به نام خدا

.

مهرداد برادران

.

گزارش پروژه سری 2 علوم اعصاب

.

جمعیت های نورونی

.

استاد خرد پیشه

.

شماره دانشجویی 99222020

.

درس علوم اعصاب محاسابتی

Computational neuroscience

در تمرین این سری با استفاده از مدل نورونی سری اول LIF و پیوند دادن این مدل های نورونی و ایجاد جمعیت نورونی به صورت یک کلاس NEURON GROUP رفتار این جمعیتای نورونی را شبیه سازی کردیم :

Implementing LIF Model

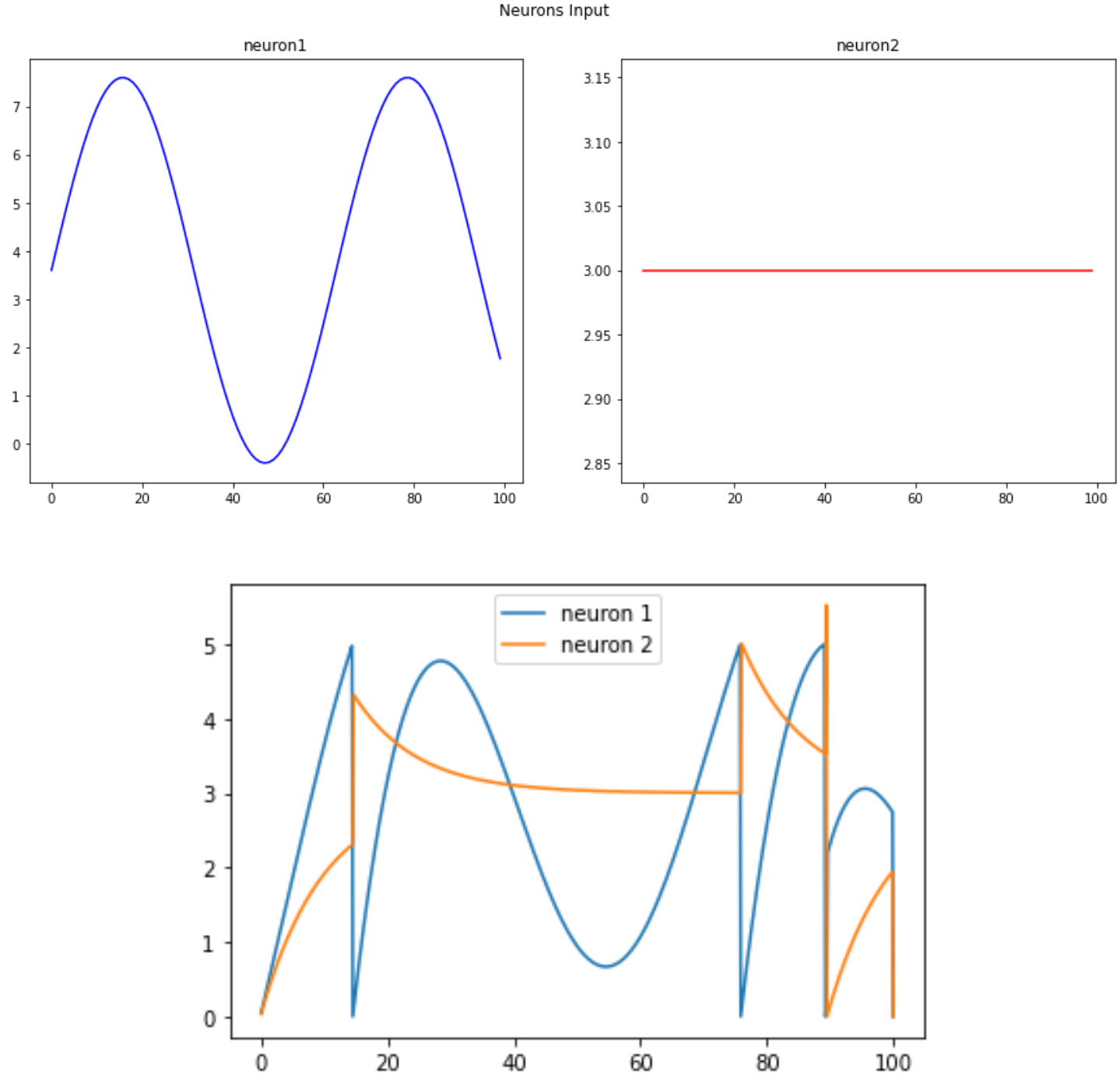
In [133]:

