# Xuenan Wang

# NetID: xw336

# Email: roderick\_wang@outlook.com

CS 525 Homework 1

## Questions:

1. IF model will continually integrate input voltage no matter how small the input is and as long as input is there, the total voltage will approach infinity. LIF model will probably ignore a very small input because as the input integrate, the leaking part will set the voltage back to rest.

2. For a very large input, IF model will integrate voltage incredibly fast and approach infinity. And LIF model will integrate voltage as well but at a relative slow rate because of the leaking mechanism.

3. The most important limitation of LIF model is that this model is isolated and can not memorize any situation or history from other neuron, or even previous status of itself. And this will result in problems like some neurons can not be precisely modeled. For instance, there is a phenomenon call adaptation in regular-spiking neuron and it's a slow process that builds up over several spikes, since LIF model can not memorize previous spike, this adaptation can not be captured. Also, LIF model can not show the depolarization process after one spike.

## Programming:

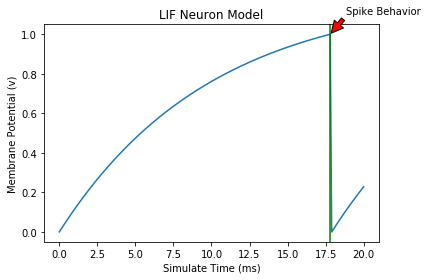
1.

Figure 1. Spike a LIF neuron with input of 1.2 and simulate time of 20

Figure 1 shows a clear spike behavior and a potential decay over time. The green line in figure 1 indicate the time where the neuron spike.

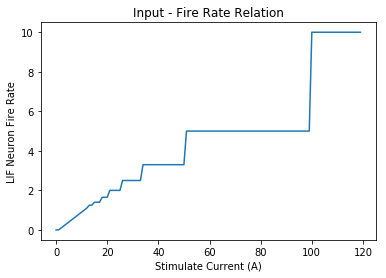
2.

Figure 2. LIF neuron fire rate vs stimulate current

Figure 2 shows that as the stimulate current increase, the LIF neuron fire rate increase as well. However, when the stimulate current goes up to a certain value, the fire rate does not continue go up, as Figure 3 shows.

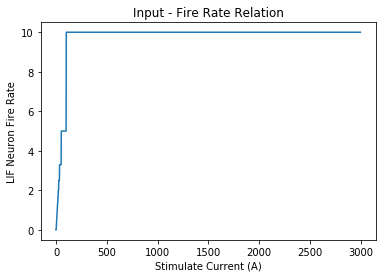
3.

Figure 3. As stimulate current increase to a certain value, the fire rate does not continue increase

If continually increase the stimulate current, the fire rate does not go up. Because every neuron has a maximum fire speed and no matter how strong the stimulation is, neuron can not break this limit.

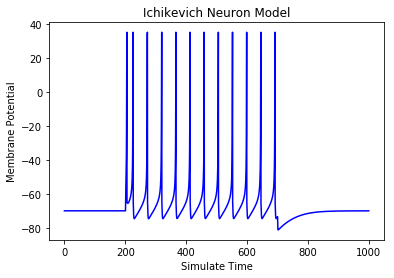
4.

Figure 4. Simulation of Ichikevich Neuron Model

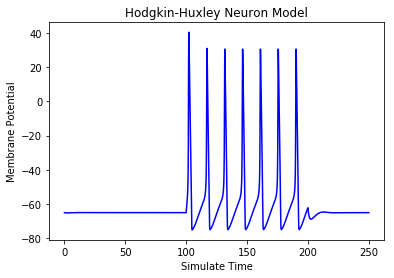
5.

