**Lab 2: NEURON Simulations**

# Introduction

This lab introduces the use of NEURON simulation software to model neuron activity. We will start by creating some simple single-compartment models, examine the effects of modifying model parameters, replicate results from a widely used neuron model, and finish by building a simplified neuron model of our own. This lab will provide you with a primer on how to use NEURON software and its potential applications.

# Software

This lab must be completed using NEURON. Use of Notepad++ (available on CAEN machines) is recommended for editing NEURON scripts. In Notepad++, select Language🡪C🡪C++ for rich text features compatible with NEURON syntax.

# Part 0) Set up NEURON and associated files

After getting NEURON onto the computer, the software will need a bit of setting up. The instructions below were written specifically for computers in the LBME CAEN lab, but you should be able to use NEURON on any CAEN computer now (including CAEN remote access).

1. Download all files in the Lab 2 folder into a directory on your computer, somewhere on the C drive. (The Downloads folder is a good choice, although you’ll have to **remember to save any work you do there to either your user folder or email before you end your session**. The Downloads folder is not synced across CAEN computers.) Choosing your user folder or a cloud folder will cause issues for NEURON.
2. Load in NEURON from the CAEN apps (same as loading Matlab). You may also want to load in Notepad++ too. Navigate to C:\VApps\NEURON\7.4\lib\hoc and open mknrndll.hoc using NEURON (you may need to Launch NEURON from the Cloudpaging Player first). When prompted to choose a directory, select the folder with the Lab 2 files. The command line should indicate that nrnmech.dll compiled successfully. This will create a handful of new files in your folder that NEURON will later refer to for running simulations.

# Part 1) Build a single-compartment neuron model

We’ll start by walking through a short tutorial on building a single-compartment model.

1. Open a blank text file to write our script. (I recommend Notepad++ on Windows.) S**ave it with the file extension .hoc in your Lab 2 folder (not .txt), so that NEURON recognizes it.**
2. To create a single-compartment neuron model, write the following code:

create soma // Make a soma

access soma // Set soma as the default object for editing

nseg = 1 // Soma modeled using just one segment

pt3dadd(0,0,0,18.8)

// Set start of soma at coordinate (0,0,0), with diameter 18.8 um

pt3dadd(18.8,0,0,18.8)

// Set end of soma at coordinate (18.8,0,0)

Ra = 123.0 // Resistivity in ohm‐cm (cm!!!)

insert hh // Add default Hodgkin-Huxley model

1. Now create a stimulus to kick off an action potential.

objectvar mystim // This is a variable name

soma mystim = new IClamp(0.5) // Put halfway along this soma

// Can also VClamp

mystim.del = 100 // ms, Delay until stim turns on

mystim.dur = 100 // ms, How long stim stays on

mystim.amp = 0.1 // nA, from inside to outside membrane

tstop = 300 // ms, defaults to 5

1. Once saved as a .hoc file, open the script with NEURON. You can do this in two ways: 1) Start NEURON and open the .hoc file using File🡪load hoc; or 2) Double-click on the .hoc file directly from Windows to open it in NEURON. Make sure to save the file before opening it in NEURON.
2. Select Tools🡪RunControl to open Run Control options and Graph🡪Voltage axis to view a graph of the compartment’s membrane voltage.
3. In the RunControl menu, click Init & Run to run the simulation. Observe the output in the graph window. You should see a burst of action potentials.

Note: NEURON is not great at handling memory. If things are acting wonky, exit the program and reopen it before continuing to debug.

Note: The code described above can also be entered line by line in NEURON’s command line interface. Using the command line is a convenient way of editing variables and parameters without reloading the .hoc file.

# Part 2) Model the effects of axon diameter on action potential speed

Now let’s use the tools from Part 1 to model an action potential traveling along two axons of varying diameter—one 10 um across, and the other 20 um.

1. Start by calculating the length constant for each of these axons. Use and . Recall the length constant equation from Lab 1:

where is the radius of the axon. These length constants should give you an idea of what to expect in the subsequent simulations.

1. Create a new .hoc file for this simulation. To simulate these axons, you will create 1000 compartments and link them together for each axon. To create the multiple compartments, use the following framework:

numParts = 1000

create axon[numParts]

access axon

for (i=0; i<numParts; i=i+1){

axon[i]{

// define properties

}

}

This creates a vector of axon compartments that you will connect in the next step. Use default Hodgkin-Huxley model parameters (do not modify Ra), with length = 100 and diam = 10 for each compartment.

1. To connect compartments, use the following syntax:

connect axon[0](1), axon[1](0)

// connects end of axon[0] to start of axon[1]

You will need to incorporate this into a for loop to connect all 1000 compartments in sequence.

