Lecture 23.1 Exercises

This tutorial will show the user how multiscale modeling can be used to understand connections between pharmacology and network dynamics.

23.1.1 Running a simulation

We use here a publicly available network model that is discussed in detail in the schizophrenia section of Chapter 23.

1. You will need Neuron to be installed on your computer. You can download and install it from here:

   http://neuron.yale.edu/neuron/download

2. You will need to have matplotlib and numpy python libraries installed on your computer.

3. We will be using the computer model at this url:

   http://modeldb.yale.edu/139421

4. Download the zip file of the model, and unzip it.

5. cd to the directory which contains the unzipped folder

   then:

   nrnivmodl at command line or follow instructions for your platform.

   This will compile the .mod files into c files

6. To run the simulation:
   nrngui -python mosinit.py

   or click on mosinit.py to run the simulation

The simulation will use multiple threads, so it will make use of multiple cores on your machine

It took around 6 min to run on my laptop (Acer Intel-Core i3)

When the simulation ends, 2 graph windows will be shown. Note that the windows might be on top of each other, and you might need to move the first one aside to see the other one.

Description of the resultant plots could be found on the model page:

http://modeldb.yale.edu/139421

23.1.2 Single cells

Now that the full network functionality was shown, we are planning to explore single cells of the network.

Exit the simulation if you have not already done that

Save the following files to the directory which contains the unzipped model files

singleCells.py

plotVoltages.py

We will be using it to manipulate single cells.

While at the directory containing the unzipped model files:

python -i singleCells.py

(or ipython -i singleCells.py if you will be using ipython)

Then at the python prompt:

```
from plotVoltages import *

pyrcell = PyrAdr(0,0,0,0)

h.tstop = 4000

h.v_init = -65 # set initial voltage to -65 mV


# use the current clamp inserted into the soma of the pyramidal cell
```

```
pyrcell.somaInj.amp = 50e-3 # amplitude of current in nA

pyrcell.somaInj.dur = 500 # duration of current injection in ms

pyrcell.somaInj.delay = 1000 # delay in ms, after which current injection starts


# setting vectors to record voltage from soma, tip of apical dendrite and basal dendrite  of

# pyramidal cell

somaVector, apicaldendVector, basaldendVector, timeVector = [h.Vector() for i in range(4)]


somaVector.record(pyrcell.soma(0.5)._ref_v)

apicaldendVector.record(pyrcell.Adend3(0.5)._ref_v)

basaldendVector.record(pyrcell.Bdend(0.5)._ref_v)

# setting vector to record time (to aid in plotting voltage later on)

timeVector.record(h._ref_t)


h.run()

plotVoltages()
```

You will notice that there are no spikes in the voltage plot. Note that the y-axis extends from -68 to -58 mV, so it hasn't fully depolarized. Now try running wtih current amplitude of 100e-3 nA, rerun and plot again (You don't have to exit the simulation to rerun).

After increasing the amount of current injected, you can see spikes

The figures will look like those:

voltageplot_current_noSpiking.png

voltageplot_current_Spiking.png

Now we will calculate the input resistance of the cell. Input resistance is a measure of how much the voltage of a membrane would change in response to injecting an amount of current. It is one of the

electrophysiological parameters which experimentalists measure. The way it is usually measured is by injecting a hyperpolarizing current of known amplitude, and then calculate the voltage change after the voltage has reached a steady state level. So, we need to reach 2 voltage steady states: 1) before injecting the current; 2) during injecting the current.

We will use the same code which we used before. The only difference is that we will inject a hyperpolarizing current (so a "negative" current) into the soma.

```
pyrcell.somaInj.amp = -50e-3
```

Start by injecting the current for 2000, 2500 and 3000 ms. Run and plot after each current amplitude. Notice which one allows you to get a voltage steady while the current is being injected.

For current duration of 2500ms, voltage change is -66 mV to -71 mV (5 mV change) in response to 0.05 nA current injection, so input resistance = voltage change / current injected = 5 mV / 0.05 nA = 100 MOhm

23.1.3 OLM cell – pyramidal cell interaction

After we explored the behavior of a single pyramidal cell, we will explore how an OLMcell inhibit a pyramidal cell.

Save the following file to the directory which contains the unzipped model files

```
plotVoltageCells.py
```

While at the directory containing the unzipped model files:

```
python -i singleCells.py

# (or again: if you will be using ipython)

ipython -i singleCells.py
```

Then at the python prompt:

```
from plotVoltageCells import *

pyrcell = PyrAdr(0,0,0,0)
```

olmcell = Ow(10,10,10,1)

olmToPyr_netcon = h.NetCon(olmcell.soma(0.5)._ref_v, pyrcell.Adend2GABAs.syn, 2, 4 * 3 * 6.0e-3, 20, sec = olmcell.soma)

h.NetCon is an object which detects the spike in a presynaptic cell (olmcell), and then delivers a response to a synapse in the postsynaptic cell (pyrcell), after a particular delay (to model the release of neurotransmitter from the presynaptic cell, its diffusion across the synaptic clift and binding to the postsynaptic receptors).

h.tstop = 2000

h.v_init = -65 # set initial voltage to -65 mV

At baseline, OLM cell is spiking spontaneously - so we will inject a hyperpolarizing current to inhibit its firing for the first 500 ms of the simulation

olmcell.somaInj.amp = -50e-3

olmcell.somaInj.dur = 500

olmcell.somaInj.delay = 0

setting vectors to record from both somas

pyrsomaVector, olmsomaVector, timeVector = [h.Vector() for i in range(3)]

pyrsomaVector.record(pyrcell.soma(0.5)._ref_v)

olmsomaVector.record(olmcell.soma(0.5)._ref_v)

setting vector to record time (to aid in plotting voltage later on)

timeVector.record(h._ref_t)

h.run()

plotVoltageCells()

The plot will look like this:

olmToPyr_voltage.png

23.1.4 Whole network simulation

Now, we will go back to the whole network simulation. We will run the simulation as we did in the first lesson.

We will plot the voltages from individual cells

Save the following file to the directory which contains the unzipped model files

indivCellVoltage.py

While at the directory containing the unzipped model files:

python -i mosinit.py

(or ipython -i mosinit.py if you are using ipython)

Will import python libraries for plotting and data analysis, at the python prompt:

from matplotlib import pyplot as mp

from indivCellVoltage import *

mp.ion() # allow for interactive plotting

import numpy as np

indivfig, indivax = mp.subplots(1,1)

indivCellVoltage('pyr',10, indivax)

networkPyrCell_10.png

Try to plot olm cell number 2 and basket cell number 159.