lead to asynchronous firing in networks of neurons. In two exercises we explore the impact of such excitation-inhibition balance. The code is in the **Elbalance** folder.

Exercise 4.1: E-I balance in a single 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 small excitation only (NE=18, NI=0) the cell fires at the same mean rate, but much more regularly.

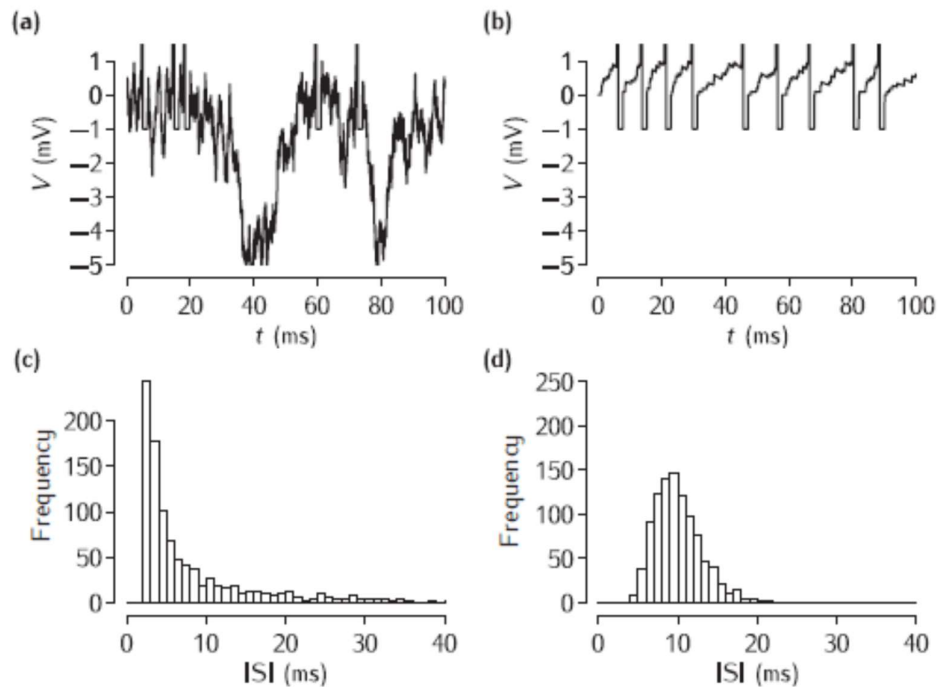


Figure 8.6 (a, c) Large, balanced excitation and inhibition (NE=300, NI=150). (b, d) Small excitation (NE=18, NI=0)

Running the code:

Run the code associated with figure 8.6 in PCNM by opening **Elbalance.hoc** in the **Elbalance** folder, in NEURON.

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 and at the end of the simulation the histogram 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 do the following:

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

Ans:

Exercise 4.2: E-I 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 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.

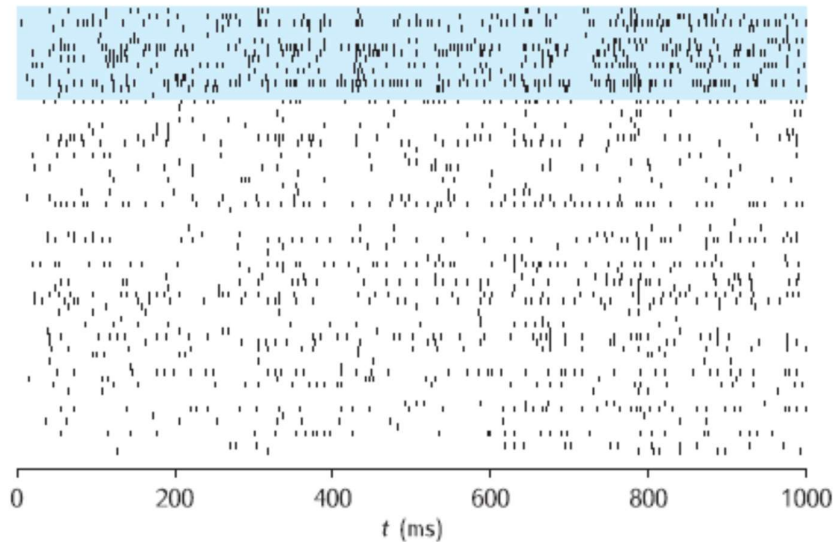


Figure 9.8 Raster plot of activity from 15 inhibitory (shaded at top) and 60 excitatory neurons exhibiting largely irregular firing.