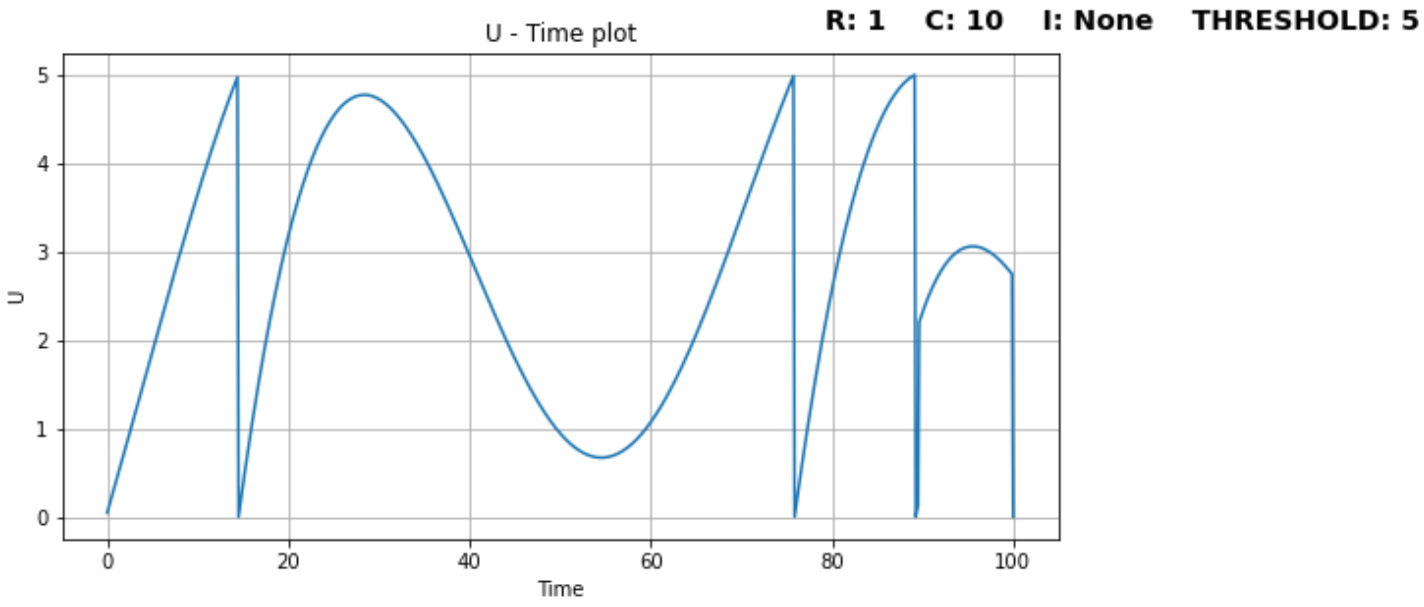
```
class LIF:
    def __init__(self, i_func, time_interval=100, dt=0.125, u_rest=0, R=1,
                 C=10, threshold=5, neuron_type='excitatory'):
        self.time_interval = time_interval
        self.dt = dt
        self.i_func = i_func
        self.u_rest = u_rest
        self.R = R
        self.C = C
        self.threshold = threshold
        self.u = []
        self.i_init = []
        self.timer = []
        self.cur_time = 0
        self.is_spiked = False
        self.type = neuron_type

     #init u values
    def start(self):
        self.timer = np.arange(0, self.time_interval + self.dt, self.dt)
        u = [self.u_rest for i in range(len(self.timer))]
        self.i_init = [self.i_func(j) for j in self.timer]

        const = self.R * self.C
        for t in range(len(self.timer)):
            self.is_spiked = False
            self.cur_time = t
            u[t] = u[t-1] + (((-u[t-1] + self.u_rest) + self.R * self.i_init[t]) * self.dt)/const
            self.u = u
            #checking for action potential
            if u[t] >= self.threshold or u[t] < self.u_rest:
                u[t] = self.u_rest
                self.is_spiked = True
                self.u = u
            yield self.is_spiked
```