1. Create a stimulus at one end of the axon as demonstrated in Part 1. Make sure to apply stimulation at an axon compartment at one of the ends.
2. Create an identical second axon in the same file, with diam = 20. You will need to create a separate stimulus for each axon. For plotting purposes, it may be easiest to have each stimulus start at a different time.
3. To measure the speed of the action potential along the axons, we will need to observe voltage from two different points along two different axons. To graph voltages from different compartments, right click on the graph and select ‘Plot what?’

In the popup window, you can select the compartment and variable to plot by double clicking options in the table. You can also directly enter or edit the symbol name in the space at the top of the popup window. For example, if you want to plot voltage at the 800th compartment of axon1, you can enter axon1[800].v(0.5).

To make different plots more distinguishable, right click on the graph and select ‘Color/Brush’. You can select a color and click on the corresponding legend entry on the graph to change its color and line width.

1. Use the crosshair tool to measure the speed of an action potential along the lengths of the two different axons.

# Part 3) Replicate findings from the Mainen neuron

The Mainen neuron is a detailed model of a layer 5 pyramidal cell that is frequently used in modeling studies. You’ll see it everywhere in literature if you start looking. Here, we’re going to replicate one of the paper’s key figures. We are going to use ModelDB, which contains hundreds of computational neuroscience models used from published research papers.

1. Download the Mainen neuron from ModelDB and extract to a new folder on the computer. Use the files associated with the 1995 paper by Mainen ZF, Joerges J, Huguenard JR, and Sejnowski TJ, A model of spike initiation in neocortical pyramidal neurons.

Do not extract to the same folder used for Parts 1 and 2.

1. You will need to rerun mknrndll.hoc (from Part 0), pointed at the new folder (‘spikeinit’). This will create new files in the folder that NEURON will use for simulating the Mainen neuron.
2. Load demo.hoc into NEURON and reproduce Figure 3A from the paper, i.e., create a figure showing the voltage traces of an action potential recorded from the soma and from a dendrite of the neuron. Also look at voltage traces from other parts of the neuron.

# Part 4) Build your own simplified neuron model

Due to the complexity of the Mainen neuron, it can be difficult to scale it to different sizes while maintaining key characteristics, like action potential shape. Here we will build a simplified neuron model that is easily scalable in size for different simulations.

1. Create a new .hoc file in the folder used for Parts 1 and 2. Download the ‘Part 4 files’ folder from Canvas and put all of the files in this folder directly into your folder from Part 1 and 2. Run mknrndll.hoc again on this folder to add in the new functionality from these files.
2. Start by creating the cell soma, dendrite, and axon. Parameters below:

|  |  |  |  |
| --- | --- | --- | --- |
|  | length (um) | diameter (um) | nseg |
| soma | 24 | 21 | 100 |
| dendrite | 50 | 12 | 222 |
| non-myelinated axon | 16 | 1 | 100 |
| myelinated axon | 300 | 1 | 100 |

Note: Multiple compartments can be created in the same .hoc file using the following syntax:

create part1, part2

access part1 // NEURON needs you to set a default object

part1{

// define properties

}

part2{

// define properties

}

Note: You MUST use pt3dadd() for creating compartments in this model. It is necessary for the data export script at the end of Part 4.

Note: You’ll have a much easier time with the later parts of this exercise if you program the compartment dimensions/coordinates in a way that is easily scalable. The last part of this exercise asks you to model neurons at half and twice the size of the original.

1. Specify membrane properties by adding the following code into each compartment’s properties:

|  |  |  |  |
| --- | --- | --- | --- |
| soma | dendrite | non-myelinated axon | myelinated axon |
| insert pas  Ra=150  cm=0.75  g\_pas=1/30000  e\_pas=-70  insert na  gbar\_na=20  insert kv  gbar\_kv=200  insert km  gbar\_km=0.1  insert kca  gbar\_kca=3  insert ca  gbar\_ca=0.3  insert cad | insert pas  Ra = 150  cm = 0.75  g\_pas = 1/30000  e\_pas = -70  insert na  gbar\_na = 20  insert km  gbar\_km = 0.1  insert kca  gbar\_kca = 3  insert ca  gbar\_ca = 0.3  insert cad | insert pas  Ra=150  cm=0.75  g\_pas=1/30000  e\_pas=-70  insert na  gbar\_na=30000  insert kv  gbar\_kv=2000 | insert pas  Ra=150  cm=0.04  g\_pas=1/30000  e\_pas=-70  insert na  gbar\_na=20 |

1. Create the axon hillock. The hillock is a conical section, so it should be created as a series of short single-compartment pieces with decreasing diameter. Parameters below:

* number of compartments = 9
* total length = 9 um
* starting diameter (soma end) equal to soma diameter
* ending diameter (axon end) equal to axon diameter
* membrane properties identical to non-myelinated axon

