BME 503: Exploration 1

Contents

[Part 1 2](#_Toc114497410)

[Part 2 Connor Stevens Model 4](#_Toc114497411)

[Part 3 Pyramidal Neuron 4](#_Toc114497412)

[Part 4 Linear Integrate and Fire Neuron 5](#_Toc114497413)

[Part 5 Threshold Parameter in Different Neural Models 6](#_Toc114497414)

[Appendix 8](#_Toc114497415)

[Appendix 1.1 8](#_Toc114497416)

[Appendix 1.2 10](#_Toc114497417)

[Appendix 1.3 12](#_Toc114497418)

[Appendix 1.4 14](#_Toc114497419)

[Appendix 2 16](#_Toc114497420)

[Appendix 3 18](#_Toc114497421)

[Appendix 4 20](#_Toc114497422)

[Appendix 5.1 23](#_Toc114497423)

[Appendix 5.2 25](#_Toc114497424)

Jinshui Zhang

09/19/2022

# Part 1

1. Using HodgkinHuxleyOriginalV2.py, implement the classical HH model as described in the handout HodgkinHuxley.pdf. In the classical model, the rest potential is 0.0 mV.

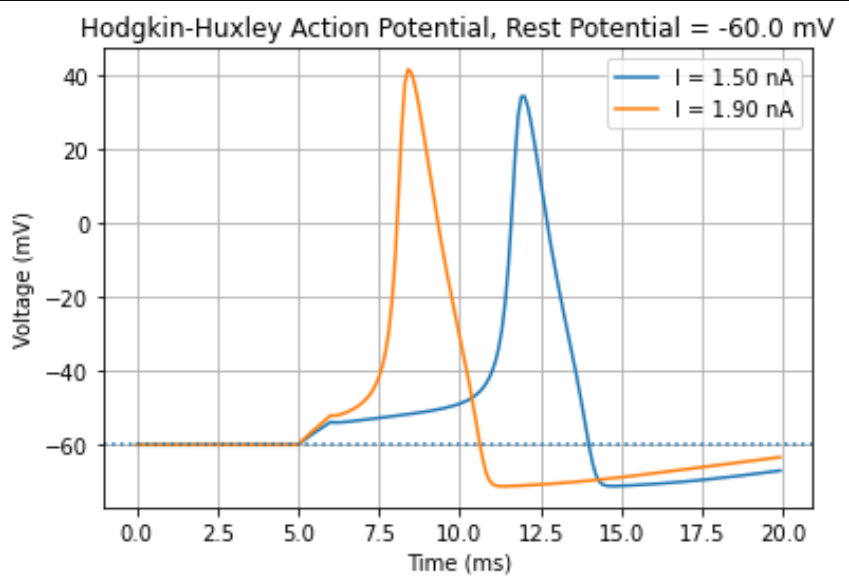
*See code at Appendix 1.*

Chart, line chart

Description automatically generated

2. Change the model so the rest potential is -60.0 mV. You will need to change the rate constants and the Nernst potentials.

*See code at Appendix 2.*



3. After 5msec, apply a 1 msec pulse and determine the threshold current in nA. What is the threshold?

*See code at Appendix 3.*

Sample the current range 0~2 nA by 100 points. Compare the action potential with -40 mV to detect if there is any pulse fired.

Chart

Description automatically generated

**Obtain the current threshold as 1.49 nA.**

4. Apply a long (2000 msec or longer) pulse with amplitudes ranging from 0.0 nA to 7nA. What is the firing rate as a function of current amplitude? Use the updates made to your code or modify IF-HodgkinHuxleyOriginal-skel2.py to implement 100 neurons in a group and apply a different current to each cell from 0-99 with the expression group.

*See code at Appendix4.*

Obtain the firing rate curve as below.

Chart, line chart

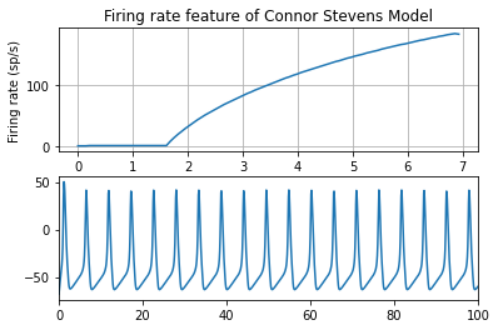
Description automatically generated

The above picture also shows a comparison between the firing curve and the threshold obtained in Q3. As predicted, the threshold current for constant stimulation is lower than the 1-ms stimulation.

# Part 2 Connor Stevens Model

*See code at Appendix 2.*

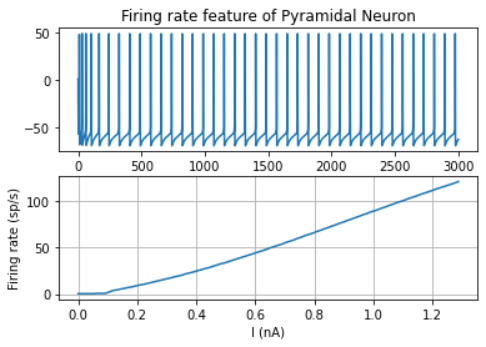
Obtain the Connor Stevens Model as below.



