BME 503: Exploration 2

Contents

[Part 1: EIAF Model 2](#_Toc115168355)

[Part 2: Izhikevich Model 3](#_Toc115168356)

[Part 3: Synapses 5](#_Toc115168357)

[A) Alpha & Biological Models of Synapses 5](#_Toc115168358)

[1. Alpha function. 5](#_Toc115168359)

[2. Biophysical Models 5](#_Toc115168360)

[B) Improved model with different rise / decay time constants 6](#_Toc115168361)

[Part 4: Refractory influence on Exponential IAF Model 8](#_Toc115168362)

[Solution 9](#_Toc115168363)

[Appendix 10](#_Toc115168364)

[Appendix 1 10](#_Toc115168365)

[Appendix 2 12](#_Toc115168366)

[Appendix 3 14](#_Toc115168367)

[3.A – Alpha function 14](#_Toc115168368)

[3.A – AMPA 16](#_Toc115168369)

[3.A – NMDA 18](#_Toc115168370)

[3.A – GABA-A 20](#_Toc115168371)

[3.A – GABA-B 22](#_Toc115168372)

[3.B 25](#_Toc115168373)

[Appendix 4 28](#_Toc115168374)

Jinshui Zhang

# Part 1: EIAF Model

*See code in Appendix 1.*

To implement the Exponential Integrate-and-Fire Model, add an extra term in the ODE equation of potential, as

We can obtain the firing rate curve as Fig. 1, with the following parameters.

* Threshold: 30 mV
* Reset: v -> rest potential (-65 mV)
* Refractory: 0 ms

Graphical user interface, chart

Description automatically generated

Fig. 1. Exponential Integrate-And-Fire Model

However, we can observe an issue that the potential never hits the threshold at 30 mV. This is caused by the large time step. Instead of showing the jump over and down the threshold, the brian2 may just skip the time step that exceeds the threshold and to the reset action. We can observe the exact (or much closer to) threshold if the time step is reduced significantly. Fig. 2. Is obtained by setting the time step to .0001 ms, and we can see the potential getting much closer to the threshold (red line).

Chart, line chart

Description automatically generated

Fig. 2. 0.0001 ms simulation step of EIAF model.

# Part 2: Izhikevich Model

*See code in Appendix 2.*

To implement Izhikevich model, two unitless variables, v\_unitless, u\_unitless, are created to perform ODE for the calculational convenience. Then their values would be assigned to the actual u, v variables by multiplying the units mV.

Different from the previously used models, the reset process of the Izhikevich Model involves two variables, u & v. The next step would be to pick values for the parameters a, b, c and d based on the expected stimulation behavior. Fig. 3. provides the data.

Chart, scatter chart

Description automatically generated

Fig. 3. Parameter map for different neuron stimulation types.

Create a dictionary that store the parameter combination as below.,

lib\_izhik = {"RS": (0.02, 0.2, -65, 8),

"IB": (0.02, 0.2, -55, 4),

"CH": (0.02, 0.2, -50, 2.1),

"FS": (0.1, 0.2, -65, 2),

"LTS": (0.02, 0.25, -65, 2),

"RZ": (0.1, 0.26, -65, 2),

"TC": (0.02, 0.2, -65, 0.05)}

Assign values to a, b, c, d by a, b, c, d = lib\_izhik[stim\_type]

We can obtain different neuron stimulation results as Fig. 4.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

Fig. 4. Izhikevich model reproducing 7 different responses, including RS, IB, CH, FS, TC, RZ and LTS.

# Part 3: Synapses

*See code in Appendix 3.*

## A) Alpha & Biological Models of Synapses

### 1. Alpha function.

Modify the code. Create variable z to eliminate the second order ODE of g.

*dv/dt = (il +g\_ampa\*msiemens/cm\*\*2\*(0\*mV-v)+I/area )/Cm: volt*

*dg\_ampa/dt = -g\_ampa/tau\_ampa + z: 1*

*dz/dt = -z / tau\_ampa : Hz*

Then the needs to be updated by

We can obtain the dynamic as Fig. 5.