In [148]:

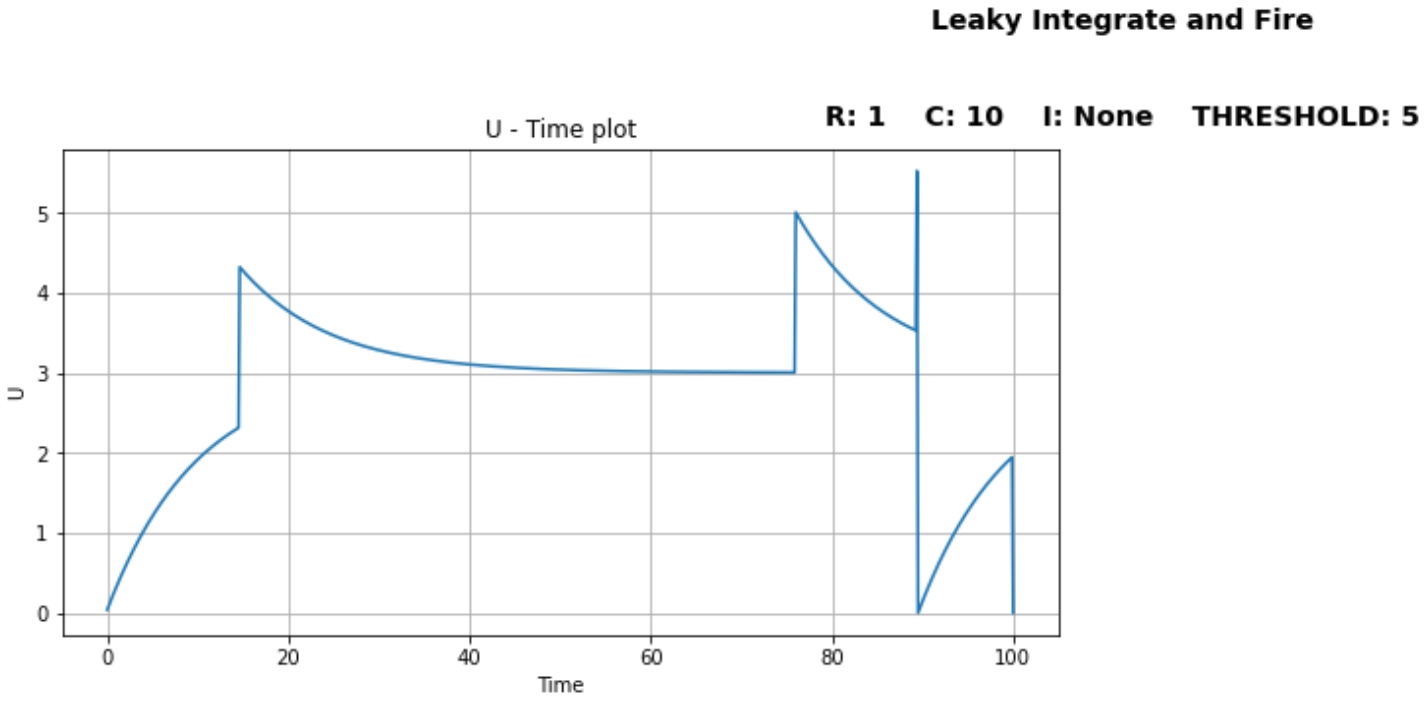
```
class NeuronsGroup:

  def __init__(self, neurons, connections, excitatory_w=2, inhibitory_w=-2, excitatory_delay=1,
               inhibitory_delay=1, iteration_count=800):
    self.neurons = neurons
    self.neroun_action = []
    for i in neurons:
      self.neroun_action.append(i.start())

    self.connections = connections
    self.excitatory_w = excitatory_w
    self.inhibitory_w = inhibitory_w
    self.iteration_count = iteration_count
    self.spikes = []
    self.excitatory_spikes_time = []
    self.excitatory_spikes = []
    self.inhibitory_spikes_time = []
    self.inhibitory_spikes = []
    self.excitatory_delay = excitatory_delay
    self.inhibitory_delay = inhibitory_delay
    self.spikes_effect = []

  def start(self):
    self.spikes_effect = [[0] * len(self.neurons) for _ in range(self.iteration_count)]
    for t in range(self.iteration_count):

      for i in range(len(self.neroun_action)):

        action_info = next(self.neroun_action[i])
        if action_info == True :

          for j in self.connections[i]:
            if self.neurons[i].type == 'excitatory':
              self.excitatory_spikes.append(i + 1)
              self.excitatory_spikes_time.append(t)
              if t+self.excitatory_delay < self.iteration_count:
                self.spikes_effect[t + self.excitatory_delay][j] += self.excitatory_w

            if self.neurons[i].type == 'inhibitory':
              self.inhibitory_spikes.append(i + 1)
              self.inhibitory_spikes_time.append(t)
              if t+self.inhibitory_delay < self.iteration_count:
                self.spikes_effect[t+self.inhibitory_delay][j] += self.inhibitory_w
      for i in range(len(self.neurons)):
        self.neurons[i].u[self.neurons[i].cur_time] += self.spikes_effect[t][i]
```