# 

# Part 3 Pyramidal Neuron

*See code at Appendix 3.*



The firing rate curve of pyramidal neurons is obtained as above. A key feature that we can observe from the result is that the firing is faster at the beginning. Then it tends to be regularly fired.

# Part 4 Linear Integrate and Fire Neuron

*See code at Appendix 4.*

Chart, line chart

Description automatically generatedChart

Description automatically generatedGraphical user interface, chart

Description automatically generated

Modify the code by getting rid of potassium and sodium channels’ equations. Also, in the simplified model, the refractory would be an important parameter that affects the firing rate.

Set the ‘refractory’ respectively as 0ms, 3ms and 5ms. We can obtain the different firing rate curves as above. The longer the refractory time is, the earlier the curve saturates, and the lower the up-limit firing rate is.

# Part 5 Threshold Parameter in Different Neural Models

For bio and simplified models.

The biological models such as HH model is derived by curve fitting on the experimental data. Therefore, their behavior doesn't need the parameter "threshold" to trigger the pulse.

Therefore, for the program below of HH model, no matter how we change the threshold, the behavior will always be the same. Instead, the parameter may play its role in counting the effective pulses.

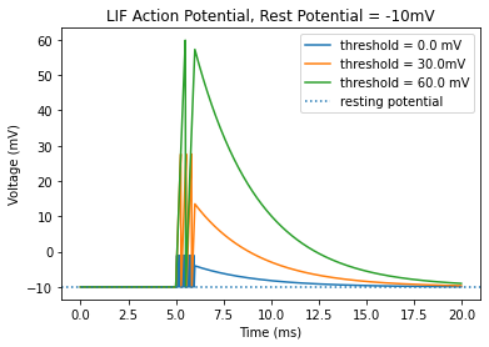
*See code Appendix 5.1.*

Chart, line chart

Description automatically generated

However, in simplified models such as LIF model, we will need the parameter "threshold" to trigger the pulse manually.

In LIF models, different thresholds will give us totally different results. As below, a smaller threshold will lead to a higher firing rate.



# Appendix

## Appendix 1.1

from brian2 import \*

import time

start = time.time()

num\_neurons = 2

# Parameters

area=20000\*umetre\*\*2

Cm = 1\*ufarad\*cm\*\*-2

El = 10.613\*mV

EK = -12.0\*mV

ENa = 115.0\*mV

E\_rest = 0\*mV

gl = 0.3\*(msiemens)/(cm\*\*2)

gNa = 120.0\*(msiemens)/(cm\*\*2)

gK = 36.0\*(msiemens)/(cm\*\*2)

defaultclock.dt=.1\*ms

div=defaultclock.dt

#The model

eqs\_ina = '''

ina=gNa \* m\*\*3 \* h \* (ENa-(v)) : amp/meter\*\*2

dm/dt = alpham \* (1-m) - betam \* m : 1

dh/dt = alphah \* (1-h) - betah \* h : 1

alpham = (0.1/mV) \* (-v+25.0\*mV) / (exp((-v+25.0\*mV) / (10.0\*mV)) - 1.0) /ms : Hz

betam = 4.0\*exp(-v/(18.0\*mV))/ms : Hz

alphah = 0.07\*exp(-v/(20.0\*mV))/ms : Hz

betah = 1.0/(exp((-v+30.0\*mV) / (10.0\*mV))+1.0)/ms : Hz

'''

eqs\_ik = '''

ik=gK \* n\*\*4 \* (EK-v):amp/meter\*\*2

dn/dt = alphan \* (1.0-n) - betan \* n : 1

alphan = (0.01/mV) \* (-v+10.0\*mV) / (exp((-v+10.0\*mV) / (10.0\*mV)) - 1.0)/ms : Hz

betan = 0.125\*exp(-v/(80.0\*mV))/ms : Hz

'''

eqs\_il = '''

il = gl \* (El-v) :amp/meter\*\*2

'''

eqs = '''

dv/dt = (ina+ik+il +I/area)/Cm : volt

I : amp

'''

eqs += (eqs\_ina+eqs\_ik+eqs\_il)

# Threshold and refractoriness are only used for spike counting

group = NeuronGroup(num\_neurons, eqs,clock=Clock(defaultclock.dt),

threshold='v > -40\*mV',

refractory='v > -40\*mV',

method='exponential\_euler')

group.v = 0\*mV

group.m=0.0529

group.n=0.3177

group.h=0.596

monitor2=StateMonitor(group,'v',record=True)

group.I = 0\*nA

run(5.0\*ms,report='text')

group.I[0] = 1.50\*nA

group.I[1] = 1.90\*nA

run(1\*ms, report='text')

group.I = 0\*nA

run(14.0\*ms)

figure(1)

plot(monitor2.t/ms, monitor2.v[0]/mV) #plot the voltage for neuron 0 (index starts at 0)

plot(monitor2.t/ms, monitor2.v[1]/mV) #plot the voltage for neuron 0 (index starts at 0)

# ylim(-20,120) #set axes limits

# xlim(0,20)

xlabel('Time (ms)')

ylabel('Voltage (mV)')

title('Hodgkin-Huxley Action Potential, Rest Potential = 0mV')

#You can dump your results to a file to visualize separately

savetxt('Vmdata.dat',(monitor2.t/ms, monitor2.v[0]/mV))

#out=np.loadtxt('Vmdata.dat')

#plot(out[0],out[1])

legend(['I = 1.50 nA', 'I = 1.90 nA'])

grid()

show()

print('Script took', time.time()-start, 'seconds.')

