**MODELING A SINGLE COMPARTMENT FAST SPIKING INTERNEURON**

**Problem Statement**: Design an interneuron with specified passive and firing properties.

You are to model a fast spiking (FS) interneuron reported in Golomb et al. (2007). For this, you will use the package NEURON, and start with the templates provided to you in the zipped folder ‘Interneuron\_Startup files’. As a first step, peruse the paper by Golomb et al. (2003) that has a complete description of the model and make sure you understand all the biological properties modeled.

Golomb D, Donner K, Shacham L, Shlosberg D, Amitai Y, Hansel D (2007) Mechanisms of spiking cortical interneurons. *PLOS* Comput Biol 3:e 156.

WHERE DO YOU BEGIN: First run the files passive.hoc and then spiking neuron model in ‘Getting started with NEURON’.

THEN, you can start with a new file named ‘Main.hoc’ (or any name you want to provide) which should make a cell from a cell-template file, and provide it with a current injection input and run it. Use the file “cell\_template.hoc” in the zipped folder “Interneuron\_Startup Files” at the Blackboard site to define your single cell model and use that folder for your project files. Note that the cell\_template.hoc uses several .mod files of appropriate channels from that folder. See Appendix A-D for additional information. Note that you will have only two ‘.hoc’ files, cell\_template.hoc and Main.hoc, which call other ‘.mod’ files provided to you in the “Interneuron\_Startup Files” folder.

**What properties need to be matched?**

1. Passive properties.
2. Firing properties:

(i) responses to various current injection values (Fig 2, Golomb et al., 2007), by varying only kD, and

(ii) the frequency-current response (FIR) curve for the neuron.

Steps you can follow:

1. Insert current channels nas and kdr into cell\_template.hoc (given to you in the “Interneuron\_Startup Files” folder (see Appendix A). questions in this file
2. Then make a main.hoc file that loads cell\_template.hoc into it (see Appendix B).
3. Assign conductances as gna\_nas = x.xxx, gkdr\_kdr = x.xxx. The default unit is Siemens/cm2. Use plotting code from the graphics code as appropriate (see Appendix C to see how they are used)
4. Try different gna\_nas and gkdr\_kdr values (within the ranges provided), and find the values that can make a cell spike.
5. Observe the frequency and amplitude for the different parameter sets.
6. Now insert kD into cell\_template.hoc if necessary to match all the properties required.

Suggestions for model development

* NEURON models the compartment membrane as a cylindrical surface without top and bottom caps In the template file provided to you, we use L=3.1831 µm, and diam=10 µm, to give a total membrane area of 100 µm2
* The default capacitance of the model is 1 µm/cm2
* Use the following reversal potential values:

L

* ELeak = - 70 mV
* Ena = 50 mV
* Ek = - 90 mV

diam

* Initialize the membrane potential to – 70.038 mV.
* Membrane capacitance, channel conductances and output currents are provided per unit area in NEURON, i.e., in the units of S/cm2, and mA/cm2, respectively. It takes care of converting values internally.
* The default units for current injection and synaptic currents are nA.
* Useful links:

NEURON website: <http://www.neuron.yale.edu/neuron/node/47>

NEURON official forum: <http://www.neuron.yale.edu/phpBB2/viewforum.php?f=15>

NEURON Course Hand-outs <http://www.neuron.yale.edu/neuron/static/courses/2008/course/handson.html>

Programmer's Reference <http://www.neuron.yale.edu/neuron/static/docs/help/quick_reference.html>

**Mod files (.mod) for the following currents are provided to you:**

* nas.mod Fast spike generating Na+ current channel
* Kdr.mod Delayed rectifier K+ current channel
* kD.mod D-type K+ current

**Appendix A:** Edit the file “cell\_template.hoc” given to you in the startup files, and insert channels (suggested ranges for parameters are provided) at the right locations in that file. Don’t change the other statements in that file.