Figure 5. Simulation of Hodgkin-Huxley Neuron Model

## Source Code:

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plt

**from** scipy **import** integrate

**from** random **import** **\***

**import** math

**class** **LIF\_Neuron:**

**def** \_\_init\_\_**(**self**,** simulate\_time**,** input\_number**):**

self**.**simulate\_time **=** simulate\_time

self**.**d\_time **=** 0.1

self**.**membrane\_potential **=** zeros**(**int**((**self**.**simulate\_time **/** self**.**d\_time**)** **+** 1**))**

self**.**threshold **=** 1

self**.**rest\_potential **=** 0

self**.**initial\_potential **=** 0

self**.**membrane\_resistance **=** 1

self**.**membrane\_capacitance **=** 10

self**.**time\_constant **=** self**.**membrane\_capacitance **\*** self**.**membrane\_resistance

self**.**firing\_rate **=** 10

self**.**potential\_difference **=** 0

self**.**spike\_number **=** 0

self**.**not\_spike\_amount **=** 0

self**.**spike\_amount **=** 0

self**.**fire\_rate\_array **=** **[]**

self**.**total\_fire\_rate **=** 0

self**.**weight **=** **[]**

self**.**spike\_time **=** **[]**

**for** i **in** range**(**input\_number**):**

random\_weight **=** math**.**ceil**(**uniform**(**0**,** 2000**)** **-** 1000**)** **/** 1000

self**.**weight**.**append**(**random\_weight**)**

**def** stimulate\_neuron**(**self**,** stimulate\_current**):**

self**.**ctr **=** 0.0

self**.**spike\_amount **=** 0

self**.**membrane\_potential **=** zeros**(**int**((**self**.**simulate\_time **/** self**.**d\_time**)** **+** 1**))**

self**.**potential\_difference **=** 0

**for** i **in** range**(**1**,** len**(**self**.**membrane\_potential**)):**

self**.**potential\_difference **=** **(-**1**\***self**.**membrane\_potential**[**i**-**1**]+**self**.**membrane\_resistance**\***stimulate\_current**)**

#print("membrane\_potential:",self.membrane\_potential)

self**.**membrane\_potential**[**i**]** **=** self**.**membrane\_potential**[**i**-**1**]+**self**.**potential\_difference**/**self**.**time\_constant**\***self**.**d\_time

**if(**self**.**membrane\_potential**[**i**]** **>=** self**.**threshold**):**

self**.**spike\_time**.**append**(**i **/** len**(**self**.**membrane\_potential**)** **\*** simulate\_time**)**

self**.**membrane\_potential**[**i**]** **=** self**.**rest\_potential

self**.**spike\_amount **+=** 1

**return** **(**math**.**ceil**((**self**.**spike\_amount **/** self**.**simulate\_time**)** **\*** 1000**))** **/** 1000

**def** plot\_neuron**(**self**):**

simulate\_time **=** arange**(**0**,** self**.**simulate\_time **+** self**.**d\_time**,** self**.**d\_time**)**

plt**.**plot**(**simulate\_time**,** self**.**membrane\_potential**)**

plt**.**xlabel**(**'Simulate Time (ms)'**)**

plt**.**ylabel**(**'Membrane Potential (v)'**)**

plt**.**title**(**'LIF Neuron Model'**)**

plt**.**annotate**(**'Spike Behavior'**,** xy **=** **(**self**.**spike\_time**[**1**],** self**.**threshold**),** xytext **=** **(**self**.**spike\_time**[**1**]** **+** 1**,** self**.**threshold **+** 1**),** arrowprops **=** dict**(**facecolor **=** 'red'**,** shrink **=** 0.05**))**

plt**.**axvline**(**x **=** self**.**spike\_time**[**1**],** color **=** 'g'**)**

plt**.**show**()**

**print(**'Spike rate for this input is '**,** self**.**spike\_amount **/** self**.**simulate\_time**)**

**def** calculate\_fire\_rate**(**self**):**

simulate\_time **=** arange**(**0**,** self**.**simulate\_time **+** self**.**d\_time**,** self**.**d\_time**)**