## Appendix 1.2

from brian2 import \*

import time

start = time.time()

num\_neurons = 2

# Parameters

area=20000\*umetre\*\*2

Cm = 1\*ufarad\*cm\*\*-2

El = (10.613 - 60)\*mV

EK = (-12.0 - 60)\*mV

ENa = (115.0 - 60)\*mV

# en\_var = 1.4

# El = 10.613\*mV \* en\_var

# EK = -12.0\*mV \* en\_var

# ENa = 115.0\*mV \* en\_var

E\_rest = (0 - 60)\*mV

gl = 0.3\*(msiemens)/(cm\*\*2)

gNa = 120.0\*(msiemens)/(cm\*\*2)

gK = 36.0\*(msiemens)/(cm\*\*2)

defaultclock.dt=.1\*ms

div=defaultclock.dt

#The model

eqs\_ina = '''

ina=gNa \* m\*\*3 \* h \* (ENa-(v)) : amp/meter\*\*2

dm/dt = alpham \* (1-m) - betam \* m : 1

dh/dt = alphah \* (1-h) - betah \* h : 1

alpham = (0.1/mV) \* (-(v+60\*mV)+25.0\*mV) / (exp((-(v+60\*mV)+25.0\*mV) / (10.0\*mV)) - 1.0) /ms : Hz

betam = 4.0\*exp(-(v+60\*mV)/(18.0\*mV))/ms : Hz

alphah = 0.07\*exp(-(v+60\*mV)/(20.0\*mV))/ms : Hz

betah = 1.0/(exp((-(v+60\*mV)+30.0\*mV) / (10.0\*mV))+1.0)/ms : Hz

'''

eqs\_ik = '''

ik=gK \* n\*\*4 \* (EK-v):amp/meter\*\*2

dn/dt = alphan \* (1.0-n) - betan \* n : 1

alphan = (0.01/mV) \* (-(v+60\*mV)+10.0\*mV) / (exp((-(v+60\*mV)+10.0\*mV) / (10.0\*mV)) - 1.0)/ms : Hz

betan = 0.125\*exp(-(v+60\*mV)/(80.0\*mV))/ms : Hz

'''

eqs\_il = '''

il = gl \* (El-v) :amp/meter\*\*2

'''

eqs = '''

dv/dt = (ina+ik+il +I/area)/Cm : volt

I : amp

'''

eqs += (eqs\_ina+eqs\_ik+eqs\_il)

# Threshold and refractoriness are only used for spike counting

group = NeuronGroup(num\_neurons, eqs,clock=Clock(defaultclock.dt),

threshold='v > -40\*mV',

refractory='v > -40\*mV',

method='exponential\_euler')

group.v = E\_rest

group.m=0.0529

group.n=0.3177

group.h=0.596

# update Nernst potentials

E\_rest = -60 \* mV

monitor2=StateMonitor(group,'v',record=True)

group.I = 0\*nA

run(5.0\*ms,report='text')

group.I[0] = 1.50\*nA

group.I[1] = 1.90\*nA

run(1\*ms, report='text')

group.I = 0\*nA

run(14.0\*ms)

figure(1)

plot(monitor2.t/ms, monitor2.v[0]/mV) #plot the voltage for neuron 0 (index starts at 0)

plot(monitor2.t/ms, monitor2.v[1]/mV) #plot the voltage for neuron 0 (index starts at 0)

axhline(E\_rest / mV, ls=":")

# ylim(-20-60,120) #set axes limits

# xlim(0,20)

xlabel('Time (ms)')

ylabel('Voltage (mV)')

title('Hodgkin-Huxley Action Potential, Rest Potential = -60.0 mV')

#You can dump your results to a file to visualize separately

savetxt('Vmdata.dat',(monitor2.t/ms, monitor2.v[0]/mV))

legend(['I = 1.50 nA', 'I = 1.90 nA'])

grid()

#out=np.loadtxt('Vmdata.dat')

#plot(out[0],out[1])

show()

print('Script took', time.time()-start, 'seconds.')

## Appendix 1.3

from brian2 import \*

import time

start = time.time()

num\_neurons = 1

# Parameters

area=20000\*umetre\*\*2

Cm = 1\*ufarad\*cm\*\*-2

El = (10.613 - 60)\*mV

EK = (-12.0 - 60)\*mV

ENa = (115.0 - 60)\*mV

# en\_var = 1.4

# El = 10.613\*mV \* en\_var

# EK = -12.0\*mV \* en\_var

# ENa = 115.0\*mV \* en\_var

E\_rest = (0 - 60)\*mV

gl = 0.3\*(msiemens)/(cm\*\*2)

gNa = 120.0\*(msiemens)/(cm\*\*2)

gK = 36.0\*(msiemens)/(cm\*\*2)

defaultclock.dt=.1\*ms

div=defaultclock.dt

#The model

eqs\_ina = '''

ina=gNa \* m\*\*3 \* h \* (ENa-(v)) : amp/meter\*\*2

dm/dt = alpham \* (1-m) - betam \* m : 1

dh/dt = alphah \* (1-h) - betah \* h : 1

alpham = (0.1/mV) \* (-(v+60\*mV)+25.0\*mV) / (exp((-(v+60\*mV)+25.0\*mV) / (10.0\*mV)) - 1.0) /ms : Hz