Running the book code:

Run the code associated with figure 9.8 in PCNM by opening **Elbalnet.hoc** in NEURON (after **rebuilding NEURON first**). 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. It may still take several minutes to set up and run on a decent PC.

Do the following:

1. Click Run in the small control window to run this half-scale model. While it is running, voltage plots from several example neurons will appear (excitatory in black, inhibitory in blue). Once the simulation has finished, the other output windows will be populated with various statistical measures of the neuronal firing rates. NOTE: you might need to resize Graph[4] slightly to see the full spike raster plot (as illustrated in Fig. 9.8 above). These firing rates should be largely irregular in this configuration.
2. Try several smaller networks i.e. reduce the scale parameter below 0.5.

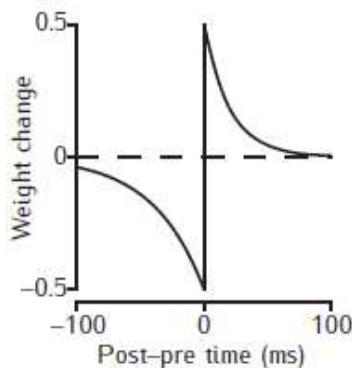
Question 4.2: How do the network firing characteristics change as the size of the network is reduced?

Ans:

5. Spike-Time-Dependent Plasticity (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 implemented in NEURON's **NMODL** in the file **stdpB.mod** in the STDP folder (equation 7.37 in PCNM with leading terms fixed at $e_i=e_j=1$):

$$\frac{dw_{ij}}{dt} = \epsilon_i \epsilon_j \left[(w_{ij}^{\text{LTP}} - w_{ij}) \sum_k A^{\text{LTP}} \exp(-(t - \hat{t}^{\text{pre}})/\tau^{\text{LTP}}) \delta(t - t_k^{\text{post}}) - (w_{ij} - w_{ij}^{\text{LTD}}) \sum_l A^{\text{LTD}} \exp(-(t - \hat{t}^{\text{post}})/\tau^{\text{LTD}}) \delta(t - t_l^{\text{pre}}) \right].$$



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. LTD occurs when the presynaptic spike follows the postsynaptic spike, decaying with a time constant of 34 ms.

Exercise 5.1: 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 msec 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 using **STDptiming.hoc** in the **STDP** folder.
2. Make sure **Tstop** is set to 200msecs in the **Run Control** window.
3. Press **Init&Run** to see the effect described above (you might need to enlarge the somatic voltage graph (Graph[2]) to see the EPSPs clearly).

Question 5.1: Do you see any change in the timing of the postsynaptic spike relative to the inputs over the course of the simulation?

Ans:

4. Press the **Plot weights** button in the **Weights & Delays** window to plot the weight values (y-axis) versus input number (x-axis) in the other Graph window. **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.

Question 5.2: What is the pattern of weights across the inputs 0 to 9?

Ans:

5. Try setting the depression rate to 0.2 and rerunning the simulation and replotting the weights.

Question 5.3: What differences can you see in the voltage trace and the weight distribution from previously?

Ans:

6. 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 5.4 (summary): How and why does the timing of the postsynaptic spike change, and why does a pattern emerge of weights across the different inputs?

Ans:

Exercise 5.2: 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 at an interval of 10 msec, so that the cells in the network fire in sequence at 10 msec intervals. 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 using **STDPnetseq.hoc**.
2. In the **Run Control** window, make sure **Tstop** is set to 1000 msec.
3. The voltage graph window (Graph[1]) will plot the voltages from each of the 10 cells in a different colour on the same plot.
4. The initial parameter values in the **Weights & Delays** window should be Input delay: 10; Base weight: 0.001; Potentiation: 0.2; Depression: 0.
5. Press **Init&Run** to run the simulation for 1000 msec.
6. 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 in the **Weights & Delays** window, 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.