**return** self**.**spike\_amount **/** self**.**simulate\_time

**class** **Ichikevich\_Neuron:**

**def** \_\_init\_\_**(**self**):**

self**.**simulate\_time **=** 1000

self**.**d\_time **=** 0.5

self**.**param\_a **=** 0.02

self**.**param\_b **=** 0.2

self**.**param\_c **=** **-**65

self**.**param\_d **=** 8

self**.**lapp **=** 10

self**.**tr **=** np**.**array**([**200**,** 700**])//**self**.**d\_time

self**.**T **=** int**(**self**.**simulate\_time**/**self**.**d\_time**)**

self**.**v **=** np**.**zeros**(**self**.**T**)**

self**.**u **=** np**.**zeros**(**self**.**T**)**

self**.**v**[**0**]** **=** **-**70

self**.**u**[**0**]** **=** **-**14

**def** stimulate\_neuron**(**self**):**

**for** i **in** np**.**arange**(**self**.**T**-**1**):**

**if** i**>**self**.**tr**[**0**]** **and** i**<**self**.**tr**[**1**]:**

l **=** self**.**lapp

**else:**

l **=** 0

**if** self**.**v**[**i**]<**35**:**

dv **=** **(**0.04**\***self**.**v**[**i**]+**5**)\***self**.**v**[**i**]+**140**-**self**.**u**[**i**]**

self**.**v**[**i**+**1**]** **=** self**.**v**[**i**]+(**dv**+**l**)\***self**.**d\_time

du **=** self**.**param\_a**\*(**self**.**param\_b**\***self**.**v**[**i**]-**self**.**u**[**i**])**

self**.**u**[**i**+**1**]** **=** self**.**u**[**i**]+**self**.**d\_time**\***du

**else:**

self**.**v**[**i**]** **=** 35

self**.**v**[**i**+**1**]** **=** self**.**param\_c

self**.**u**[**i**+**1**]** **=** self**.**u**[**i**]** **+** self**.**param\_d

**def** plot\_neuron**(**self**):**

tvec **=** np**.**arange**(**0**,** self**.**simulate\_time**,** self**.**d\_time**)**

plt**.**plot**(**tvec**,** self**.**v**,** 'b'**,** label **=** 'Voltage Trace'**)**