betam = 4.0\*exp(-(v+60\*mV)/(18.0\*mV))/ms : Hz

alphah = 0.07\*exp(-(v+60\*mV)/(20.0\*mV))/ms : Hz

betah = 1.0/(exp((-(v+60\*mV)+30.0\*mV) / (10.0\*mV))+1.0)/ms : Hz

'''

eqs\_ik = '''

ik=gK \* n\*\*4 \* (EK-v):amp/meter\*\*2

dn/dt = alphan \* (1.0-n) - betan \* n : 1

alphan = (0.01/mV) \* (-(v+60\*mV)+10.0\*mV) / (exp((-(v+60\*mV)+10.0\*mV) / (10.0\*mV)) - 1.0)/ms : Hz

betan = 0.125\*exp(-(v+60\*mV)/(80.0\*mV))/ms : Hz

'''

eqs\_il = '''

il = gl \* (El-v) :amp/meter\*\*2

'''

eqs = '''

dv/dt = (ina+ik+il +I/area)/Cm : volt

I : amp

'''

eqs += (eqs\_ina+eqs\_ik+eqs\_il)

# Threshold and refractoriness are only used for spike counting

group = NeuronGroup(num\_neurons, eqs,clock=Clock(defaultclock.dt),

threshold='v > -40\*mV',

refractory='v > 40\*mV',

method='exponential\_euler')

group.v = E\_rest

group.m=0.0529

group.n=0.3177

group.h=0.596

monitor2=StateMonitor(group,'v',record=True)

current\_array = np.linspace(0, 2, 100)

peak\_bi\_indicator = []

for i in current\_array:

group.I = 0\*nA

run(5.0\*ms)

group.I = i\*nA

run(1\*ms)

group.I = 0\*nA

run(14.0\*ms)

peak\_bi\_indicator.append((monitor2.v[0] > 40\*mV).any())

scatter(current\_array, peak\_bi\_indicator)

xlabel("Current [ns]")

ylabel("Indicator of Stimulation [Y/N]")

index\_thre = where(peak\_bi\_indicator)[0][0]

scatter(current\_array[index\_thre], peak\_bi\_indicator[index\_thre], marker='o', c='red', s=3)

title("The threshold current is {} nA. ".format(current\_array[index\_thre]))

show()

## Appendix 1.4

from brian2 import \*

num\_neurons = 100

duration = 2\*second

# Parameters

area=20000\*umetre\*\*2

Cm = 1\*ufarad\*cm\*\*-2

El = (10.613 - 60)\*mV

EK = (-12.0 - 60)\*mV

ENa = (115.0 - 60)\*mV

# en\_var = 1.4

# El = 10.613\*mV \* en\_var

# EK = -12.0\*mV \* en\_var

# ENa = 115.0\*mV \* en\_var

E\_rest = (0 - 60)\*mV

gl = 0.3\*(msiemens)/(cm\*\*2)

gNa = 120.0\*(msiemens)/(cm\*\*2)

gK = 36.0\*(msiemens)/(cm\*\*2)

defaultclock.dt=.1\*ms

div=defaultclock.dt

#The model

eqs\_ina = '''

ina=gNa \* m\*\*3 \* h \* (ENa-(v)) : amp/meter\*\*2

dm/dt = alpham \* (1-m) - betam \* m : 1

dh/dt = alphah \* (1-h) - betah \* h : 1

alpham = (0.1/mV) \* (-(v+60\*mV)+25.0\*mV) / (exp((-(v+60\*mV)+25.0\*mV) / (10.0\*mV)) - 1.0) /ms : Hz

betam = 4.0\*exp(-(v+60\*mV)/(18.0\*mV))/ms : Hz

alphah = 0.07\*exp(-(v+60\*mV)/(20.0\*mV))/ms : Hz

betah = 1.0/(exp((-(v+60\*mV)+30.0\*mV) / (10.0\*mV))+1.0)/ms : Hz

'''

eqs\_ik = '''

ik=gK \* n\*\*4 \* (EK-v):amp/meter\*\*2

dn/dt = alphan \* (1.0-n) - betan \* n : 1

alphan = (0.01/mV) \* (-(v+60\*mV)+10.0\*mV) / (exp((-(v+60\*mV)+10.0\*mV) / (10.0\*mV)) - 1.0)/ms : Hz

betan = 0.125\*exp(-(v+60\*mV)/(80.0\*mV))/ms : Hz

'''

eqs\_il = '''

il = gl \* (El-v) :amp/meter\*\*2

'''

eqs = '''

dv/dt = (ina+ik+il +I/area)/Cm : volt

I : amp

'''

eqs += (eqs\_ina+eqs\_ik+eqs\_il)

# Threshold and refractoriness are only used for spike counting

group = NeuronGroup(num\_neurons, eqs,

threshold='v > -40\*mV',

refractory='v > -40\*mV',

method='exponential\_euler')

group.v = E\_rest

group.m=0.0529

group.n=0.3177

group.h=0.596

monitor2=StateMonitor(group,'v',record=True)

#group.I = 0\*nA

#run(25.0\*ms,report='text')