1. Now attach the different pieces together in order: dendrite🡪soma🡪axon hillock🡪non-myelinated axon🡪myelinated axon.
2. Apply stimulation to the middle of the axon hillock at 5 ms. Use 0.62 nA amplitude over 5 ms.
3. Copy and paste the following code to the end of the script to write currents to a data file:

// --------------------------------------------------------------

// Exports Currents and Geometry

// --------------------------------------------------------------

forall {

insert extracellular

insert xtra

}

load\_file("interpxyz.hoc") // only interpolates sections that have extracellular

load\_file("setpointers.hoc") // automatically calls grindaway() in interpxyz.hoc

// RECORD SECTION POSITIONS

objref f2

f2=new File()

f2.wopen("coordinates") // coordinate file name

f2.printf("name\tx\ty\tz\n")

forall{

for (x) if(x!=0 && x!=1){

f2.printf("%s(%g)\t%f\t%f\t%f\n", secname(), x, x\_xtra(x), y\_xtra(x), z\_xtra(x))

}

}

f2.close()

// RECORD MEMBRANE CURRENTS

objref f1

f1 = new File()

f1.wopen("currents") // current file name

finitialize()

fcurrent()

// write 'time' and section names (ms)

f1.printf("time (ms)\t")

forall {

for (x) if(x!=0 && x!=1){

f1.printf("%s(%g)\t",secname(),x)

}

}

f1.printf("\n")

// write x values (um)

f1.printf("-1\t")

forall {

for (x) if(x!=0 && x!=1){

f1.printf("%f\t",x\_xtra(x))

}

}

f1.printf("\n")

// write y values (um)

f1.printf("-1\t")

forall {

for (x) if(x!=0 && x!=1){

f1.printf("%f\t",y\_xtra(x))

}

}

f1.printf("\n")

// write z values (um)

f1.printf("-1\t")

forall {

for (x) if(x!=0 && x!=1){

f1.printf("%f\t",z\_xtra(x))

}

}

f1.printf("\n")

// write currents (in Amps)

// note i\_membrane is in mA/cm2 & area is in um2

proc advance() {

f1.printf("%f\t", t)

forall {

for(x) if (x!=0 && x!=1){

f1.printf("%e\t", i\_membrane(x)\*area(x)\*(1e-11)) //current in Amps

}

}

f1.printf("\n")

fadvance()

}

run()

1. Run the simulation over 30 ms with 0.025 ms time steps, and observe voltage traces from a few different spots along the neuron. Make sure to examine at least one spot along the dendrite, soma, axon hillock, and axon.
2. The script will create two files in your folder: ‘currents’ and ‘coordinates’. Rename them to reflect the size of the neuron used in the model. They are tab delimited text files that can be opened with Notepad or Notepad++.
3. To see the effects of cell size on recorded action potentials, scale the size of the neuron by a factor of 0.5x and 2.0x. Observe the currents recorded from these models and note any differences. (You may need to adjust the stimulus amplitudes to get resized cells to fire.) Save and rename the exported ‘currents’ and ‘coordinates’ files so that you know which model was used to create them. We will use these files in a later lab.
4. Use MATLAB (‘importdata’ function) to examine the currents at the axon hillock for each cell size. When examining the currents in Matlab, be sure to exclude the first 100 or so data points. Otherwise, your plots might be scaled too far out due to the initial conditions.

# Guidelines for Lab Report (on Labs 1 and 2 together)

*Introduction:* The introduction should be one paragraph long summarizing what detailed single neuron simulations can be used for (motivation), what data they draw upon from past experiments, and a brief summary of everything you will show in this lab report.

*Methods:* From Lab 2, there should be three methods paragraphs (and diagrams if you like) on:

1. Assumptions of the models used
   1. Parameters, based on Hodgkin-huxley,
2. How the model axons were designed
   1. Spatially how are voltages going through neuron
3. The use of the Mainen neuron
4. How the simplified neuron was designed and the experiments carried out with it

Include the code as an Appendix to your report. Always say where you looked up a value, for example from the Mainen paper.

*Other Methods from Lab 1 will also go into the methods section. Make it one cohesive report.*

*Results:* You should include the following in your Results:

1. The outcome of stimulation with a one-compartment model.
2. The effects of axon diameter on action potential propagation, with quantitative descriptions.
3. Recapitulate the key figure from Mainen 1995.
4. Describe and explain the output of the simplified neuron at different points along the cell, and describe the effects of scaling the neuron’s size.

Include all figures produced by NEURON that could help explain results.

*Discussion:* Should be 2-3 paragraphs long describing what you could use these models for in the future.

The report (not including Appendix and figures) should be no longer than four pages. Please upload your report to Canvas and leave a hard-copy with your GSI in lab. The hard-copy will be graded, so be sure different lines on your plots are distinguishable (using color or different line styles).