در ابتدا دو نورون excitatory را با ارگومان های مشخص به شکل یک جمعیت نورونی در اورده و نوع کانکشن این نورون هارا به همراه یک جریان ورودی مشخص میکنیم و نمودار خروجی این جمعیت نورونی به شکل زیر است :

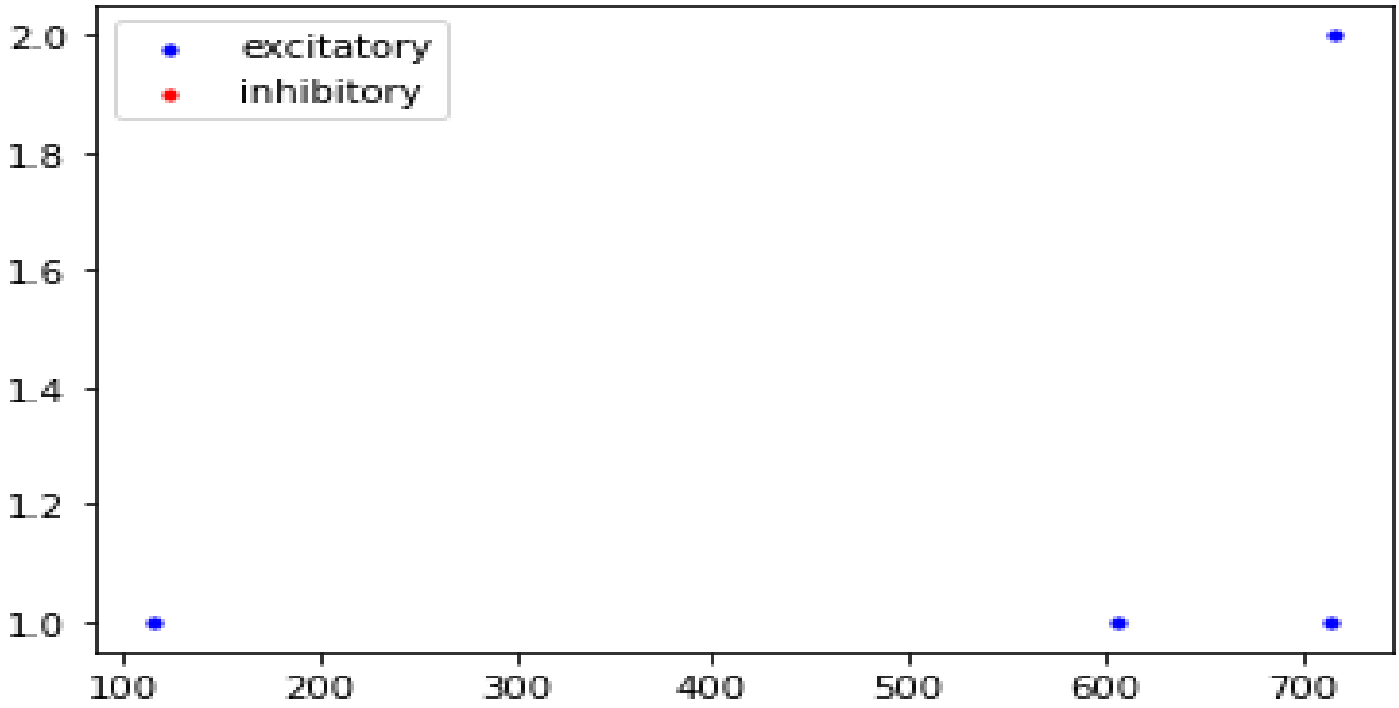

Neurons Input
neuron1
neuron2
neuron 1
neuron 2



نمودار پتانسیل نورون اول



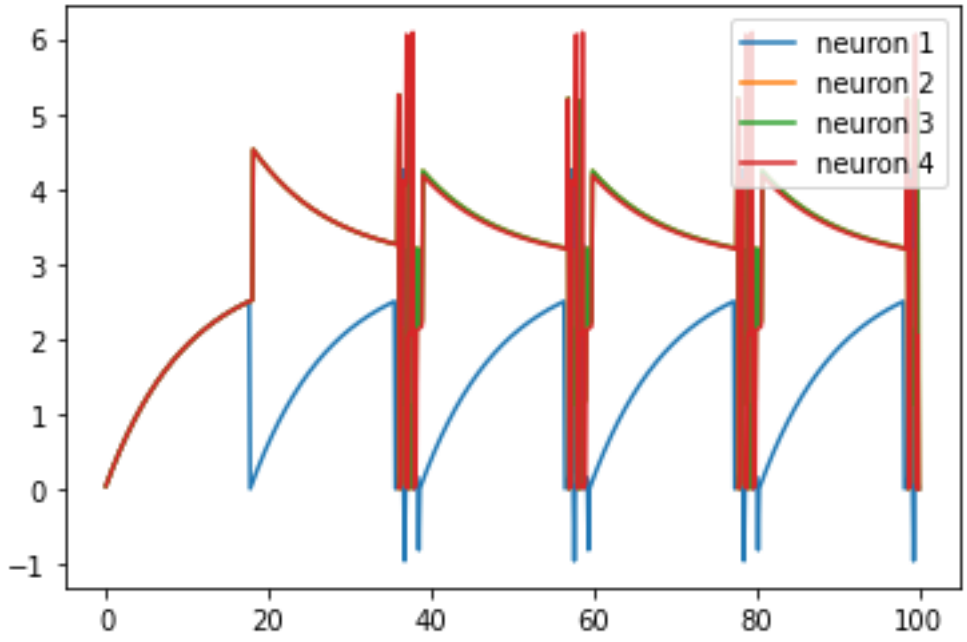
نمودار پتانسیل نورون دوم

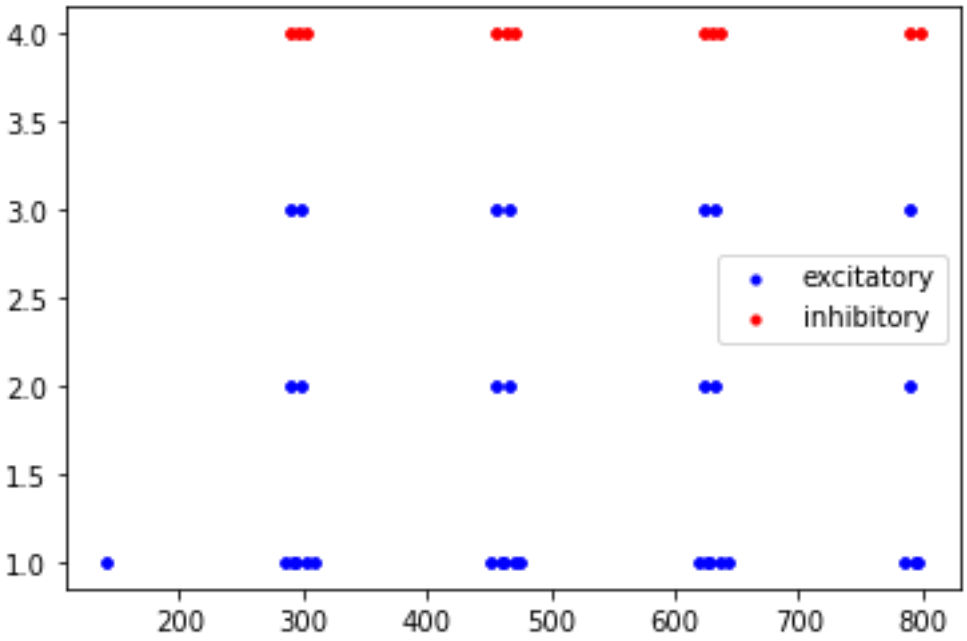


نمودار زمانی اسپایک دو نورون

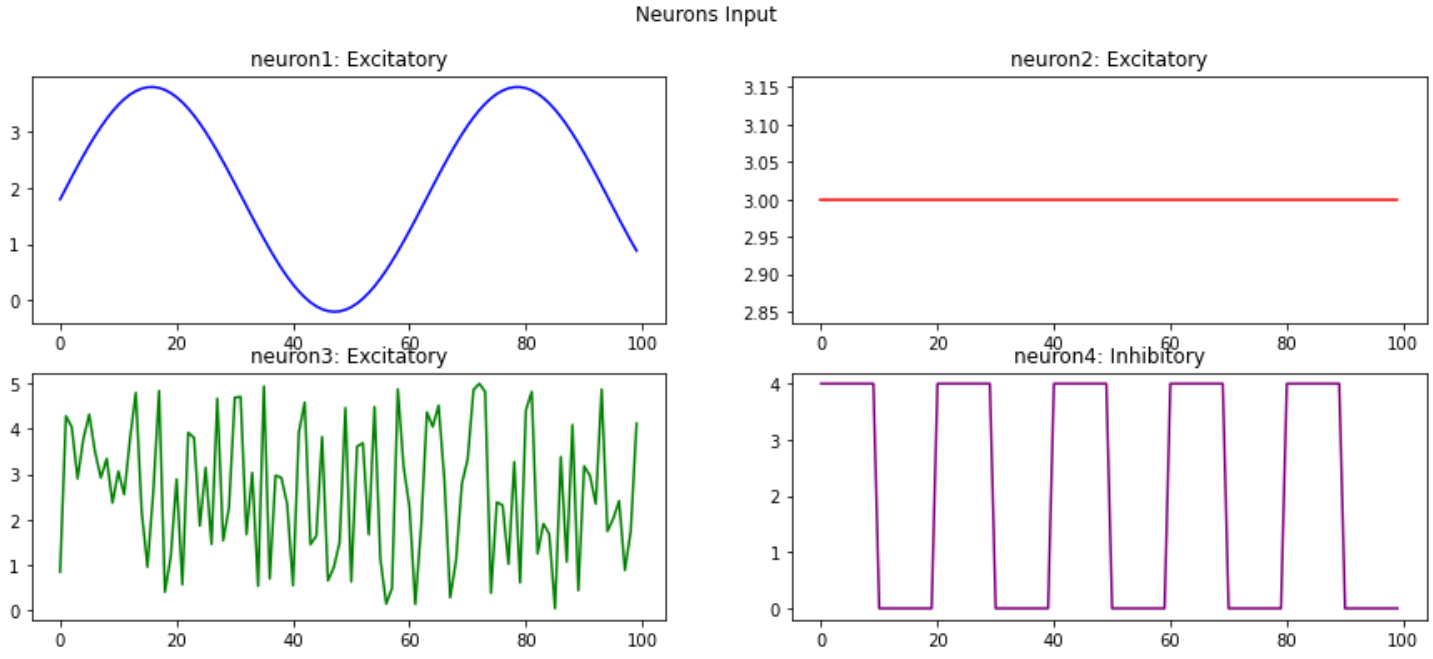
تاثیر پذیری و گزاری این دو نورون روی یکدیگر به وضوح قابل مشاهده است.