#group.I = 1.7\*nA

#run(205.0\*ms, report='text')

#group.I = 0\*nA

#run(10.0\*ms)

group.I = '(7.0\*nA \* i) / num\_neurons'

monitor = SpikeMonitor(group)

run(duration)

figure(1)

plot(group.I/nA, monitor.count / duration)

xlabel('I (nA)')

ylabel('Firing rate (sp/s)')

axvline(current\_array[index\_thre], ls=":", c='red')

#figure(2)

#ylim(-80,60) #set axes limits

legend(["Firing rate curve", "Threshold for 1 ms current stimulation"])

grid()

show()

plot(monitor2.t/ms, monitor2.v[-1]/mV) #plot the voltage for neuron 0 (index starts at 0)

xlim((0, 100))

show()

## Appendix 2

from brian2 import \*

num\_neurons = 100

duration = 2\*second

# Parameters

area = 20000\*umetre\*\*2

Cm = 1\*ufarad\*cm\*\*-2

El = -17.0\*mV

EK = -72\*mV

ENa = 55.0\*mV

EA= -75.0\*mV

#gl = 0.3\*msiemens/cm\*\*2

#gNa = 120\*msiemens/cm\*\*2

#gK = 20.0\*msiemens/cm\*\*2

#gA=47.7\*msiemens/cm\*\*2

gl = 0.3\*msiemens/cm\*\*2

gNa = 120.0\*msiemens/cm\*\*2

gK = 20\*msiemens/cm\*\*2

gA= 47.7\*msiemens/cm\*\*2

#The model

eqs\_ina = '''

ina=gNa \* m\*\*3 \* h \* (ENa-v) : amp/meter\*\*2

dm/dt = alpham \* (1-m) - betam \* m : 1

dh/dt = alphah \* (1-h) - betah \* h : 1

alpham = 0.38/mV\*(v+29.7\*mV)/(1-exp(-0.1\*(v+29.7\*mV)/mV ) )/ms : Hz

betam = 15.2\*exp(-0.0556\*(v+54.7\*mV)/mV)/ms : Hz

alphah = 0.266\*exp(-0.05\*(v+48\*mV)/mV)/ms : Hz

betah = 3.8/(1+exp(-0.1\*(v+18.\*mV)/mV))/ms : Hz

'''

eqs\_iA = '''

iA = gA \* a\*\*3 \* b \* (EA-v) : amp/meter\*\*2

da/dt = (a\_inf - a) / tau\_a : 1

db/dt = (b\_inf - b) / tau\_b : 1

a\_inf = (0.0761 \* exp(0.0314\*(v+94.22\*mV)/mV) / (1+exp(0.0346\*(v+1.17\*mV)/mV)))\*\*(1/3) : 1

tau\_a = 0.3632\*ms + 1.158\*ms / (1 + exp(0.0497 \* (v + 55.968\*mV)/mV)) : second

b\_inf = (1 / (1 + exp(0.0688\*(v+53.3\*mV)/mV)))\*\*4 : 1

tau\_b = 1.24\*ms + 2.678\*ms / (1 + exp(0.0624 \* (v + 50\*mV)/mV)) : second

'''

eqs\_ik = '''

ik=gK \* n\*\*4 \* (EK-v):amp/meter\*\*2

dn/dt = alphan \* (1-n) - betan \* n : 1

alphan = (0.02\*(v+45.7\*mV)/mV)/(1-exp(-0.1\*(v+45.7\*mV)/mV))/ms : Hz

betan = 0.25\*exp(-0.0125\*(v+55.7\*mV)/mV)/ms : Hz

'''

eqs\_il = '''

il = gl \* (El-v) :amp/meter\*\*2

'''

eqs = '''

dv/dt = (ina+ik+il+iA+I/area)/Cm: volt

I : amp

'''

eqs += (eqs\_ina+eqs\_ik+eqs\_il+eqs\_iA)

# Threshold and refractoriness are only used for spike counting

group = NeuronGroup(num\_neurons, eqs,

threshold='v > 40\*mV',

refractory='v > 40\*mV',

method='exponential\_euler')

group.v = -68.0\*mV

group.m=0.0529

group.n=0.3177

group.h=0.596

monitor2=StateMonitor(group,'v',record=True)

#group.I = 0\*nA

#run(25.0\*ms,report='text')

#group.I = 1.7\*nA

#run(205.0\*ms, report='text')

#group.I = 0\*nA

#run(10.0\*ms)

group.I = '(7.0\*nA \* i) / num\_neurons'

monitor = SpikeMonitor(group)

run(duration)

figure(1)

plot(group.I/nA, monitor.count / duration)

xlabel('I (nA)')

ylabel('Firing rate (sp/s)')

title('Firing rate feature of Connor Stevens Model')

grid()

#figure(2)

#plot(monitor2.t/ms, monitor2.v[0]/mV) #plot the voltage for neuron 0 (index starts at 0)

#ylim(-80,60) #set axes limits

show()

## Appendix 3

from brian2 import \*

num\_neurons = 100

duration = 3.0\*second

# Parameters

area = 20000\*umetre\*\*2

Cm = (1\*ufarad\*cm\*\*-2)

defaultclock.dt=.02\*ms

div=defaultclock.dt

# The model

ENa=50.0 \*mV

gnabar=50\*msiemens\*cm\*\*-2

VT1=-61.5

VT=-61.5

EK=-90.0\*mV

gkbar=4.8\*msiemens\*cm\*\*-2

EKm=-90.0\*mV

#

gmbar = 0.15\*msiemens\*cm\*\*-2

#

glbar= 0.0205\*msiemens/cm\*\*2

El=-70\*mV

eqs\_na = """

ina = gnabar\*m\*\*3\*h\*(ENa-v) : amp/meter\*\*2

dm/dt = (am1\*(1-m)-bm1\*m): 1

dh/dt = (ah1\*(1-h)-bh1\*h): 1

am1=0.32\*(13-(vu-VT))/(exp((13-(vu-VT))/4.0)-1.0)/ms: Hz

bm1=(0.28\*((vu-VT)-40)/(exp(((vu-VT)-40)/5.0)-1.0))/ms: Hz

ah1 = 0.128\*exp(-(vu-17-VT)/18)/ms: Hz

bh1 = 4/(1+exp(-(vu-40-VT)/5))/ms: Hz

"""

# IM channel ()

# Non-inactivating potassium current

eqs\_m = """

im = gmbar\*c\*(EKm-v) : amp/meter\*\*2

dc/dt = (c\_inf - c) / tau\_c : 1

c\_inf = 1 / (1 + exp(-(v + 35\*mV) / (10\*mV))) : 1

tau\_c = 1123.5\*ms / (3.3 \* exp((v+35\*mV)/(20\*mV)) + exp(-(v + 35\*mV) / (20\*mV))) : second

"""

# K channel

eqs\_k = """

ik = gkbar\*b\*\*4\*(EK-v): amp/meter\*\*2

db/dt = (ab\*(1-b)-bb\*b): 1

ab=0.032\*(vu-15-VT1)/(1.0 - exp(-(vu-15-VT1)/5.0))/ms:Hz

bb=0.5\*exp(-(vu-10-VT1)/40)/ms : Hz

"""

# Leak

eqs\_leak = """

il = glbar\*(El-v) : amp/meter\*\*2

"""

eqs = """

dv/dt = (il + ik+ +ina+ im + I/area)/Cm : volt

vu = v/mV : 1 # unitless v

I: amp

"""

eqs += eqs\_leak + eqs\_k + eqs\_na +eqs\_m

# Threshold and refractoriness are only used for spike counting

P1 = NeuronGroup(num\_neurons, eqs,clock=Clock(defaultclock.dt),

threshold='v > -40\*mV',refractory='v > -40\*mV',method='euler')

P1.I='1.3\*nA \* i / num\_neurons'

monitor = SpikeMonitor(P1)

monitor2=StateMonitor(P1, ('v'), record=True)

net = Network(P1, monitor, monitor2)

net.run(duration)

figure(1)

subplot(2,1,1)

title('Firing rate feature of Pyramidal Neuron')

plot(monitor2.t/ms, monitor2.v[20]/mV)

subplot(2,1,2)

#Idata=I(numpy.arange(0,duration/ms,(div/ms))\*ms)

#plt.plot((numpy.arange(0,duration/ms,div/ms)),Idata)

plot(P1.I/nA, monitor.count / duration)

xlabel('I (nA)')

ylabel('Firing rate (sp/s)')

grid()

show()

## Appendix 4

from brian2 import \*

num\_neurons = 75

duration = 3.0\*second

# duration = 0.3\*second

# Parameters

area = 20000\*umetre\*\*2

Cm = (1\*ufarad\*cm\*\*-2)

defaultclock.dt=.02\*ms

div=defaultclock.dt

# The model

El=-10\*mV

E\_rest = El.copy()

gl = 0.3\*(msiemens)/(cm\*\*2)

# Leak

eqs\_leak = """

il = gl \* (El-v) :amp/meter\*\*2

"""

eqs = """

dv/dt = (il + I/area)/Cm : volt

I: amp

"""

eqs += eqs\_leak

# Threshold and refractoriness are only used for spike counting

P1 = NeuronGroup(num\_neurons,

eqs,

clock=Clock(defaultclock.dt),

threshold='v > 30\*mV',

# reset='v = 0\*mV',

reset='v = E\_rest',

refractory=0\*ms,

method='euler')

# group = NeuronGroup(n, eqs, threshold='v > 10\*mV', reset='v = 0\*mV',

# refractory=5\*ms, method='exact')

P1.I='7.0\*nA \* i / num\_neurons'

monitor = SpikeMonitor(P1)

monitor2=StateMonitor(P1, ('v'), record=True)

net = Network(P1, monitor, monitor2)

net.run(duration)

figure(1)

subplot(2,1,1)

plot(monitor2.t/ms, monitor2.v[60]/mV)

axhline(E\_rest/mV, ls=':')

subplot(2,1,2)

#Idata=I(numpy.arange(0,duration/ms,(div/ms))\*ms)

#plt.plot((numpy.arange(0,duration/ms,div/ms)),Idata)

plot(P1.I/nA, monitor.count / duration)

xlabel('I (nA)')

ylabel('Firing rate (sp/s)')

legend(["Zero refractory time"])

show()