Question 5.5: Can you see a pattern in the weights that have developed across the neurons?

Ans:

7. The initial simulation contains only LTP. Set the depression rate (Depression in Weights & Delays window) to 0.2 as well and rerun the simulation. Update the Shape window as above.

Question 5.6: Can you see any difference in the pattern of weights now?

Ans:

8. Try different LTP (Potentiation) and LTD (Depression) rates.
9. Try also changing the delay between successive inputs.

Question 5.7: How do you think this delay might interact with the time windows of LTP and LTD (see the STDP figure above)?

Ans:

Question 5.8 (summary): 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?

Ans:

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.

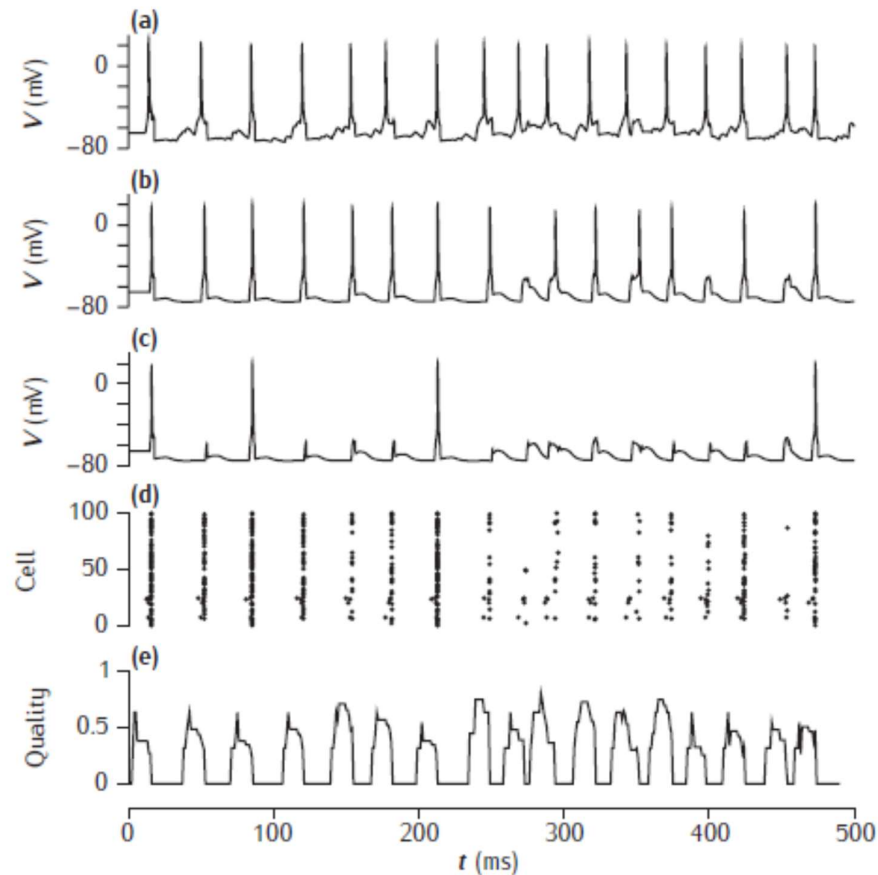


Figure 9.10 (a) Example cue cell. (b) Pattern cell. (c) Non-pattern cell. (d) Raster plot of all cells. (e) Quality of recall.

Running the code:

To run the code associated with figure 9.10 in the book, rebuild NEURON in the **AssocMem** folder and then run **AssocMem.hoc** in NEURON. 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, but there is also occasional activity of neurons who do not belong to the pattern.

Do the following:

1. Click Init & Run in the RunControl window to run this “pattern recall” simulation. The voltage graphs show the spiking activity of three example cells.

Question 6.1: Which of these 3 cells do you think belong to the stored pattern and which do not?

Ans:

2. Click **Spike Plot** in the **Control** window to see a raster plot of the entire network activity.
3. In the **Control** window, 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.
4. Return the inhibitory weight to its original value and try changing the inhibitory delay instead. What happens?
5. Try changing the excitatory weights and delays as well.

Question 6.2: How sensitive does pattern recall seem to be to these network parameters?

Ans: