BME 503: Exploration 3

Contents

[Part 1: Izhikevich Model with Alpha Function Synapses 2](#_Toc115957537)

[Q1 2](#_Toc115957538)

[Q2 4](#_Toc115957539)

[Q3 5](#_Toc115957540)

[Q4 6](#_Toc115957541)

[Q5 6](#_Toc115957542)

[Part 2: Braitenbug – Coward Type 10](#_Toc115957543)

[Appendix 12](#_Toc115957544)

[Appendix 1 12](#_Toc115957545)

[Appendix 1.1 12](#_Toc115957546)

[Appendix 1.2 24](#_Toc115957547)

[Appendix 1.3 24](#_Toc115957548)

[Appendix 1.4 24](#_Toc115957549)

[Appendix 1.5 24](#_Toc115957550)

[Appendix 2 24](#_Toc115957551)

[Appendix 2.1: Create bug and move bug 24](#_Toc115957552)

[Appendix 2.2: Show BUG 24](#_Toc115957553)

Jinshui Zhang

# Part 1: Izhikevich Model with Alpha Function Synapses

*See code in Appendix 1.*

## Q1

Connect neuron 0 to neuron 1 and neuron 1 to neuron 2 with excitatory connections with a synaptic time constant of 5ms and a reversal potential of 0. Establish a steady state with no input current to any neurons for 200 ms. Apply a suprathreshold stimulus for 20 ms neuron 0. Show how the response changes in neurons 1 and 2 as you change from 0.01 to 0.08.

The Model is built as follows:

1. Build the basic Izhikevich model with unitless variable .
2. Add the synaptic current in the equation’s right side, as .
3. Initialize the variables as

First run the steady state with no current for 200 ms, we can obtain the potential and conductance as Fig. 1.

|  |  |
| --- | --- |
|  |  |
| Fig. 1. Steady state of Izhikevich-Alpha Model | Fig. 2. Dynamics before steady state if variables not initialized properly. |

It should be noted that the variables must be correctly initialized, otherwise the steady state would go through a dynamic process, as shown in Fig. 2.

Apply a stimulation current of 8 nA for 20 ms, we can obtain the results as Fig. 3.

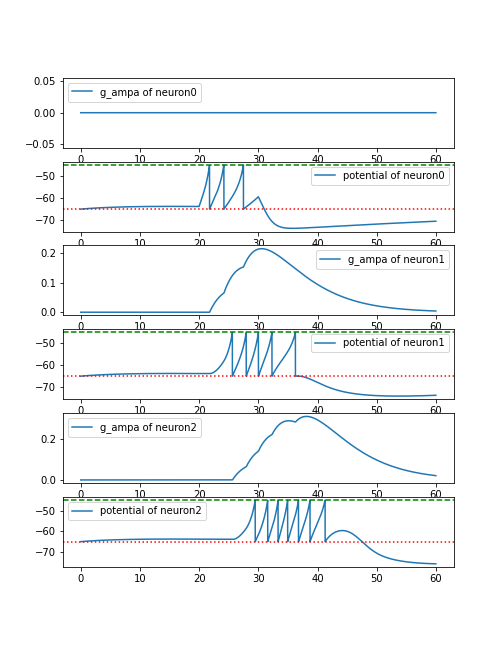


Fig. 3. Conductance and potential response when apply a 8 nA current to Neuron 0.

The neuron 0 is stimulated and 3 spikes are fired. The pulses increase the conductance of neuron 1 and induced 5 spikes of neuron 1. And the spikes of neuron 1 excite even more spiking of neuron 2, reaching 7 pulses in total. Some of them happens even after the stimulation current is cut out.

Change the value of from 0.01 to 0.08, we can obtain the results as Fig. 4.

|  |  |  |  |
| --- | --- | --- | --- |
| Graphical user interface  Description automatically generated |  |  |  |
|  |  |  |  |

Fig. 4. Stimulation with different values of , respectively 0.01, 0.02, …, 0.08.

The spikes are summarized as Fig. 5.

Chart, line chart

Description automatically generated

Fig. 5. Change of spike quantity with the g peak value.

## Q2

Incorporate synaptic delays to the case above of 5ms. How does the response change?

Apply a 5 ms delay on the synaptic delays, we can obtain the results as Fig. 6.

Graphical user interface

Description automatically generated

Fig. 6. Stimulation results after adding a 5 ms delay.

The results obtained here only differs the Q (1) on the performance of neuron 1&2. Neuron 1’s conductance and potential change is delayed by 5ms with the same shape and spiking rate as Q (1). Neuron 2 is delayed by 10 ms.

Neuron 0 is not affected.

This result makes sense because the whole system is an open-loop one, the later neurons won’t have influence on the former ones. Nor is there synaptic connections between the same-level neurons.

If the neuron circuit is a closed loop, then we would expect the delay has significant influence.

## Q3

Connect neuron 0 to neuron 1 with inhibitory connections with a synaptic time constant of 5ms and a reversal potential of -80. Do not connect any cell to neuron 2. Reach steady state at 200ms with no stimulus and apply the same suprathreshold input currents to each cell. Here neuron 0 and 2 should generate the same response. What happens to neuron 1 as is varied from 0.02 to 0.12.

Connect neuron 0 to neuron 1. And neuron 2 is connected to nothing. The reversal potential is set to -80 mV, which corresponds to inhibitive synapses.

Apply a 200 ms steady state with no current to any neuron. Then apply a stimulation current of 8 nA to all the neurons. For different values of We can obtain the results as Fig. 7.

|  |  |  |
| --- | --- | --- |
| Graphical user interface, application  Description automatically generated |  |  |
|  |  |  |

Fig. 7. Stimulation results with different values of , respectively 0.02, 0.04, …, 0.12.

The figures show a trend that with the increasing, the neuron 0 & 2 stays unchanged, and their potentials are always consistent with each other.

Neuron 1 is the only one that has been affected. The larger the conductance is, the fewer the spikes are.

## Q4

Incorporate synaptic delays to the case above using a 5ms time constant. How does the response change?

Apply a 5 ms delay to the synapses, we can obtain the results as Fig. 7.

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

Fig. 8. Stimulation results with 5 ms delay.

The results obtained here only differs the Q (3) on the performance of neuron 1. Neuron 1’s conductance and potential change is delayed by 5ms with the same shape and spiking rate as Q (3). Neurons 0 & 2 are not affected.

The reason behind the phenomenon is still the open-loop structure of the system.

## Q5

Design several interesting circuits using excitatory AND inhibitory connections with 4 neurons. Note that one cell type should be able to produce only one type of synapse (inhibitory or excitatory) but a cell can receive both types. Explain the output based on your design.

In this part, we are building a 4-neuron circuit. This is a warming-up for the incoming project 1.

Background: Imagine a control system of 1 sensor and 3 executors (concepts like motors). The 3 executors would be identical submodules that can perform the same function.

The sensor acts as a commander for the system and decides how many submodules should work. However, when one submodule starts to work, the other ones should not join the task until the sensor increased the excitation level.

Therefore, when one submodule starts to work or be stimulated, it should try to inhibit the other modules. The most direct method to inhibit the others would be connecting inhibitive synapses to all the other SM neurons, as shown in Fig. 9.

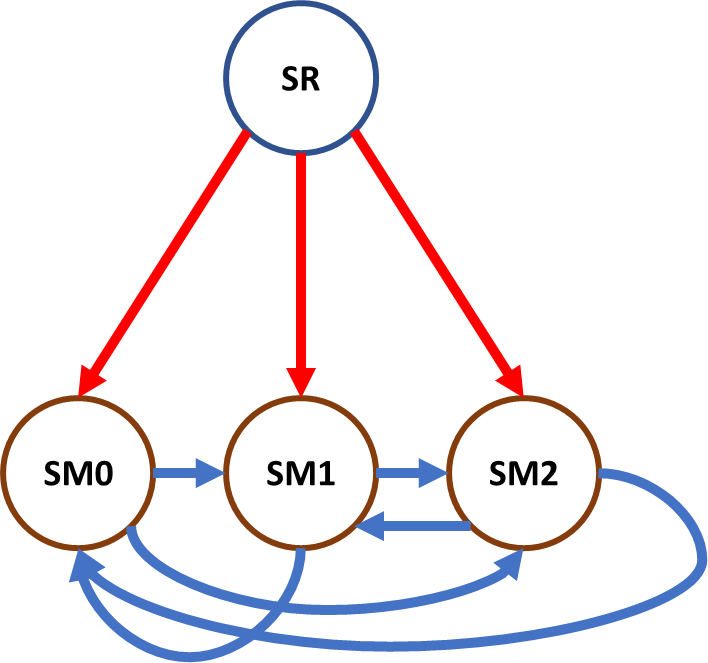
****

Fig. 9. Inhibitive synapses only among SM neurons.

This structure may look acceptable for 3-SM systems. However, the synapse quantity increases crazily as the number of neurons increasing.

where n is the number of submodules. The synapses would be physically a signal wire if we try to build this control system. We need to reduce the number of synapses.

There might be another method to realize the inhibition of the SMx to the others. Instead of inhibiting all other SM neurons, we can also let the SMx inhibit the SR neuron, as in Fig. 10.

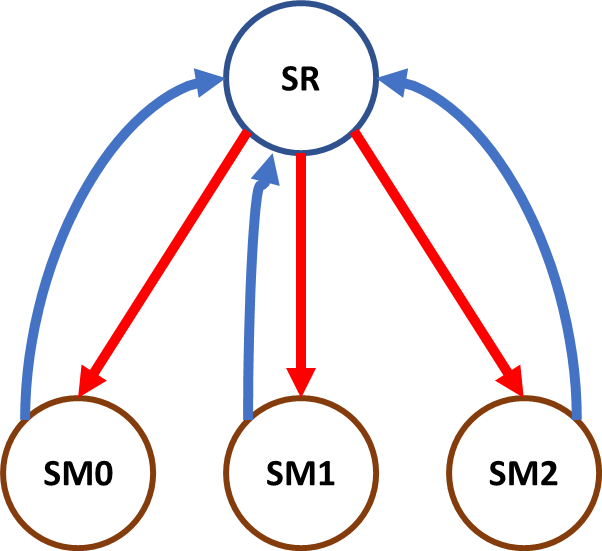


Fig. 10. Inhibitory synapses only connect the SR.

In this way, the number of synapses becomes

Next, we would try to make the two models realize the same response facing the same input excitation.

It should be noted that the waveforms are not necessarily required to be the same. But at least we should expect the spiking rate of SM neurons if the second structure can replace the first one.

After tuning the parameters including time constants and , concerning the performance of SM neurons, models 1 and 2 show quite different features. The delay time is set 0 ms. All Izhikevich parameters are consistent with ‘LTS’ mode.

Model 1 gives the response as Fig. 11. SM#1 neuron is zoomed for better details presentation. SM 0, 1 &2 have identical performance. Model 2 gives the response as Fig. 12. SM#1 neuron is zoomed for better details presentation. SM 0, 1 &2 have identical performance.

Model 1 finally gets into a steady state where neurons SR & SMs all possess a regular spiking feature.

Model 2 however finally shows a Chattering feature.

For our specific goal that whether we can make the two model have the same spiking rate, the answer is YES. The two models **CAN** indeed possess the same spiking rate by adjusting the parameters.

More features will be added, designed, and explored after gaining more knowledge on how to simulate the ON/OFF switch performance of neurons.

|  |  |
| --- | --- |
|  |  |
| Fig. 11. Model 1 response. | Fig. 12. Model 2 response. |

# Part 2: Braitenbug – Coward Type

*See code in Appendix 2*

**Video link here.** [***https://youtu.be/rrpAFQ-\_QEA***](https://youtu.be/rrpAFQ-_QEA%20)

Build a Braitenbug model with Izhikevich neural model and exponential synaptic model following the structure provided in Fig. 13.

Diagram

Description automatically generated

Fig. 13. Topology of coward Braitenbug neural circuit.

Since the exponential function is used to describe the synapses, variable that has been used in Alpha function needs to be removed. The key ODE of potential and the synaptic conductance becomes

The movement ODEs for the motors are

It should be noted that the variable is not used in the motor ODE. Instead, the width of the bug is used, which should be equivalent.

During the @network\_operation, the location of the bug is updated by the motor matrix.

Finally, we can create a Braitenbug that always tries to avoid the target.

See the video here. [*https://youtu.be/rrpAFQ-\_QEA*](https://youtu.be/rrpAFQ-_QEA%20)

During the 1 minute of moving, the bug has **NEVER** entered the circle region that centered the target and radiate of 4.

# Appendix

## Appendix 1

### Appendix 1.1

from brian2 import \*

defaultclock.dt=.01\*ms

num\_neurons = 3

duration = 2\*second

# Parameters

area = 20000\*umetre\*\*2

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

El = -60\*mV

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

tau\_ampa=5\*ms

g\_synpk=1.5

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

print("g\_synmax = {}".format(g\_synmaxval))

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),

}

a, b, c, d = lib\_izhik['LTS']

E\_rest = c \* mV

eqs = """

dv\_unitless/dt = ((0.04 \* (v\_unitless)\*\*2) + 5 \* (v\_unitless) + 140 - u\_unitless + I / nA + g\_ampa\*msiemens/cm\*\*2\*(0\*mV - v) / Cm / mV \* ms) / ms : 1

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

v = v\_unitless \* mV : volt

u = u\_unitless \* mV : volt

g\_ampa\_syn : 1

I: amp

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

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

"""

reset\_eq = """

v\_unitless = c

u\_unitless = u\_unitless + d

"""

V\_th = -45 \* mV

# Threshold and refractoriness are only used for spike counting

group = NeuronGroup(num\_neurons,

eqs,

clock=Clock(defaultclock.dt),

# threshold='v > -45\*mV',

threshold='v > V\_th',

reset=reset\_eq,

method='euler')

# group.v\_unitless = E\_rest / mV

# group.u\_unitless = E\_rest \* b / mV

Sr = Synapses(group,

group,

clock=group.clock,

model='''

g\_synmax : 1

''',

on\_pre='''

z += g\_synmax / ms

''')

Sr.connect(i=[0, 1],j=[1, 2])

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', 'z'),record=True)

group.I[0] = 0\*nA

group.I[1] = 0\*nA

run(200.0\*ms)

# group.I[0] = 15\*nA

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

# run(10\*ms, report='text')

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

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

# run(10.0\*ms)

figure(figsize=(7,9))

subplot(6,1,1)

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

legend(["g\_ampa"])

subplot(6,1,2)

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

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

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

legend(["potential"])

subplot(6,1,3)

plot(monitor2.t/ms, monitor2.g\_ampa[1])

legend(["g\_ampa"])

subplot(6,1,4)

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

legend(["potential"])

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

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

subplot(6,1,5)

plot(monitor2.t/ms, monitor2.g\_ampa[2])

legend(["g\_ampa"])

subplot(6,1,6)

plot(monitor2.t/ms, monitor2.v[2]/mV)

legend(["potential"])

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

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

print(monitor2.v[2][-1])

savefig("img/P1Q1steady\_state")

from brian2 import \*

defaultclock.dt=.01\*ms

num\_neurons = 3

duration = 2\*second

# Parameters

area = 20000\*umetre\*\*2

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

El = -60\*mV

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

tau\_ampa=5\*ms

g\_synpk=0.08

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

print("g\_synmax = {}".format(g\_synmaxval))

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),

}

a, b, c, d = lib\_izhik['LTS']

E\_rest = c \* mV

eqs = """

dv\_unitless/dt = ((0.04 \* (v\_unitless)\*\*2) + 5 \* (v\_unitless) + 140 - u\_unitless + I / nA + g\_ampa\*msiemens/cm\*\*2\*(0\*mV - v) / Cm / mV \* ms) / ms : 1

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

v = v\_unitless \* mV : volt

u = u\_unitless \* mV : volt

g\_ampa\_syn : 1

I: amp

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

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

"""

reset\_eq = """

v\_unitless = c

u\_unitless = u\_unitless + d

"""

V\_th = -45 \* mV

# Threshold and refractoriness are only used for spike counting

group = NeuronGroup(num\_neurons,

eqs,

clock=Clock(defaultclock.dt),

# threshold='v > -45\*mV',

threshold='v > V\_th',

reset=reset\_eq,

method='euler')

group.v\_unitless = E\_rest / mV

group.u\_unitless = E\_rest \* b / mV

Sr = Synapses(group,

group,

clock=group.clock,

model='''

g\_synmax : 1

''',

on\_pre='''

z += g\_synmax / ms

''')

Sr.connect(i=[0, 1],j=[1, 2])

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', 'z'),record=True)

group.I[0] = 0\*nA

group.I[1] = 0\*nA

group.I[2] = 0\*nA

run(20.0\*ms)

group.I[0] = 8\*nA

group.I[1] = 0\*nA

group.I[2] = 0\*nA

run(10\*ms, report='text')

group.I[0] = 0\*nA

group.I[1] = 0\*nA

group.I[2] = 0\*nA

run(30.0\*ms)

figure(figsize=(7,9))

subplot(6,1,1)

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

legend(["g\_ampa of neuron0"], loc="upper left")

subplot(6,1,2)

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

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

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

legend(["potential of neuron0"])

subplot(6,1,3)

plot(monitor2.t/ms, monitor2.g\_ampa[1])

legend(["g\_ampa of neuron1"])

subplot(6,1,4)

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

legend(["potential of neuron1"])

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

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

subplot(6,1,5)

plot(monitor2.t/ms, monitor2.g\_ampa[2])

legend(["g\_ampa of neuron2"])

subplot(6,1,6)

plot(monitor2.t/ms, monitor2.v[2]/mV)

legend(["potential of neuron2"])

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

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

print(monitor2.v[2][-1])

savefig("img/P1Q1\_stimulated.png")

from brian2 import \*

defaultclock.dt=.01\*ms

num\_neurons = 3

duration = 2\*second

# Parameters

area = 20000\*umetre\*\*2

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

El = -60\*mV

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

tau\_ampa=5\*ms

g\_synpk=1.5

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

print("g\_synmax = {}".format(g\_synmaxval))

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),

}

a, b, c, d = lib\_izhik['LTS']

E\_rest = c \* mV

eqs = """

dv\_unitless/dt = ((0.04 \* (v\_unitless)\*\*2) + 5 \* (v\_unitless) + 140 - u\_unitless + I / nA + g\_ampa\*msiemens/cm\*\*2\*(0\*mV - v) / Cm / mV \* ms) / ms : 1

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

v = v\_unitless \* mV : volt

u = u\_unitless \* mV : volt

g\_ampa\_syn : 1

I: amp

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

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

"""

reset\_eq = """

v\_unitless = c

u\_unitless = u\_unitless + d

"""

V\_th = -45 \* mV

# Threshold and refractoriness are only used for spike counting

# Change the value of gpeak

g\_list = []

neuron\_spikecounter0 = []

neuron\_spikecounter1 = []

neuron\_spikecounter2 = []

for i in range(8):

group = NeuronGroup(num\_neurons,

eqs,

clock=Clock(defaultclock.dt),

# threshold='v > -45\*mV',

threshold='v > V\_th',

reset=reset\_eq,

method='euler')

group.v\_unitless = E\_rest / mV

group.u\_unitless = E\_rest \* b / mV

Sr = Synapses(group,

group,

clock=group.clock,

model='''

g\_synmax : 1

''',

on\_pre='''

z += g\_synmax / ms

''')

Sr.connect(i=[0, 1],j=[1, 2])

g\_synpk=0.01 \* (i + 1)

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

Sr.g\_synmax=g\_synmaxval

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

spikecounter = SpikeMonitor(group)

group.I[0] = 0\*nA

run(20.0\*ms)

group.I[0] = 8\*nA

run(20\*ms)

group.I[0] = 0\*nA

run(30.0\*ms)

figure(figsize=(7,9))

subplot(6,1,1)

title("Response with g peak at {}".format(g\_synpk))

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

legend(["g\_ampa of neuron0"], loc="upper left")

subplot(6,1,2)

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

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

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

legend(["potential of neuron0"])

subplot(6,1,3)

plot(monitor2.t/ms, monitor2.g\_ampa[1])

legend(["g\_ampa of neuron1"])

subplot(6,1,4)

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

legend(["potential of neuron1"])

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

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

subplot(6,1,5)

plot(monitor2.t/ms, monitor2.g\_ampa[2])

legend(["g\_ampa of neuron2"])

subplot(6,1,6)

plot(monitor2.t/ms, monitor2.v[2]/mV)

legend(["potential of neuron2"])

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

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

g\_list.append(g\_synpk)

neuron\_spikecounter0.append(spikecounter.count[0])

neuron\_spikecounter1.append(spikecounter.count[1])

neuron\_spikecounter2.append(spikecounter.count[2])

savefig("img/P1Q1\_stimulated\_w\_g{}.png".format(i))

### Appendix 1.2

from brian2 import \*

defaultclock.dt=.01\*ms

num\_neurons = 3

duration = 2\*second

# Parameters

area = 20000\*umetre\*\*2

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

El = -60\*mV

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

tau\_ampa=5\*ms

g\_synpk=1.5

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

print("g\_synmax = {}".format(g\_synmaxval))

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),

}

a, b, c, d = lib\_izhik['LTS']

E\_rest = c \* mV

eqs = """

dv\_unitless/dt = ((0.04 \* (v\_unitless)\*\*2) + 5 \* (v\_unitless) + 140 - u\_unitless + I / nA + g\_ampa\*msiemens/cm\*\*2\*(0\*mV - v) / Cm / mV \* ms) / ms : 1

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

v = v\_unitless \* mV : volt

u = u\_unitless \* mV : volt

g\_ampa\_syn : 1

I: amp

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

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

"""

reset\_eq = """

v\_unitless = c

u\_unitless = u\_unitless + d

"""

V\_th = -45 \* mV

# Threshold and refractoriness are only used for spike counting

group = NeuronGroup(num\_neurons,

eqs,

clock=Clock(defaultclock.dt),

# threshold='v > -45\*mV',

threshold='v > V\_th',

reset=reset\_eq,

method='euler')

group.v\_unitless = E\_rest / mV

group.u\_unitless = E\_rest \* b / mV

Sr = Synapses(group,

group,

clock=group.clock,

model='''

g\_synmax : 1

''',

on\_pre='''

z += g\_synmax / ms

''')

Sr.connect(i=[0, 1],j=[1, 2])

g\_synpk=0.08

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

Sr.g\_synmax=g\_synmaxval

Sr.delay=5\*ms #introduces a fixed delay between the firing of the pre cell and the postsynaptic response

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

group.I[0] = 0\*nA

group.I[1] = 0\*nA

run(20.0\*ms)

group.I[0] = 8\*nA

group.I[1] = 0\*nA

run(20.0\*ms)

group.I[0] = 0\*nA

group.I[1] = 0\*nA

run(100.0\*ms)

figure(figsize=(7,9))

subplot(6,1,1)

title("Response with g peak at {}".format(g\_synpk))

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

legend(["g\_ampa of neuron0"], loc="upper left")

subplot(6,1,2)

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

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

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

legend(["potential of neuron0"])

subplot(6,1,3)

plot(monitor2.t/ms, monitor2.g\_ampa[1])

legend(["g\_ampa of neuron1"])

subplot(6,1,4)

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

legend(["potential of neuron1"])

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

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

subplot(6,1,5)

plot(monitor2.t/ms, monitor2.g\_ampa[2])

legend(["g\_ampa of neuron2"])

subplot(6,1,6)

plot(monitor2.t/ms, monitor2.v[2]/mV)

legend(["potential of neuron2"])

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

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

savefig("img/P1Q2\_delay.png")

### Appendix 1.3

from brian2 import \*

defaultclock.dt=.01\*ms

num\_neurons = 3

duration = 2\*second

# Parameters

area = 20000\*umetre\*\*2

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

El = -60\*mV

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

tau\_ampa=5\*ms

g\_synpk=0.08

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

print("g\_synmax = {}".format(g\_synmaxval))

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),

}

a, b, c, d = lib\_izhik['LTS']

E\_rest = c \* mV

eqs = """

dv\_unitless/dt = ((0.04 \* (v\_unitless)\*\*2) + 5 \* (v\_unitless) + 140 - u\_unitless + I / nA + g\_ampa\*msiemens/cm\*\*2\*(-80\*mV - v) / Cm / mV \* ms) / ms : 1

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

v = v\_unitless \* mV : volt

u = u\_unitless \* mV : volt

g\_ampa\_syn : 1

I: amp

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

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

"""

reset\_eq = """

v\_unitless = c

u\_unitless = u\_unitless + d

"""

V\_th = -45 \* mV

# Threshold and refractoriness are only used for spike counting

group = NeuronGroup(num\_neurons,

eqs,

clock=Clock(defaultclock.dt),

# threshold='v > -45\*mV',

threshold='v > V\_th',

reset=reset\_eq,

method='euler')

group.v\_unitless = E\_rest / mV

group.u\_unitless = E\_rest \* b / mV

Sr = Synapses(group,

group,

clock=group.clock,

model='''

g\_synmax : 1

''',

on\_pre='''

z += g\_synmax / ms

''')

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', 'z'),record=True)

group.I[0] = 0\*nA

group.I[1] = 0\*nA

group.I[2] = 0\*nA

run(2.0\*ms,report='text')

group.I[0] = 15\*nA

group.I[1] = 15\*nA

group.I[2] = 15\*nA

run(20\*ms, report='text')

group.I[0] = 0\*nA

group.I[1] = 0\*nA

group.I[2] = 0\*nA

run(100.0\*ms)

figure(3)

subplot(6,1,1)

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

subplot(6,1,2)

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

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

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

subplot(6,1,3)

plot(monitor2.t/ms, monitor2.g\_ampa[1])

subplot(6,1,4)

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

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

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

subplot(6,1,5)

plot(monitor2.t/ms, monitor2.g\_ampa[2])

subplot(6,1,6)

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

# axhline(V\_th / mV, ls='--', c='g')

plot(monitor2.t/ms, monitor2.v[2]/mV)

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

# legend(['potential 0', 'potential 2'], loc='upper right')

from brian2 import \*

defaultclock.dt=.01\*ms

num\_neurons = 3

duration = 2\*second

# Parameters

area = 20000\*umetre\*\*2

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

El = -60\*mV

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

tau\_ampa=5\*ms

g\_synpk=1.5

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

print("g\_synmax = {}".format(g\_synmaxval))

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),

}

a, b, c, d = lib\_izhik['LTS']

E\_rest = c \* mV

eqs = """

dv\_unitless/dt = ((0.04 \* (v\_unitless)\*\*2) + 5 \* (v\_unitless) + 140 - u\_unitless + I / nA + g\_ampa\*msiemens/cm\*\*2\*(-80\*mV - v) / Cm / mV \* ms) / ms : 1

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

v = v\_unitless \* mV : volt

u = u\_unitless \* mV : volt

g\_ampa\_syn : 1

I: amp

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

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

"""

reset\_eq = """

v\_unitless = c

u\_unitless = u\_unitless + d

"""

V\_th = -45 \* mV

# Threshold and refractoriness are only used for spike counting

for i in range(6):

group = NeuronGroup(num\_neurons,

eqs,

clock=Clock(defaultclock.dt),

# threshold='v > -45\*mV',

threshold='v > V\_th',

reset=reset\_eq,

method='euler')

group.v\_unitless = E\_rest / mV

group.u\_unitless = E\_rest \* b / mV

Sr = Synapses(group,

group,

clock=group.clock,

model='''

g\_synmax : 1

''',

on\_pre='''

z += g\_synmax / ms

''')

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

g\_synpk=0.02 \* (i + 1)

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

Sr.g\_synmax=g\_synmaxval

print(Sr.g\_synmax)

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', 'z'),record=True)

group.I[0] = 0\*nA

group.I[1] = 0\*nA

group.I[2] = 0\*nA

run(200.0\*ms)

group.I[0] = 15\*nA

group.I[1] = 15\*nA

group.I[2] = 15\*nA

run(20\*ms)

group.I[0] = 0\*nA

group.I[1] = 0\*nA

group.I[2] = 0\*nA

run(100.0\*ms)

figure(figsize=(7,9))

subplot(6,1,1)

title("Response with g peak at {}".format(g\_synpk))

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

legend(["g\_ampa of neuron0"], loc="upper left")

subplot(6,1,2)

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

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

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

legend(["potential of neuron0"])

subplot(6,1,3)

plot(monitor2.t/ms, monitor2.g\_ampa[1])

legend(["g\_ampa of neuron1"])

subplot(6,1,4)

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

legend(["potential of neuron1"])

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

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

subplot(6,1,5)

plot(monitor2.t/ms, monitor2.g\_ampa[2])

legend(["g\_ampa of neuron2"])

subplot(6,1,6)

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

# axhline(V\_th / mV, ls='--', c='g')

plot(monitor2.t/ms, monitor2.v[2]/mV)

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

legend(['potential 0', 'potential 2'], loc='upper right')

savefig("img/P1Q3\_g{}.png".format(i))

### Appendix 1.4

from brian2 import \*

defaultclock.dt=.01\*ms

num\_neurons = 3

duration = 2\*second

# Parameters

area = 20000\*umetre\*\*2

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

El = -60\*mV

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

tau\_ampa=5\*ms

g\_synpk=1.5

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

# print("g\_synmax = {}".format(g\_synmaxval))

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),

}

a, b, c, d = lib\_izhik['LTS']

E\_rest = c \* mV

eqs = """

dv\_unitless/dt = ((0.04 \* (v\_unitless)\*\*2) + 5 \* (v\_unitless) + 140 - u\_unitless + I / nA + g\_ampa\*msiemens/cm\*\*2\*(-80\*mV - v) / Cm / mV \* ms) / ms : 1

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

v = v\_unitless \* mV : volt

u = u\_unitless \* mV : volt

g\_ampa\_syn : 1

I: amp

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

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

"""

reset\_eq = """

v\_unitless = c

u\_unitless = u\_unitless + d

"""

V\_th = -45 \* mV

# Threshold and refractoriness are only used for spike counting

group = NeuronGroup(num\_neurons,

eqs,

clock=Clock(defaultclock.dt),

# threshold='v > -45\*mV',

threshold='v > V\_th',

reset=reset\_eq,

method='euler')

group.v\_unitless = E\_rest / mV

group.u\_unitless = E\_rest \* b / mV

Sr = Synapses(group,

group,

clock=group.clock,

model='''

g\_synmax : 1

''',

on\_pre='''

z += g\_synmax / ms

''')

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

for i in range(6):

g\_synpk=0.02 \* (i + 1)

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

Sr.g\_synmax=g\_synmaxval

# print(Sr.g\_synmax)

Sr.delay=5\*ms #introduces a fixed delay between the firing of the pre cell and the postsynaptic response

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

group.I[0] = 0\*nA

group.I[1] = 0\*nA

group.I[2] = 0\*nA

run(200.0\*ms)

group.I[0] = 15\*nA

group.I[1] = 15\*nA

group.I[2] = 15\*nA

run(20\*ms)

group.I[0] = 0\*nA

group.I[1] = 0\*nA

group.I[2] = 0\*nA

run(100.0\*ms)

figure(figsize=(7,9))

subplot(6,1,1)

title("Response with g peak at {}".format(g\_synpk))

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

legend(["g\_ampa of neuron0"], loc="upper left")

subplot(6,1,2)

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

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

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

legend(["potential of neuron0"])

subplot(6,1,3)

plot(monitor2.t/ms, monitor2.g\_ampa[1])

legend(["g\_ampa of neuron1"])

subplot(6,1,4)

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

legend(["potential of neuron1"])

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

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

subplot(6,1,5)

plot(monitor2.t/ms, monitor2.g\_ampa[2])

legend(["g\_ampa of neuron2"])

subplot(6,1,6)

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

# axhline(V\_th / mV, ls='--', c='g')

plot(monitor2.t/ms, monitor2.v[2]/mV)

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

legend(['potential 0', 'potential 2'], loc='upper right')

savefig("img/P1Q3\_delayed\_g{}.png".format(i))

### Appendix 1.5

# Define Neurons.

from brian2 import \*

defaultclock.dt=.01\*ms

num\_neurons = 1

duration = 2\*second

# Parameters

area = 20000\*umetre\*\*2

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

El = -60\*mV

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

tau\_ampa = 5\*ms

tau\_gaba = 7\*ms

g\_ampa\_synmax = 0.08

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

g\_gaba\_synmax = 0.7

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

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),

}

a, b, c, d = lib\_izhik['LTS']

E\_rest = c \* mV

eqs = """

dv\_unitless/dt = ((0.04 \* (v\_unitless)\*\*2) + 5 \* (v\_unitless) + 140 - u\_unitless + I / nA + g\_ampa\*msiemens/cm\*\*2\*(0\*mV - v) / Cm / mV \* ms + g\_gaba\*msiemens/cm\*\*2\*(-80\*mV - v) / Cm / mV \* ms) / ms: 1

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

v = v\_unitless \* mV : volt

u = u\_unitless \* mV : volt

g\_ampa\_syn : 1

I: amp

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

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

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

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

"""

reset\_eq = """

v\_unitless = c

u\_unitless = u\_unitless + d

"""

V\_th = -45 \* mV

# SR sensor neuron

SR = NeuronGroup(1,

eqs,

clock=Clock(defaultclock.dt),

# threshold='v > -45\*mV',

threshold='v > V\_th',

reset=reset\_eq,

method='euler')

SR.v\_unitless = E\_rest / mV

SR.u\_unitless = E\_rest \* b / mV

# Submodule neurons

SM = NeuronGroup(3,

eqs,

clock=Clock(defaultclock.dt),

# threshold='v > -45\*mV',

threshold='v > V\_th',

reset=reset\_eq,

method='euler')

SM.v\_unitless = E\_rest / mV

SM.u\_unitless = E\_rest \* b / mV

ex\_con = Synapses(SR,

SM,

clock=SM.clock,

model='''

g\_ampa\_synpk : 1

''',

on\_pre='''

z\_ampa += g\_ampa\_synpk / ms

''')

ex\_con.connect(i=[0, 0, 0],j=[0, 1, 2])

ex\_con.g\_ampa\_synpk = g\_ampa\_synmaxval

ex\_con.delay = 0\*ms

in\_con = Synapses(SM,

SR,

clock=SM.clock,

model='''

g\_gaba\_synpk : 1

''',

on\_pre='''

z\_gaba += g\_gaba\_synpk / ms

''')

in\_con.connect(i=[0, 1, 2], j=[0, 0, 0])

in\_con.g\_gaba\_synpk = g\_gaba\_synmaxval

in\_con.delay = 0\*ms

monitor1=StateMonitor(SR,('v', 'g\_gaba', 'g\_ampa', 'z\_ampa', 'z\_gaba'),record=True)

monitor2=StateMonitor(SM,('v', 'g\_gaba', 'g\_ampa', 'z\_ampa', 'z\_gaba'),record=True)

SR.I[0] = 0\*nA

SM.I[0] = 0\*nA

SM.I[1] = 0\*nA

SM.I[2] = 0\*nA

run(100.0\*ms)

SR.I[0] = 7\*nA

SM.I[0] = 0\*nA

SM.I[1] = 0\*nA

SM.I[2] = 0\*nA

run(1000.0\*ms)

figure(figsize=(7,21))

xx = 12

subplot(xx,1,1)

plot(monitor1.t/ms, monitor1.v[0] / mV)

subplot(xx,1,2)

plot(monitor1.t/ms, monitor1.g\_ampa[0])

subplot(xx,1,3)

plot(monitor1.t/ms, monitor1.g\_gaba[0])

subplot(xx,1,4)

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

xlim(800, 1000)

subplot(xx,1,5)

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

xlim(800, 1000)

subplot(xx,1,6)

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

xlim(800, 1000)

subplot(xx,1,7)

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

subplot(xx,1,8)

plot(monitor2.t/ms, monitor2.g\_ampa[1])

subplot(xx,1,9)

plot(monitor2.t/ms, monitor2.g\_gaba[1])

subplot(xx,1,10)

plot(monitor2.t/ms, monitor2.v[2] / mV)

subplot(xx,1,11)

plot(monitor2.t/ms, monitor2.g\_ampa[2])

subplot(xx,1,12)

plot(monitor2.t/ms, monitor2.g\_gaba[2])

savefig("img/P1Q5\_model1.png")

# Define Neurons.

from brian2 import \*

defaultclock.dt=.01\*ms

num\_neurons = 1

duration = 2\*second

# Parameters

area = 20000\*umetre\*\*2

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

El = -60\*mV

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

tau\_ampa = 5\*ms

tau\_gaba = 5\*ms

g\_ampa\_synmax = 0.08

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

g\_gaba\_synmax = 0.08

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

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),

}

a, b, c, d = lib\_izhik['LTS']

E\_rest = c \* mV

eqs = """

dv\_unitless/dt = ((0.04 \* (v\_unitless)\*\*2) + 5 \* (v\_unitless) + 140 - u\_unitless + I / nA + g\_ampa\*msiemens/cm\*\*2\*(0\*mV - v) / Cm / mV \* ms + g\_gaba\*msiemens/cm\*\*2\*(-80\*mV - v) / Cm / mV \* ms) / ms: 1

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

v = v\_unitless \* mV : volt

u = u\_unitless \* mV : volt

g\_ampa\_syn : 1

I: amp

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

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

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

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

"""

reset\_eq = """

v\_unitless = c

u\_unitless = u\_unitless + d

"""

V\_th = -45 \* mV

# SR sensor neuron

SR = NeuronGroup(1,

eqs,

clock=Clock(defaultclock.dt),

# threshold='v > -45\*mV',

threshold='v > V\_th',

reset=reset\_eq,

method='euler')

SR.v\_unitless = E\_rest / mV

SR.u\_unitless = E\_rest \* b / mV

# Submodule neurons

SM = NeuronGroup(3,

eqs,

clock=Clock(defaultclock.dt),

# threshold='v > -45\*mV',

threshold='v > V\_th',

reset=reset\_eq,

method='euler')

SM.v\_unitless = E\_rest / mV

SM.u\_unitless = E\_rest \* b / mV

ex\_con = Synapses(SR,

SM,

clock=SM.clock,

model='''

g\_ampa\_synpk : 1

''',

on\_pre='''

z\_ampa += g\_ampa\_synpk / ms

''')

ex\_con.connect(i=[0, 0, 0],j=[0, 1, 2])

ex\_con.g\_ampa\_synpk = g\_ampa\_synmaxval

ex\_con.delay = 0\*ms

in\_con = Synapses(SM,

SM,

clock=SM.clock,

model='''

g\_gaba\_synpk : 1

''',

on\_pre='''

z\_gaba += g\_gaba\_synpk / ms

''')

in\_con.connect(i=[0, 0, 1, 1, 2, 2], j=[1, 2, 0, 2, 0, 1])

in\_con.g\_gaba\_synpk = g\_gaba\_synmaxval

in\_con.delay = 0\*ms

monitor1=StateMonitor(SR,('v', 'g\_gaba', 'g\_ampa', 'z\_ampa', 'z\_gaba'),record=True)

monitor2=StateMonitor(SM,('v', 'g\_gaba', 'g\_ampa', 'z\_ampa', 'z\_gaba'),record=True)

SR.I[0] = 0\*nA

SM.I[0] = 0\*nA

SM.I[1] = 0\*nA

SM.I[2] = 0\*nA

run(100.0\*ms)

SR.I[0] = 10\*nA

SM.I[0] = 0\*nA

SM.I[1] = 0\*nA

SM.I[2] = 0\*nA

run(1000.0\*ms)

# group.I[0] = 15\*nA

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

# run(10\*ms, report='text')

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

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

# run(10.0\*ms)

figure(figsize=(7,21))

xx = 12

subplot(xx,1,1)

plot(monitor1.t/ms, monitor1.v[0] / mV)

subplot(xx,1,2)

plot(monitor1.t/ms, monitor1.g\_ampa[0])

subplot(xx,1,3)

plot(monitor1.t/ms, monitor1.g\_gaba[0])

subplot(xx,1,4)

plot(monitor2.t/ms, monitor2.v[0] / mV, c='r')

xlim(800, 1000)

subplot(xx,1,5)

plot(monitor2.t/ms, monitor2.g\_ampa[0], c='r')

xlim(800, 1000)

subplot(xx,1,6)

plot(monitor2.t/ms, monitor2.g\_gaba[0], c='r')

xlim(800, 1000)

subplot(xx,1,7)

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

subplot(xx,1,8)

plot(monitor2.t/ms, monitor2.g\_ampa[1])

subplot(xx,1,9)

plot(monitor2.t/ms, monitor2.g\_gaba[1])

subplot(xx,1,10)

plot(monitor2.t/ms, monitor2.v[2] / mV)

subplot(xx,1,11)

plot(monitor2.t/ms, monitor2.g\_ampa[2])

subplot(xx,1,12)

plot(monitor2.t/ms, monitor2.g\_gaba[2])

savefig("img/P1Q5\_model2.png")

## Appendix 2

### Appendix 2.1: Create bug and move bug

from brian2 import \*

import matplotlib.pyplot as plt

map\_size = 100

global foodx, foody, food\_count, bug\_plot, food\_plot, sr\_plot, sl\_plot,outbugx,outbugy,outbugang,outfoodx,outfoody,outsrx,outsry,outslx,outsly

food\_count = 0

foodx=50

foody=50

duration=100000

outbugx=np.zeros(int(duration/2))

outbugy=np.zeros(int(duration/2))

outbugang=np.zeros(int(duration/2))

outfoodx=np.zeros(int(duration/2))

outfoody=np.zeros(int(duration/2))

outsrx=np.zeros(int(duration/2))

outsry=np.zeros(int(duration/2))

outslx=np.zeros(int(duration/2))

outsly=np.zeros(int(duration/2))

# Sensor neurons

a = 0.02

b = 0.2

c = -65

d = 0.5

I0 = 1250

tau\_ampa=0.7\*ms

# g\_synpk=0.4

g\_synpk=0.2

g\_synmaxval=(g\_synpk)

sensor\_eqs = '''

# equations for neurons

dv/dt = ((0.04 \* (v)\*\*2) + 5 \* (v) + 140 - u + I + g\_ampa\*(0-v)) / ms : 1

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

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

x : 1

y : 1

x\_disp : 1

y\_disp : 1

foodxx : 1

foodyy : 1

mag :1

I = I0 / sqrt(((x-foodxx)\*\*2+(y-foodyy)\*\*2)): 1

'''

sensor\_reset = '''

v = c

u = u + d

'''

# right sensor

sr = NeuronGroup(1, sensor\_eqs, clock=Clock(0.2\*ms), threshold = "v>=30", reset = sensor\_reset,method='euler')

sr.v = c

sr.u = c\*b

sr.x\_disp = 5

sr.y\_disp = 5

sr.x = sr.x\_disp

sr.y = sr.y\_disp

sr.foodxx = foodx

sr.foodyy = foody

sr.mag=1

# left sensor

sl = NeuronGroup(1, sensor\_eqs, clock=Clock(0.2\*ms), threshold = "v>=30", reset = sensor\_reset,method='euler')

sl.v = c

sl.u = c\*b

sl.x\_disp = -5

sl.y\_disp = 5

sl.x = sl.x\_disp

sl.y = sl.y\_disp

sl.foodxx = foodx

sl.foodyy = foody

sl.mag=1

# right bug motor neuron

sbr = NeuronGroup(1, sensor\_eqs, clock=Clock(0.2\*ms), threshold = "v>=30", reset = sensor\_reset,method='euler')

sbr.v = c

sbr.u = c\*b

sbr.foodxx = foodx

sbr.foodyy = foody

sbr.mag=0

# left bug motor neuron

sbl = NeuronGroup(1, sensor\_eqs, clock=Clock(0.2\*ms), threshold = "v>=30", reset = sensor\_reset,method='euler')

sbl.v = c

sbl.u = c\*b

sbl.foodxx = foodx

sbl.foodyy = foody

sbl.mag=0

# The virtual bug

# taum = 4\*ms

base\_speed = 0#1.5

turn\_rate = 5\*Hz

bug\_eqs = '''

#equations for movement here

dx/dt = v \* cos(angle) / ms: 1

dy/dt = v \* sin(angle) / ms: 1

v = 0.5 \* (motorr + motorl) + base\_speed: 1

dangle/dt = (motorr - motorl) / L / ms: 1

dmotorr/dt = -motorr / taum: 1

dmotorl/dt = -motorl / taum: 1

L=4 : 1

z : 1

taum=1\*ms : second

'''

#These are the equation for the motor and speed

bug = NeuronGroup(1, bug\_eqs, clock=Clock(0.2\*ms),method='euler')

bug.motorl = 0

bug.motorr = 0

bug.angle = pi/2

bug.x = 0

bug.y = 0

# Synapses (sensors communicate with bug motor)

w = 5

syn\_rr=Synapses(sr, sbr, clock=Clock(0.2\*ms), model='''

g\_synmax:1

''',

on\_pre='''

g\_ampa+= g\_synmax

''')

syn\_rr.connect(i=[0],j=[0])

syn\_rr.g\_synmax=g\_synmaxval

syn\_ll=Synapses(sl, sbl, clock=Clock(0.2\*ms), model='''

g\_synmax:1

''',

on\_pre='''

g\_ampa+= g\_synmax

''')

syn\_ll.connect(i=[0],j=[0])

syn\_ll.g\_synmax=g\_synmaxval

syn\_r = Synapses(sbr, bug, clock=Clock(0.2\*ms), on\_pre='motorr += w')

syn\_r.connect(i=[0],j=[0])

syn\_l = Synapses(sbl, bug, clock=Clock(0.2\*ms), on\_pre='motorl += w')

syn\_l.connect(i=[0],j=[0])

f = figure(1)

bug\_plot = plot(bug.x, bug.y, 'ko')

food\_plot = plot(foodx, foody, 'b\*')

sr\_plot = plot([0], [0], 'w') # Just leaving it blank for now

sl\_plot = plot([0], [0], 'w')

# Additional update rules (not covered/possible in above eqns)

@network\_operation()

def update\_positions():

global foodx, foody, food\_count

sr.x = bug.x + sr.x\_disp\*sin(bug.angle)+ sr.y\_disp\*cos(bug.angle)

sr.y = bug.y + - sr.x\_disp\*cos(bug.angle) + sr.y\_disp\*sin(bug.angle)

sl.x = bug.x + sl.x\_disp\*sin(bug.angle)+sl.y\_disp\*cos(bug.angle)

sl.y = bug.y - sl.x\_disp\*cos(bug.angle)+sl.y\_disp\*sin(bug.angle)

if ((bug.x-foodx)\*\*2+(bug.y-foody)\*\*2) < 16:

food\_count += 1

foodx = randint(-map\_size+10, map\_size-10)

foody = randint(-map\_size+10, map\_size-10)

if (bug.x < -map\_size):

bug.x = -map\_size

bug.angle = pi - bug.angle

if (bug.x > map\_size):

bug.x = map\_size

bug.angle = pi - bug.angle

if (bug.y < -map\_size):

bug.y = -map\_size

bug.angle = -bug.angle

if (bug.y > map\_size):

bug.y = map\_size

bug.angle = -bug.angle

sr.foodxx = foodx

sr.foodyy = foody

sl.foodxx = foodx

sl.foodyy = foody

@network\_operation(dt=2\*ms)

def update\_plot(t):

global foodx, foody, bug\_plot, food\_plot, sr\_plot, sl\_plot,outbugx,outbugy,outbugang,outfoodx,outfoody,outsrx,outsry,outslx,outsly

indx=int(.5\*t/ms+1)

# bug\_plot[0].remove()

# food\_plot[0].remove()

# sr\_plot[0].remove()

# sl\_plot[0].remove()

bug\_x\_coords = [bug.x, bug.x-4\*cos(bug.angle), bug.x-8\*cos(bug.angle)] # ant-like body

bug\_y\_coords = [bug.y, bug.y-4\*sin(bug.angle), bug.y-8\*sin(bug.angle)]

outbugx[indx-1]=bug.x[0]

outbugy[indx-1]=bug.y[0]

outbugang[indx-1]=bug.angle[0]

outfoodx[indx-1]=foodx

outfoody[indx-1]=foody

outsrx[indx-1]=sr.x[0]

outsry[indx-1]=sr.y[0]

outslx[indx-1]=sl.x[0]

outsly[indx-1]=sl.y[0]

# bug\_plot = plot(bug\_x\_coords, bug\_y\_coords, 'ko') # Plot the bug's current position

# sr\_plot = plot([bug.x, sr.x], [bug.y, sr.y], 'b')

# sl\_plot = plot([bug.x, sl.x], [bug.y, sl.y], 'r')

# food\_plot = plot(foodx, foody, 'b\*')

# axis([-100,100,-100,100])

# draw()

#print "."

# pause(0.01)

ML = StateMonitor(sl, ('v', 'I'), record=True)

MR = StateMonitor(sr, ('v', 'I'), record=True)

#MB = StateMonitor(bug, ('motorl', 'motorr', 'speed', 'angle', 'x', 'y'), record = True)

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

np.save('outbugx',outbugx)

np.save('outbugy',outbugy)

np.save('outbugang',outbugang)

np.save('outfoodx',outfoodx)

np.save('outfoody',outfoody)

np.save('outsrx',outsrx)

np.save('outsry',outsry)

np.save('outslx',outslx)

np.save('outsly',outsly)

print("{} food is eaten. ".format(food\_count))

### Appendix 2.2: Show BUG

from brian2 import \*

import numpy as np

import plotly.graph\_objects as go

import numpy as np

import plotly.io as pio

show\_duration = 5000

pio.renderers.default='browser'

Ox=np.load('outbugx.npy')

Oy=np.load('outbugy.npy')

srx=np.load('outsrx.npy')

sry=np.load('outsry.npy')

slx=np.load('outslx.npy')

sly=np.load('outsly.npy')

Ba=np.load('outbugang.npy')

Fx=np.load('outfoodx.npy')

Fy=np.load('outfoody.npy')

bug\_x\_coords=np.zeros((show\_duration,3))

bug\_y\_coords=np.zeros((show\_duration,3))

for i in range(0,show\_duration):

# Remove the last bug's position from the figure window

bug\_x\_coords[i] = [Ox[i], Ox[i]-4\*cos(Ba[i]), Ox[i]-8\*cos(Ba[i])]

bug\_y\_coords[i] = [Oy[i], Oy[i]-4\*sin(Ba[i]), Oy[i]-8\*sin(Ba[i])]

# Create figure

fig = go.Figure(

data=[go.Scatter(x=Fx, y=Fy,

mode="lines",

line=dict(width=2, color="blue"))],

layout=go.Layout(

xaxis=dict(range=[-100, 100], autorange=False, zeroline=False),

yaxis=dict(range=[-100, 100], autorange=False, zeroline=False),

height=600,

width=600,

title\_text="Kinematic Generation of a Planar Curve", hovermode="closest",

updatemenus=[dict(type="buttons",

buttons=[dict(

label="Play",

method="animate",

args = [None, {"frame": {"duration": 10,

"redraw": False},

"fromcurrent": True,

"transition": {"duration": 0}}])])]),

frames=[

# go.Frame(

# data=[go.Scatter(

# x=[bug\_x\_coords[k][0],bug\_x\_coords[k][1]],

# y=[bug\_y\_coords[k][0],bug\_y\_coords[k][1]],

# mode="markers",

# marker=dict(color=["red","blue"], size=[20,20]))])

go.Frame(

data=[go.Scatter(

x=np.concatenate((bug\_x\_coords[k],[srx[k], slx[k]],[Fx[k]] )),

y=np.concatenate((bug\_y\_coords[k],[sry[k], sly[k]], [Fy[k]])),

mode="markers+markers+markers+markers+markers+markers",

marker=dict(color=["black","black","black","blue","red","green"], size=[10,10,10,8,8,20]))])

for k in range(1,show\_duration)]

)

fig.show()