# Threshold and refractoriness are only used for spike counting

P1 = NeuronGroup(num\_neurons,

eqs,

clock=Clock(defaultclock.dt),

threshold='v > 30\*mV',

# reset='v = 0\*mV',

reset='v = E\_rest',

refractory=3\*ms,

method='euler')

# group = NeuronGroup(n, eqs, threshold='v > 10\*mV', reset='v = 0\*mV',

# refractory=5\*ms, method='exact')

P1.I='7.0\*nA \* i / num\_neurons'

monitor = SpikeMonitor(P1)

monitor2=StateMonitor(P1, ('v'), record=True)

net = Network(P1, monitor, monitor2)

net.run(duration)

figure(1)

subplot(2,1,1)

plot(monitor2.t/ms, monitor2.v[60]/mV)

axhline(E\_rest/mV, ls=':')

subplot(2,1,2)

#Idata=I(numpy.arange(0,duration/ms,(div/ms))\*ms)

#plt.plot((numpy.arange(0,duration/ms,div/ms)),Idata)

plot(P1.I/nA, monitor.count / duration)

xlabel('I (nA)')

ylabel('Firing rate (sp/s)')

legend(["3ms refractory time"])

show()

# Threshold and refractoriness are only used for spike counting

P1 = NeuronGroup(num\_neurons,

eqs,

clock=Clock(defaultclock.dt),

threshold='v > 30\*mV',

# reset='v = 0\*mV',

reset='v = E\_rest',

refractory=5\*ms,

method='euler')

# group = NeuronGroup(n, eqs, threshold='v > 10\*mV', reset='v = 0\*mV',

# refractory=5\*ms, method='exact')

P1.I='7.0\*nA \* i / num\_neurons'

monitor = SpikeMonitor(P1)

monitor2=StateMonitor(P1, ('v'), record=True)

net = Network(P1, monitor, monitor2)

net.run(duration)

figure(1)

subplot(2,1,1)

plot(monitor2.t/ms, monitor2.v[60]/mV)

axhline(E\_rest/mV, ls=':')

subplot(2,1,2)

#Idata=I(numpy.arange(0,duration/ms,(div/ms))\*ms)

#plt.plot((numpy.arange(0,duration/ms,div/ms)),Idata)

plot(P1.I/nA, monitor.count / duration)

xlabel('I (nA)')

ylabel('Firing rate (sp/s)')

legend(["5ms refractory time"])

show()

## Appendix 5.1

from brian2 import \*

import time

start = time.time()

num\_neurons = 1

# Parameters

area=20000\*umetre\*\*2

Cm = 1\*ufarad\*cm\*\*-2

El = (10.613 - 60)\*mV

EK = (-12.0 - 60)\*mV

ENa = (115.0 - 60)\*mV

# en\_var = 1.4

# El = 10.613\*mV \* en\_var

# EK = -12.0\*mV \* en\_var

# ENa = 115.0\*mV \* en\_var

E\_rest = (0 - 60)\*mV

gl = 0.3\*(msiemens)/(cm\*\*2)

gNa = 120.0\*(msiemens)/(cm\*\*2)

gK = 36.0\*(msiemens)/(cm\*\*2)

defaultclock.dt=.1\*ms

div=defaultclock.dt

#The model

eqs\_ina = '''

ina=gNa \* m\*\*3 \* h \* (ENa-(v)) : amp/meter\*\*2

dm/dt = alpham \* (1-m) - betam \* m : 1

dh/dt = alphah \* (1-h) - betah \* h : 1

alpham = (0.1/mV) \* (-(v+60\*mV)+25.0\*mV) / (exp((-(v+60\*mV)+25.0\*mV) / (10.0\*mV)) - 1.0) /ms : Hz

betam = 4.0\*exp(-(v+60\*mV)/(18.0\*mV))/ms : Hz

alphah = 0.07\*exp(-(v+60\*mV)/(20.0\*mV))/ms : Hz

betah = 1.0/(exp((-(v+60\*mV)+30.0\*mV) / (10.0\*mV))+1.0)/ms : Hz

'''

eqs\_ik = '''

ik=gK \* n\*\*4 \* (EK-v):amp/meter\*\*2

dn/dt = alphan \* (1.0-n) - betan \* n : 1

alphan = (0.01/mV) \* (-(v+60\*mV)+10.0\*mV) / (exp((-(v+60\*mV)+10.0\*mV) / (10.0\*mV)) - 1.0)/ms : Hz

betan = 0.125\*exp(-(v+60\*mV)/(80.0\*mV))/ms : Hz

'''

eqs\_il = '''

il = gl \* (El-v) :amp/meter\*\*2

'''

eqs = '''

dv/dt = (ina+ik+il +I/area)/Cm : volt

I : amp

'''

eqs += (eqs\_ina+eqs\_ik+eqs\_il)

# Threshold and refractoriness are only used for spike counting

group = NeuronGroup(num\_neurons, eqs,clock=Clock(defaultclock.dt),

threshold='v > -40\*mV',

refractory='v > -40\*mV',

method='exponential\_euler')

group.v = E\_rest

group.m=0.0529

group.n=0.3177

group.h=0.596

figure(1)

