NEURON Scripting Exercises

Exercise 1:

Given a `list` of sections:

```
from neuron import h, gui

secs = [
    h.Section(name='soma'),
    h.Section(name='dend'),
    h.Section(name='axon')
]
```

and given some data

```
data = {
    "axon": {"L": 100, "diam": 1},
    "soma": {"L": 10, "diam": 10},
    "dend": {"L": 50, "diam": 3}
}
```

Write a script that uses the data to set the `L` and `diam` parameters for each section. Set the number of segments such that there is (1) an odd number of segments and (2) each segment is as close to 5 microns long as possible without going over. Connect all sections not named `soma` to the center of the soma.

Use a `h.PlotShape()` to check your work. Store the object as, e.g. `ps`. Then use `ps.show(0)` to display the diameters.

Bonus challenge: Move the data into a JSON file and use Python's `json` module to read it from the file.

Exercise 2:

Simulate and visualize a propagating action potential on a long cable with Hodgkin-Huxley dynamics triggered by a current clamp.

Things to consider:

What is an appropriate diameter, length of cable, and current injection amplitude?

How can you best illustrate a propagating action potential in a single figure?

What discretization should you use?

Exercise 3:

Continuing with the previous exercise, record and plot the membrane potential, sodium current, and potassium current at the center of the cable as a function of time.

Hint: Use the `Vector` class' `record` method.

Things to consider:

How does the relationship between the sodium and potassium currents differ in a Hodgkin-Huxley simulation from the relationship in mammalian cells?

Exercise 4:

Construct two Y-shaped cells, where each branch is 2 microns in diameter and 100 microns long. Add Hodgkin-Huxley channels with the default parameters. Inject current into one cell and ensure that you are able to generate a propagating action potential in that cell while the other one remains at rest.

Connect the two cells with a gap junction (see `halfgap.mod`) at a location of your choosing. A 3 MΩ resistance as in `halfgap.mod` should allow an action potential in one cell to initiate an action potential in the other. Verify this. What is the strongest resistance that still allows action potential initiation in the second cell?

Bonus challenge: Make each cell an instance of a Python `class`.

Bonus challenge: Modify the `class` to allow repositioning (moving and/or rotating) the cells.

Exercise 5:

Download the model from http://modeldb.yale.edu/126814 Create a Python script that loads the model, injects current into the center of the soma from t=2 to t=4 ms sufficient to generate an

action potential, and records and plots membrane potential, sodium current, and potassium current as functions of time from t=0 to t=10 ms.

Hint: `h.load_file('mosinit.hoc')`

Hint: This model uses the variable time step solver. Be sure to record t or switch to a fixed step solver with `h.cvode_active(0)`.

Bonus challenge: Export the recorded data to a CSV file and then open and plot it in Excel, MATLAB, or a similar tool of your choice.

Bonus challenge: How does the plotted data change as `h.celsius` is varied?

Exercise 6:

NEURON's `ExpSyn` mechanism generates synaptic currents of the form

$$i = g(v - E_{syn})$$

where

$$g' = -g/\tau$$

What is the role of E_{syn}? How does it change for an excitatory vs an inhibitory synapse? (Note, this is the parameter `e` in the code.)

Construct two single compartment neurons with Hodgkin-Huxley dynamics, one of which receives a current pulse at 2 ms, another which receives a current pulse at 10 ms. Ensure that both cells fire action potentials after the input.

Now, using `NetCon` and `ExpSyn`, construct an inhibitory synapse between the two with the cell that fires later as the post-synaptic cell. Choose a delay and strength such that the post-synaptic cell is inhibited from firing. Plot the membrane potentials vs time.

halfgap.mod

```
NEURON {
  POINT_PROCESS HalfGap
  POINTER vgap
  RANGE r, i
  ELECTRODE_CURRENT i
}

PARAMETER {r = 3 (megohm)}

ASSIGNED {
  v (millivolt)
  vgap (millivolt)
  i (nanoamp)
}

BREAKPOINT { i = (vgap - v) / r }
```