// In the cell\_template.hoc file, add the following as appropriate

Leak channel(passive channel):

insert pas

g\_pas= 0.00025 (Siemens/cm2)

Na channel:

insert na

gna\_nas = xx // (0.05~0.3 Siemens/cm2)

Kdr channel:

insert kdr

gkdr\_kdr = xx // (0.1~0.3 Siemens/cm2)

kD channel:

insert kD

gkD\_kD = xx // (.0001~0.0018 Siemens/cm2)

Indicate the reversal potentials after inserting all channels.

e\_pas= -70// (mV)

ena = xx // (mV)

ek = xx // (mV)

eh = xx // (mV)

**Appendix B: Make a file main.hoc; define a cell using the template you made, and then provide it with current injection (Note: some commands may be missing!)**

load\_file("nrngui.hoc")

load\_file("Cell\_template.hoc")

v\_init = xxxx

objref cell

cell = new Cell(1)

access cell.soma

secondorder = 2

dt = 0.01 // (ms)

steps\_per\_ms = 50

tstop = 1000

///// current clamp ////

delay\_Int = 48 //(ms)

duration\_Int = 1e+09//(ms)

amplitude\_Int = 3.35 //(pA)

objref ccl, cclA

cell.soma cclA = new IClamp(0.5)

cclA.del = delay\_Int // (ms)

cclA.dur = duration\_Int // (ms)

cclA.amp = amplitude\_Int\*1e-3 // (nA)

**Appendix C: Using the graphic GUI in main.hoc and some other NEURON functions (You can add modules like graphs, conductance menus, current/voltage clamp setting etc….)**

///////////////////////////////////////////////////////////////////////////////

SubVBoxNum = 4

objref MainHBoxObj,SubVBoxObj[SubVBoxNum]

proc MainBox() { local i

MainHBoxObj = new HBox()

for i=0,SubVBoxNum-1 SubVBoxObj[i] = new VBox()

SubVBoxObj[0].intercept(1)

newPlotVoltage1()

SubVBoxObj[0].intercept(0)

SubVBoxObj[1].intercept(1)

SubVBoxObj[1].intercept(0)

SubVBoxObj[2].intercept(1)

access cell.soma

nrnsecmenu(.5,1)

SubVBoxObj[2].intercept(0)

SubVBoxObj[3].intercept(1)

nrncontrolmenu()

xpanel(" ")

xbutton("QUIT","quit()")

xpanel(0)

nrnpointmenu(cclA)

SubVBoxObj[3].intercept(0)

MainHBoxObj.intercept(1)

for i=0,SubVBoxNum-1 SubVBoxObj[i].map()

MainHBoxObj.intercept(0)

MainHBoxObj.map("Re-Tune",10,25,800,600)

}

strdef tstr,tstr1

proc newPlotVoltage1() {

newPlot(0,tstop,-100,50)

graphItem.save\_name("graphList[0].")

graphList[0].append(graphItem)

graphItem.addexpr("cell.soma.v(.5)")

graphItem.label(.08,.925,"mV")

}

MainBox()

objref cells\_list

cells\_list=new List()

cells\_list.append(cell)

objref tvec, idvec // will be Vectors that record all spike times (tvec)

// and the corresponding id numbers of the cells that spiked (idvec)

proc spikerecord() {local i localobj nc, nil

tvec = new Vector()

idvec = new Vector()

for i=0, cells\_list.count()-1 {

nc = cells\_list.object(i).connect2target(nil)

nc.record(tvec, idvec, i)

// the Vector will continue to record spike times even after the NetCon has been destroyed

}

}

spikerecord()

run()

objref savetspikes

savetspikes = new File()

savetspikes.wopen("data.dat")

proc sspikeout() { local i

for i=0, tvec.size-1 {

savetspikes.printf("%8.4f\t %d\n", tvec.x[i], idvec.x[i]) }

}

sspikeout()

savetspikes.close()