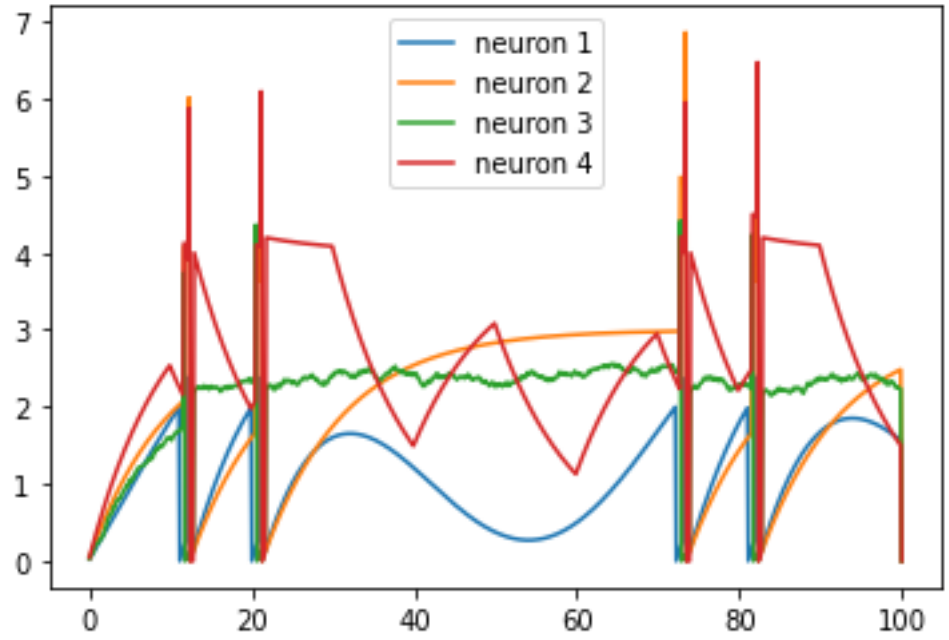
در مرحله بعد یک جمعیت نورونی که شامل نورون inhibitory هم هست را تشکیل میدیم مانند قبل ورودی های دلخواه را وارد کرده و به عنوان یک جریان ورودی میدهیم :