A picture containing graphical user interface

Description automatically generated

Fig. 5. Neuron behavior with synaptic model of Alpha function. The subplots are respectively 1) Pre-synaptic potential. 2) Pre-synaptic conductance. 3) Post-synaptic conductance. 4) Post-synaptic potential

We can see that two pulses are stimulated, resulting the post-synaptic conductance increased twice. This in turn causes the post-synaptic neurons to fire multiple pulses. The quantity of post-synaptic pulses is not directly related to the quantity of pre-synapse. It’s directly decided by the conductance change.

### 2. Biophysical Models

Modify the code based on the equations from lecture notes. We can obtain the 4 different biophysical synaptic models as shown in Fig. 6, including two excitatory synapses AMPA and NMDA, and two inhibitory ones GABA-A and GABA-B.

Besides the equations in lecture notes, Key information used in the program includes:

* The potential for inhibitory synapses is -80 mV: while excitatory 0 mV.
* For GABA-B model, the parameter is 100.

|  |  |
| --- | --- |
|  |  |
|  |  |

Fig. 6. Biophysical synaptic models, respectively AMPA, NMDA, GABA-A and GABA-B.

The time scale of the plot has been adjusted for different models, where AMPA and GABA-A has faster response thus a shorter time scale and constant, while NMDA and GABA-B has slower response thus a longer time scale.

These synapses all show a much faster rise speed than decaying.

GABA-B shows a much slighter influence on the post-synaptic neurons compared with the other three.

## B) Improved model with different rise / decay time constants

Adjust the time constants to approach the 4 biological models obtained in A). Meanwhile, calculate the peak time and calibrate the peak conductance. We can obtain the approximated neuron spikes and conductance dynamics as in Table I.

By manipulating time constants , we can well approximate the alpha function model and 4 biophysical models.

Table I. Differing to approximate the 4 biophysical synaptic models.

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Original** | **Approximated by model** |
| |  |  | | --- | --- | |  | 0.35 ms | |  | 0.34 ms | |  | 1.5 | |  | 0.34 ms | |  | 11.82 | | A picture containing graphical user interface  Description automatically generated | Graphical user interface  Description automatically generated with low confidence |
| |  |  | | --- | --- | |  | 6 ms | |  | 0.14 ms | |  | 0.13 | |  | 0.54 ms | |  | 0.99 | |  |  |
| |  |  | | --- | --- | |  | 150 ms | |  | 0.05 ms | |  | 0.021 | |  | 0.40 ms | |  | 0.42 | |  |  |
| |  |  | | --- | --- | |  | 5.5 ms | |  | 0.1 ms | |  | 0.51 | |  | 0.41 ms | |  | 5.53 | |  |  |
| |  |  | | --- | --- | |  | 40 ms | |  | 200 ms | |  | 2.27e-6 | |  | 80.5 ms | |  | 8.47e-8 | |  |  |

# Part 4: Refractory influence on Exponential IAF Model

While exploring the EIAF model in part 1, we set the refractory time to 0 ms. The firing rate shows a nearly proportional to the stimulus current, as in Fig. 1.

If we extend the current range, we can observe a more exponential-like trend, as shown in Fig. 7.

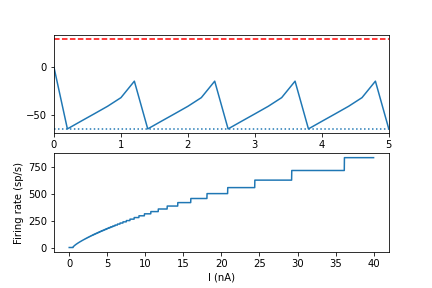


Fig. 7. EIAF firing rate curve

Another important parameter for the neurons is the refractory. When we deal with LIAF models, adding a refractory time will result in a saturation / up-limit of the firing rate.

Chart, histogram

Description automatically generated

Fig. 8. LIAF firing rate curve

However, when we do the same thing to the EIAF models, there is calculation error, as shown in Fig. 9.