# update Nernst potentials

E\_rest = -60 \* mV

monitor2=StateMonitor(group,'v',record=True)

group.I = 0\*nA

run(5.0\*ms,report='text')

group.I[0] = 1.50\*nA

run(1\*ms, report='text')

group.I = 0\*nA

run(14.0\*ms)

plot(monitor2.t/ms, monitor2.v[0]/mV) #plot the voltage for neuron 0 (index starts at 0)

group = NeuronGroup(num\_neurons, eqs,clock=Clock(defaultclock.dt),

threshold='v > 0\*mV',

refractory='v > -40\*mV',

method='exponential\_euler')

group.v = E\_rest

group.m=0.0529

group.n=0.3177

group.h=0.596

monitor2=StateMonitor(group,'v',record=True)

group.I = 0\*nA

run(5.0\*ms,report='text')

group.I[0] = 1.50\*nA

run(1\*ms, report='text')

group.I = 0\*nA

run(14.0\*ms)

plot(monitor2.t/ms, monitor2.v[0]/mV) #plot the voltage for neuron 0 (index starts at 0)

xlabel('Time (ms)')

ylabel('Voltage (mV)')

title('Hodgkin-Huxley Action Potential, Rest Potential = 0mV')

group = NeuronGroup(num\_neurons, eqs,clock=Clock(defaultclock.dt),

threshold='v > 30\*mV',

refractory='v > -40\*mV',

method='exponential\_euler')

group.v = E\_rest

group.m=0.0529

group.n=0.3177

group.h=0.596

monitor2=StateMonitor(group,'v',record=True)

group.I = 0\*nA

run(5.0\*ms,report='text')

group.I[0] = 1.50\*nA

run(1\*ms, report='text')

group.I = 0\*nA

run(14.0\*ms)

plot(monitor2.t/ms, monitor2.v[0]/mV) #plot the voltage for neuron 0 (index starts at 0)

xlabel('Time (ms)')

ylabel('Voltage (mV)')

title('Hodgkin-Huxley Action Potential, Rest Potential = -60.0 mV')

savetxt('Vmdata.dat',(monitor2.t/ms, monitor2.v[0]/mV))

axhline(E\_rest / mV, ls=":")

legend(['threshold = -40.0 mV', 'threshold = 0.0mV', 'threshold = 40.0 mV', 'resting potential'])

show()

## Appendix 5.2

from brian2 import \*

import time

start = time.time()

num\_neurons = 1

# Parameters

area = 20000\*umetre\*\*2

Cm = (1\*ufarad\*cm\*\*-2)

defaultclock.dt=.02\*ms

div=defaultclock.dt

# The model

El=-10\*mV

E\_rest = El.copy()

gl = 0.3\*(msiemens)/(cm\*\*2)

# Leak

eqs\_leak = """

il = gl \* (El-v) :amp/meter\*\*2

"""

eqs = """

dv/dt = (il + I/area)/Cm : volt

I: amp

"""

eqs += eqs\_leak

# Threshold and refractoriness are only used for spike counting

group = NeuronGroup(num\_neurons,

eqs,

clock=Clock(defaultclock.dt),

threshold='v > 0\*mV',

# reset='v = 0\*mV',

reset='v = E\_rest',

refractory=0\*ms,

method='euler')

group.v = E\_rest

figure(1)

# update Nernst potentials

monitor2=StateMonitor(group,'v',record=True)

group.I = 0\*nA

run(5.0\*ms,report='text')

group.I[0] = 30\*nA

run(1\*ms, report='text')

group.I = 0\*nA

run(14.0\*ms)

plot(monitor2.t/ms, monitor2.v[0]/mV) #plot the voltage for neuron 0 (index starts at 0)

group = NeuronGroup(num\_neurons,

eqs,

clock=Clock(defaultclock.dt),

threshold='v > 30\*mV',

# reset='v = 0\*mV',

reset='v = E\_rest',

refractory=0\*ms,

method='euler')

group.v = E\_rest

monitor2=StateMonitor(group,'v',record=True)

group.I = 0\*nA

run(5.0\*ms,report='text')

group.I[0] = 30\*nA

run(1\*ms, report='text')

group.I = 0\*nA

run(14.0\*ms)

plot(monitor2.t/ms, monitor2.v[0]/mV) #plot the voltage for neuron 0 (index starts at 0)

xlabel('Time (ms)')

ylabel('Voltage (mV)')

title('Hodgkin-Huxley Action Potential, Rest Potential = 0mV')

group = NeuronGroup(num\_neurons,

eqs,

clock=Clock(defaultclock.dt),

threshold='v > 60\*mV',

# reset='v = 0\*mV',

reset='v = E\_rest',

refractory=0\*ms,

method='euler')

group.v = E\_rest

monitor2=StateMonitor(group,'v',record=True)

group.I = 0\*nA

run(5.0\*ms,report='text')

group.I[0] = 30\*nA

run(1\*ms, report='text')

group.I = 0\*nA

run(14.0\*ms)

plot(monitor2.t/ms, monitor2.v[0]/mV) #plot the voltage for neuron 0 (index starts at 0)

xlabel('Time (ms)')

ylabel('Voltage (mV)')

title('LIF Action Potential, Rest Potential = -10mV')

savetxt('Vmdata.dat',(monitor2.t/ms, monitor2.v[0]/mV))

axhline(E\_rest / mV, ls=":")

legend(['threshold = -40.0 mV', 'threshold = 0.0mV', 'threshold = 40.0 mV', 'resting potential'])

show()