plt**.**xlabel**(**'Simulate Time'**)**

plt**.**ylabel**(**'Membrane Potential'**)**

plt**.**title**(**'Ichikevich Neuron Model'**)**

plt**.**show**()**

**class** **HodgkinHuxley\_Neuron():**

**def** \_\_init\_\_**(**self**):**

self**.**C\_m **=** 1.0

self**.**g\_Na **=** 120.0

self**.**g\_K **=** 36.0

self**.**g\_L **=** 0.3

self**.**E\_Na **=** 50.0

self**.**E\_K **=** **-**77.0

self**.**E\_L **=** **-**54.387

self**.**time **=** np**.**arange**(**0.0**,** 450.0**,** 0.01**)**

**def** alpha\_m**(**self**,** V**):**

**return** 0.1**\*(**V**+**40.0**)/(**1.0 **-** math**.**exp**(-(**V**+**40.0**)** **/** 10.0**))**

**def** beta\_m**(**self**,** V**):**

**return** 4.0**\***math**.**exp**(-(**V**+**65.0**)** **/** 18.0**)**

**def** alpha\_h**(**self**,** V**):**

**return** 0.07**\***math**.**exp**(-(**V**+**65.0**)** **/** 20.0**)**

**def** beta\_h**(**self**,** V**):**

**return** 1.0**/(**1.0 **+** math**.**exp**(-(**V**+**35.0**)** **/** 10.0**))**

**def** alpha\_n**(**self**,** V**):**

**return** 0.01**\*(**V**+**55.0**)/(**1.0 **-** math**.**exp**(-(**V**+**55.0**)** **/** 10.0**))**

**def** beta\_n**(**self**,** V**):**

**return** 0.125**\***math**.**exp**(-(**V**+**65**)** **/** 80.0**)**

**def** I\_Na**(**self**,** V**,** m**,** h**):**

**return** self**.**g\_Na **\*** m**\*\***3 **\*** h **\*** **(**V **-** self**.**E\_Na**)**

**def** I\_K**(**self**,** V**,** n**):**

**return** self**.**g\_K **\*** n**\*\***4 **\*** **(**V **-** self**.**E\_K**)**

**def** I\_L**(**self**,** V**):**

**return** self**.**g\_L **\*** **(**V **-** self**.**E\_L**)**

**def** I\_inj**(**self**,** t**):**

**return** 10**\*(**t**>**100**)** **-** 10**\*(**t**>**200**)** **+** 35**\*(**t**>**300**)** **-** 35**\*(**t**>**400**)**

@staticmethod

**def** dALLdt**(**X**,** t**,** self**):**

V**,** m**,** h**,** n **=** X

dVdt **=** **(**self**.**I\_inj**(**t**)** **-** self**.**I\_Na**(**V**,** m**,** h**)** **-** self**.**I\_K**(**V**,** n**)** **-** self**.**I\_L**(**V**))** **/** self**.**C\_m

dmdt **=** self**.**alpha\_m**(**V**)\*(**1.0**-**m**)** **-** self**.**beta\_m**(**V**)\***m

dhdt **=** self**.**alpha\_h**(**V**)\*(**1.0**-**h**)** **-** self**.**beta\_h**(**V**)\***h

dndt **=** self**.**alpha\_n**(**V**)\*(**1.0**-**n**)** **-** self**.**beta\_n**(**V**)\***n

**return** dVdt**,** dmdt**,** dhdt**,** dndt

**def** stimulate\_neuron**(**self**):**

X **=** odeint**(**self**.**dALLdt**,** **[-**65**,** 0.05**,** 0.6**,** 0.32**],** self**.**time**,** args**=(**self**,))**

V **=** X**[:,**0**]**

m **=** X**[:,**1**]**

h **=** X**[:,**2**]**

n **=** X**[:,**3**]**

ina **=** self**.**I\_Na**(**V**,** m**,** h**)**

ik **=** self**.**I\_K**(**V**,** n**)**

il **=** self**.**I\_L**(**V**)**

plt**.**title**(**'Hodgkin-Huxley Neuron Model'**)**

plt**.**plot**(**self**.**time**,** V**,** 'k'**)**

plt**.**ylabel**(**'Membrane Potential'**)**

plt**.**xlabel**(**'Simulate Time'**)**

plt**.**show**()**

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

#LIF\_Neuron Model

simulate\_time **=** 20

stimulation\_number **=** 5

stimulate\_current **=** 1.2

lif\_neuron **=** LIF\_Neuron**(**simulate\_time**,** stimulation\_number**)**

lif\_neuron**.**stimulate\_neuron**(**stimulate\_current**)**

lif\_neuron**.**plot\_neuron**()**

'''

fire\_rate = []

for i in range(0, 1500):

lif\_neuron\_new = LIF\_Neuron(simulate\_time, stimulation\_number)

lif\_neuron\_new.stimulate\_neuron(i)

fire\_rate.append(lif\_neuron\_new.calculate\_fire\_rate())

plt.plot(range(0, 1500), fire\_rate)

plt.xlabel('Stimulate Current (A)')

plt.ylabel('LIF Neuron Fire Rate')

plt.title('Input - Fire Rate Relation')

plt.show()

'''

#Ichikevich Neuron Model

ichikevich\_neuron **=** Ichikevich\_Neuron**()**

ichikevich\_neuron**.**stimulate\_neuron**()**

ichikevich\_neuron**.**plot\_neuron**()**

#Hodgkin-Huxley Neuron Model

hodgkin\_huxley **=** HodgkinHuxley\_Neuron**()**

hodgkin\_huxley**.**stimulate\_neuron**()**