Chart, line chart

Description automatically generated

Fig.9. EIAF model firing curve with 4 ms refractory time, leading to a zero-firing rate after hitting the theoretical saturation value 250 Hz.

The problematic potential waveform is picked out, showing an infinite integration, which explains why the firing rate goes to zero.

## Solution

WARNING neurongroup\_1's variable 'v' has NaN, very large values, or encountered an error in numerical integration. This is usually a sign that an unstable or invalid integration method was chosen. [brian2.groups.group.invalid\_values]

Based on the warning message, we tried different ODE solving methods, such as ‘linear’, ‘exponential\_euler’, ‘rk2’, ‘rk4’. However, they all showed the similar error, which means the error should be caused by the model itself instead of the solving algorithm.

This is caused by the definition of action during refractory. During the above model, the potential is still updating during the refractory. Therefore the voltage exceeds the reasonable range and become NaN, which is a disaster in Python.

Changing the action during the refractory can solve this problem. Instead of keep updating, we pause the integration during refractory. Adding (unless refractory) to the ODE of potential.

dv/dt = ((E\_L - v) + F\_v + R\_m \* I) / tau\_m : volt **(unless refractory)**

The modified results are given as below.

Graphical user interface

Description automatically generated

Fig. 10. No updating during refractory interval.

# Appendix

## Appendix 1

*from brian2 import \**

*num\_neurons = 75*

*duration = 3.0\*second*

*# Parameters*

*area = 20000\*umetre\*\*2*

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

*# defaultclock.dt=.001\*ms*

*defaultclock.dt=.02\*ms*

*div=defaultclock.dt*

*E\_L = -65 \* mV*

*V\_rest = E\_L.copy()*

*tau\_m = 10 \* ms*

*R\_m = 10 \* Mohm*

*Delta\_E = 5.0 \* mV*

*V\_T = -55 \* mV*

*V\_max = 30 \* mV*

*V\_th = -30 \* mV*

*eqs = """*

*dv/dt = ((E\_L - v) + F\_v + R\_m \* I) / tau\_m : volt*

*F\_v = Delta\_E \* exp((v - V\_T) / Delta\_E) : volt*

*I: amp*

*"""*

*# Threshold and refractoriness are only used for spike counting*

*P1 = NeuronGroup(num\_neurons,*

*eqs,*

*clock=Clock(defaultclock.dt),*

*threshold='v > V\_th',*

*reset='v = V\_rest',*

*# refractory=1\*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[-20]/mV)*

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

*axhline(V\_th/mV, ls='--', c='r')*

*# xlim(3.83, 3.84)*

*xlim(0, 30)*

*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()*

## Appendix 2

*from brian2 import \**

*num\_neurons = 100*

*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*

*lib\_izhik = {*

*"RS": (0.02, 0.2, -65, 8),*

*"IB": (0.02, 0.2, -55, 4),*

*"CH": (0.02, 0.2, -50, 2.1),*

*"FS": (0.1, 0.2, -65, 2),*

*"LTS": (0.02, 0.25, -65, 2),*

*"RZ": (0.1, 0.26, -65, 2),*

*"TC": (0.02, 0.2, -65, 0.05),*

*}*

*eqs = """*

*dv\_unitless/dt = ((0.04 \* (v\_unitless)\*\*2) + 5 \* (v\_unitless) + 140 - u\_unitless + I / nA) / ms : 1*

*du\_unitless/dt = a \* (b \* v\_unitless - u\_unitless) / ms : 1*

*v = v\_unitless \* mV : volt*

*u = u\_unitless \* mV : volt*

*I: amp*

*"""*

*reset\_eq = """*

*v\_unitless = c*

*u\_unitless = u\_unitless + d*

*"""*

*for stim\_type in lib\_izhik.keys():*

*a, b, c, d = lib\_izhik[stim\_type]*

*E\_rest = c \* mV*

*# Threshold and refractoriness are only used for spike counting*

*P1 = NeuronGroup(num\_neurons,*

*eqs,*

*clock=Clock(defaultclock.dt),*

*threshold='v > 50\*mV',*

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

*reset=reset\_eq,*

*# refractory=1\*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)*

*monitor3=StateMonitor(P1, ('u'), record=True)*

*net = Network(P1, monitor, monitor2)*

*net.run(duration)*

*figure(1)*

*subplot(2,1,1)*

*title(stim\_type)*

*plot(monitor2.t/ms, monitor2.v[-1]/mV)*

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

*xlim(0, 1000)*

*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"])*

*savefig("img/{}.png".format(stim\_type))*

*show()*

## Appendix 3

### 3.A – Alpha function

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Mon Jan 27 15:30:32 2020

@author: chenriq

"""

from brian2 import \*

defaultclock.dt=.01\*ms

num\_neurons = 2

duration = 2\*second

# Parameters

area = 20000\*umetre\*\*2

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

El = -60\*mV

gl = 0.7\*msiemens/cm\*\*2

tau\_ampa=0.3\*ms

g\_synpk=1.5

g\_synmaxval= g\_synpk / (tau\_ampa / ms \* exp(-1))

eqs\_il = '''

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

'''

eqs = '''

dv/dt = (il +g\_ampa\*msiemens/cm\*\*2\*(0\*mV-v)+I/area )/Cm: volt

dg\_ampa/dt = -g\_ampa/tau\_ampa + z: 1

dz/dt = -z / tau\_ampa : Hz

I : amp

'''

eqs += (eqs\_il)

# Threshold and refractoriness are only used for spike counting

group = NeuronGroup(num\_neurons, eqs, clock=Clock(defaultclock.dt),threshold='v > -45\*mV',reset='v = -60\*mV', method='euler')

group.v = El

Sr = Synapses(group, group, clock=group.clock,model='''

g\_synmax:1 ''',

on\_pre='''

z += g\_synmax / ms

''')

g\_synmax = g\_synmaxval

Sr.connect(i=[0],j=[1])

Sr.g\_synmax=g\_synmaxval

Sr.delay=0\*ms #introduces a fixed delay between the firing of the pre cell and the postsynaptic response

monitor2=StateMonitor(group,('v', 'g\_ampa'),record=True)

group.I[0] = 0\*nA

group.I[1] = 0\*nA

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

group.I[0] = 8\*nA

group.I[1] = 0\*nA

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

group.I[0] = 0\*nA

group.I[1] = 0\*nA

run(10.0\*ms)

figure(3)

subplot(4,1,1)

title("alpha\_3a")

plot(monitor2.t/ms, monitor2.v[0]/mV)

subplot(4,1,2)

plot(monitor2.t/ms, monitor2.g\_ampa[0])

subplot(4,1,3)

plot(monitor2.t/ms, monitor2.g\_ampa[1],'r')

subplot(4,1,4)

plot(monitor2.t/ms, monitor2.v[1]/mV,'r')

savefig("img/alpha\_3a.png")

### 3.A – AMPA

*#!/usr/bin/env python3*

*# -\*- coding: utf-8 -\*-*

*"""*

*Created on Mon Jan 27 15:30:32 2020*

*@author: chenriq*

*"""*

*from brian2 import \**

*defaultclock.dt=.01\*ms*

*num\_neurons = 2*

*duration = 2\*second*

*# Parameters*

*area = 20000\*umetre\*\*2*

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

*El = -60\*mV*

*gl = 0.7\*msiemens/cm\*\*2*

*tau\_ampa=0.3\*ms*

*g\_synpk=1.5*

*eqs\_il = '''*

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

*'''*

*eqs = '''*

*dv/dt = (il +GAMPA\*g\_synpk\*msiemens/cm\*\*2\*(0\*mV-v)+I/area )/Cm: volt*

*I : amp*

*GAMPA:1*

*'''*

*eqs += (eqs\_il)*

*# Threshold and refractoriness are only used for spike counting*

*group = NeuronGroup(num\_neurons, eqs, clock=Clock(defaultclock.dt),threshold='v > -45\*mV',reset='v = -60\*mV', method='euler')*

*group.v = El*

*Sr = Synapses(group, group, clock=group.clock,method='euler',model='''*

*Trpre=.25\*(tanh((t/ms-tspike/ms)/.005)-tanh((t/ms-(tspike/ms +.3))/.005)):1*

*aAMPA=0.94/ms :Hz*

*bAMPA=0.18/ms: Hz*

*tspike: second*

*dsAMPA\_syn/dt = aAMPA\*Trpre\*(1-sAMPA\_syn)-bAMPA\*sAMPA\_syn:1*

*GAMPA\_post=sAMPA\_syn : 1 (summed)''',on\_pre='''*

*tspike=t''')*

*Sr.connect(i=[0],j=[1])*

*Sr.tspike[0]=-.5\*ms # needed to get rid of spurious first tspike*

*Sr.delay=0\*ms #introduces a fixed delay between the firing of the pre cell and the postsynaptic response*

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

*group.I[0] = 0\*nA*

*group.I[1] = 0\*nA*

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

*group.I[0] = 5\*nA*

*group.I[1] = 0\*nA*

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

*group.I[0] = 0\*nA*

*group.I[1] = 0\*nA*

*run(10.0\*ms)*

*figure(3)*

*subplot(4,1,1)*

*title("AMPA")*

*plot(monitor2.t/ms, monitor2.v[0]/mV)*

*subplot(4,1,2)*

*plot(monitor2.t/ms, monitor2.GAMPA[0])*

*subplot(4,1,3)*

*plot(monitor2.t/ms, monitor2.GAMPA[1],'r')*

*subplot(4,1,4)*

*plot(monitor2.t/ms, monitor2.v[1]/mV,'r')*

*savefig("img/ampa\_3a.png")*

*print(max(monitor2.GAMPA[1]))*

### 3.A – NMDA

*#!/usr/bin/env python3*

*# -\*- coding: utf-8 -\*-*

*"""*

*Created on Mon Jan 27 15:30:32 2020*

*@author: chenriq*

*"""*

*from brian2 import \**

*defaultclock.dt=.01\*ms*

*num\_neurons = 2*

*duration = 2\*second*

*# Parameters*

*area = 20000\*umetre\*\*2*

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

*El = -60\*mV*

*gl = 0.7\*msiemens/cm\*\*2*

*tau\_ampa=0.3\*ms*

*g\_synpk=1.5*

*eqs\_il = '''*

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

*'''*

*eqs = '''*

*dv/dt = (il +GNMDA\*g\_synpk\*msiemens/cm\*\*2\*(0\*mV-v)+I/area )/Cm: volt*

*I : amp*

*GNMDA:1*

*'''*

*eqs += (eqs\_il)*

*# Threshold and refractoriness are only used for spike counting*

*group = NeuronGroup(num\_neurons, eqs, clock=Clock(defaultclock.dt),threshold='v > -45\*mV',reset='v = -60\*mV', method='euler')*

*group.v = El*

*Sr = Synapses(group, group, clock=group.clock,method='euler',model='''*

*Trpre=.25\*(tanh((t/ms-tspike/ms)/.005)-tanh((t/ms-(tspike/ms +.3))/.005)):1*

*aNMDA=0.072/ms :Hz*

*bNMDA=0.0066/ms: Hz*

*tspike: second*

*dsNMDA\_syn/dt = aNMDA\*Trpre\*(1-sNMDA\_syn)-bNMDA\*sNMDA\_syn:1*

*GNMDA\_post=sNMDA\_syn : 1 (summed)''',on\_pre='''*

*tspike=t''')*

*Sr.connect(i=[0],j=[1])*

*Sr.tspike[0]=-.5\*ms # needed to get rid of spurious first tspike*

*Sr.delay=0\*ms #introduces a fixed delay between the firing of the pre cell and the postsynaptic response*

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

*group.I[0] = 0\*nA*

*group.I[1] = 0\*nA*

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

*group.I[0] = 8\*nA*

*group.I[1] = 0\*nA*

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

*group.I[0] = 0\*nA*

*group.I[1] = 0\*nA*

*run(300.0\*ms)*

*figure(3)*

*subplot(4,1,1)*

*title("NMDA")*

*plot(monitor2.t/ms, monitor2.v[0]/mV)*

*subplot(4,1,2)*

*plot(monitor2.t/ms, monitor2.GNMDA[0])*

*subplot(4,1,3)*

*plot(monitor2.t/ms, monitor2.GNMDA[1],'r')*

*subplot(4,1,4)*

*plot(monitor2.t/ms, monitor2.v[1]/mV,'r')*

*savefig("img/nmda\_3a.png")*

*print(max(monitor2.GNMDA[1]))*

### 3.A – GABA-A

*#!/usr/bin/env python3*

*# -\*- coding: utf-8 -\*-*

*"""*

*Created on Mon Jan 27 15:30:32 2020*

*@author: chenriq*

*"""*

*from brian2 import \**

*defaultclock.dt=.01\*ms*

*num\_neurons = 2*

*duration = 2\*second*

*# Parameters*

*area = 20000\*umetre\*\*2*

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

*El = -60\*mV*

*gl = 0.7\*msiemens/cm\*\*2*

*tau\_ampa=0.3\*ms*

*g\_synpk=1.5*

*eqs\_il = '''*

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

*'''*

*eqs = '''*

*dv/dt = (il + GGABAA\*g\_synpk\*msiemens/cm\*\*2\*(-80\*mV-v)+I/area )/Cm: volt*

*I : amp*

*GGABAA:1*

*'''*

*eqs += (eqs\_il)*

*# Threshold and refractoriness are only used for spike counting*

*group = NeuronGroup(num\_neurons, eqs, clock=Clock(defaultclock.dt),threshold='v > -45\*mV',reset='v = -60\*mV', method='euler')*

*group.v = El*

*Sr = Synapses(group, group, clock=group.clock,method='euler',model='''*

*Trpre=.25\*(tanh((t/ms-tspike/ms)/.005)-tanh((t/ms-(tspike/ms +.3))/.005)):1*

*aGABAA=5/ms :Hz*

*bGABAA=0.18/ms: Hz*

*tspike: second*

*dsGABAA\_syn/dt = aGABAA\*Trpre\*(1-sGABAA\_syn)-bGABAA\*sGABAA\_syn:1*

*GGABAA\_post=sGABAA\_syn : 1 (summed)''',on\_pre='''*

*tspike=t''')*

*Sr.connect(i=[0],j=[1])*

*Sr.tspike[0]=-.5\*ms # needed to get rid of spurious first tspike*

*Sr.delay=0\*ms #introduces a fixed delay between the firing of the pre cell and the postsynaptic response*

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

*group.I[0] = 0\*nA*

*group.I[1] = 0\*nA*

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

*group.I[0] = 5\*nA*

*group.I[1] = 0\*nA*

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

*group.I[0] = 0\*nA*

*group.I[1] = 0\*nA*

*run(20.0\*ms)*

*figure(3)*

*subplot(4,1,1)*

*title("GABA-A")*

*plot(monitor2.t/ms, monitor2.v[0]/mV)*

*subplot(4,1,2)*

*plot(monitor2.t/ms, monitor2.GGABAA[0])*

*subplot(4,1,3)*

*plot(monitor2.t/ms, monitor2.GGABAA[1],'r')*

*subplot(4,1,4)*

*# ylim(-200, 100)*

*plot(monitor2.t/ms, monitor2.v[1]/mV,'r')*

*savefig("img/gabaa\_3a.png")*

*print(max(monitor2.GGABAA[1]))*

### 3.A – GABA-B

*#!/usr/bin/env python3*

*# -\*- coding: utf-8 -\*-*

*"""*

*Created on Mon Jan 27 15:30:32 2020*

*@author: chenriq*

*"""*

*from brian2 import \**

*defaultclock.dt=.01\*ms*

*num\_neurons = 2*

*duration = 2\*second*

*# Parameters*

*area = 20000\*umetre\*\*2*

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

*El = -60\*mV*

*gl = 0.7\*msiemens/cm\*\*2*

*tau\_ampa=0.3\*ms*

*g\_synpk=1.5*

*eqs\_il = '''*

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

*'''*

*eqs = '''*

*dv/dt = (il +GGABAB\*g\_synpk\*msiemens/cm\*\*2\*(-80\*mV-v)+I/area )/Cm: volt*

*I : amp*

*GGABAB:1*

*'''*

*eqs += (eqs\_il)*

*# Threshold and refractoriness are only used for spike counting*

*group = NeuronGroup(num\_neurons, eqs, clock=Clock(defaultclock.dt),threshold='v > -45\*mV',reset='v = -60\*mV', method='euler')*

*group.v = El*

*Sr = Synapses(group, group, clock=group.clock,method='euler',model='''*

*Trpre=.25\*(tanh((t/ms-tspike/ms)/.005)-tanh((t/ms-(tspike/ms +.3))/.005)):1*

*aGABAB=0.09/ms :Hz*

*bGABAB\_p=0.0012/ms: Hz*

*K3=0.18/ms : Hz*

*K4=0.034/ms : Hz*

*tspike: second*

*dsGABAB\_syn/dt = K3 \* rGABAB - K4 \* sGABAB\_syn : 1*

*drGABAB/dt = aGABAB\*Trpre\*(1-rGABAB)-bGABAB\_p\*rGABAB:1*

*GGABAB\_post=sGABAB\_syn \*\* 4 / (sGABAB\_syn \*\* 4 + 100) : 1 (summed)*

*''',*

*on\_pre='''*

*tspike=t*

*''')*

*Sr.connect(i=[0],j=[1])*

*Sr.tspike[0]=-.5\*ms # needed to get rid of spurious first tspike*

*Sr.delay=0\*ms #introduces a fixed delay between the firing of the pre cell and the postsynaptic response*

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

*group.I[0] = 0\*nA*

*group.I[1] = 0\*nA*

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

*group.I[0] = 8\*nA*

*group.I[1] = 0\*nA*

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

*group.I[0] = 0\*nA*

*group.I[1] = 0\*nA*

*run(1000.0\*ms)*

*figure(3)*

*subplot(4,1,1)*

*title("GABA-B")*

*plot(monitor2.t/ms, monitor2.v[0]/mV)*

*subplot(4,1,2)*

*plot(monitor2.t/ms, monitor2.GGABAB[0])*

*subplot(4,1,3)*

*# ylim(0, 1e-5)*

*plot(monitor2.t/ms, monitor2.GGABAB[1],'r')*

*subplot(4,1,4)*

*plot(monitor2.t/ms, monitor2.v[1]/mV,'r')*

*savefig("img/gabab\_3a.png")*

*print(max(monitor2.GGABAB[1]))*

### 3.B

*#!/usr/bin/env python3*

*# -\*- coding: utf-8 -\*-*

*"""*

*Created on Mon Jan 27 15:30:32 2020*

*@author: chenriq*

*"""*

*from brian2 import \**

*import math*

*defaultclock.dt=.01\*ms*

*num\_neurons = 2*

*duration = 2\*second*

*# Parameters*

*area = 20000\*umetre\*\*2*

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

*El = -60\*mV*

*gl = 0.7\*msiemens/cm\*\*2*

*# AMPA*

*# tau\_rise=6\*ms*

*# tau\_decay=.1\*ms*

*# g\_synpk=0.12627*

*# NMDA*

*# tau\_decay=0.05\*ms*

*# tau\_rise=150\*ms*

*# g\_synpk=0.02095*

*# GABA-A*

*# tau\_rise=5.5\*ms*

*# tau\_decay=.1\*ms*

*# g\_synpk=0.5133*

*# GABA-B*

*# tau\_decay=90\*ms*

*# tau\_rise=110\*ms*

*# g\_synpk=2.2659371976631717e-06*

*peak\_time = (tau\_rise \* tau\_rise / (tau\_decay - tau\_rise)) \* math.log(tau\_decay / tau\_rise)*

*print(peak\_time)*

*g\_synmaxval=g\_synpk / ((tau\_rise \* tau\_rise / (tau\_decay - tau\_rise)) \* (exp(-peak\_time / tau\_decay) - exp(-peak\_time / tau\_rise)))*

*print(g\_synmaxval)*

*eqs\_il = '''*

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

*'''*

*eqs = '''*

*dv/dt = (il +g\*msiemens/cm\*\*2\*(-80\*mV-v)+I/area )/Cm: volt*

*dg/dt = -g/tau\_decay + z: 1*

*dz/dt = -z / tau\_rise : Hz*

*I : amp*

*'''*

*eqs += (eqs\_il)*

*# Threshold and refractoriness are only used for spike counting*

*group = NeuronGroup(num\_neurons, eqs, clock=Clock(defaultclock.dt),threshold='v > -45\*mV',reset='v = -60\*mV', method='euler')*

*group.v = El*

*Sr = Synapses(group, group, clock=group.clock,model='''*

*g\_synmax = g\_synmaxval \* ms : 1*

*''',*

*on\_pre='''*

*z += g\_synmax / ms*

*''')*

*Sr.connect(i=[0],j=[1])*

*Sr.delay=0\*ms #introduces a fixed delay between the firing of the pre cell and the postsynaptic response*

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

*group.I[0] = 0\*nA*

*group.I[1] = 0\*nA*

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

*group.I[0] = 5\*nA*

*group.I[1] = 0\*nA*

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

*group.I[0] = 0\*nA*

*group.I[1] = 0\*nA*

*# run(10.0\*ms)*

*# run(20.0\*ms)*

*run(1000.0\*ms)*

*figure(3)*

*subplot(4,1,1)*

*plot(monitor2.t/ms, monitor2.v[0]/mV)*

*subplot(4,1,2)*

*plot(monitor2.t/ms, monitor2.g[0])*

*subplot(4,1,3)*

*plot(monitor2.t/ms, monitor2.g[1],'b')*

*subplot(4,1,4)*

*plot(monitor2.t/ms, monitor2.v[1]/mV,'b')*

*savefig("img/2tau.png")*

## Appendix 4

*from brian2 import \**

*num\_neurons = 100*

*duration = 3.0\*second*

*# duration = 0.1\*second*

*# Parameters*

*area = 20000\*umetre\*\*2*

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

*# defaultclock.dt=.001\*ms*

*defaultclock.dt=.02\*ms*

*div=defaultclock.dt*

*E\_L = -65 \* mV*

*V\_rest = E\_L.copy()*

*tau\_m = 10 \* ms*

*R\_m = 10 \* Mohm*

*Delta\_E = 5.0 \* mV*

*V\_T = -55 \* mV*

*V\_max = 30 \* mV*

*V\_th = 30 \* mV*

*eqs = """*

*dv/dt = ((E\_L - v) + F\_v + R\_m \* I) / tau\_m : volt (unless refractory)*

*F\_v = Delta\_E \* exp((v - V\_T) / Delta\_E) : volt*

*I: amp*

*"""*

*# Threshold and refractoriness are only used for spike counting*

*P1 = NeuronGroup(num\_neurons,*

*eqs,*

*clock=Clock(defaultclock.dt),*

*threshold='v > V\_th',*

*reset='v = V\_rest',*

*refractory=10\*ms,*

*# refractory='v>1000\*mV',*

*method='euler',*

*# method='exponential\_euler',*

*# method='linear',*

*# method='rk2',*

*# method='heun',*

*# method='milstein',*

*# method='rk4',*

*)*

*# 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[-1]/mV)*

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

*axhline(V\_th/mV, ls='--', c='r')*

*# xlim(3.83, 3.84)*

*xlim(0, 100)*

*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"])*

*savefig("img/eiaf\_result.png")*

*show()*