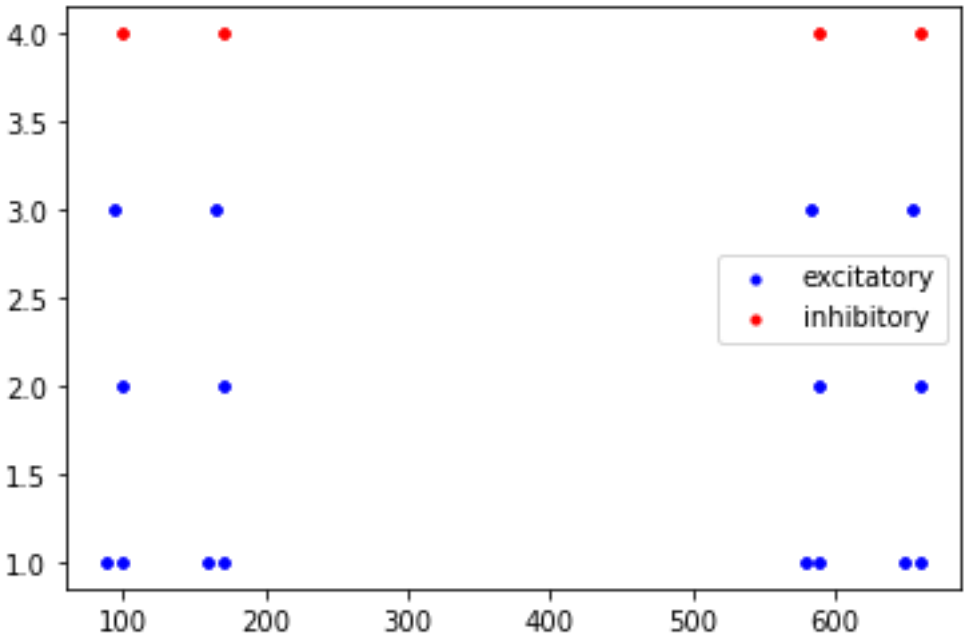


حال به هر یک از 4 نورون قبلی جریان های ورودی متفاوتی را وارد میکنیم :





تاثیر پذیری نورون هارا میتوان مشاهده کرد که در لحظاتی که نورون های تحریکی اسپایک میزنند پتانسیل نورون های متصل به ان در جمعیت نورونی افزایش میابد و عکس این قضیه برای نورون مهاری صادق است.



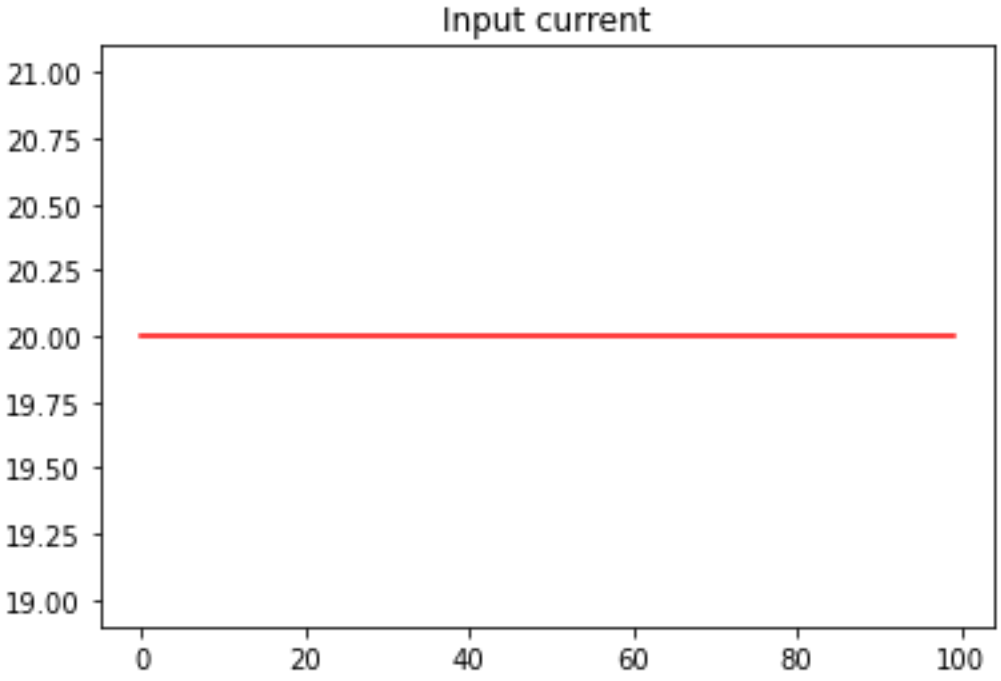
در ساخت جمعیت نورونی با تعداد بالا نیازمند الگوریتمی هستیم تا بتوانیم نورون هارا به یکدیگر متصل کنیم در اینجا به هریک از نوع های نورونی یک درصد احتمالی برای تعداد کانکشن ها میدیم تا بتونیم مشخص کنیم هر یک از نوع نورون ها به چند نورون دیگر متصل هستند . علاوه بر آن پس از مشخص شدن تعداد کانکشن ها لیستی برای مشخص کردن کانکشن هر نورون تعیین میکنیم . میتوان ورودی های اضافه lif و جمعیت نورونی را با یک دیکشنری مانند kwargs** به عنوان ورودی به تابع داد تا در صورت نیاز به تغییر نوع داده ها آن هارا اعمال کرد.

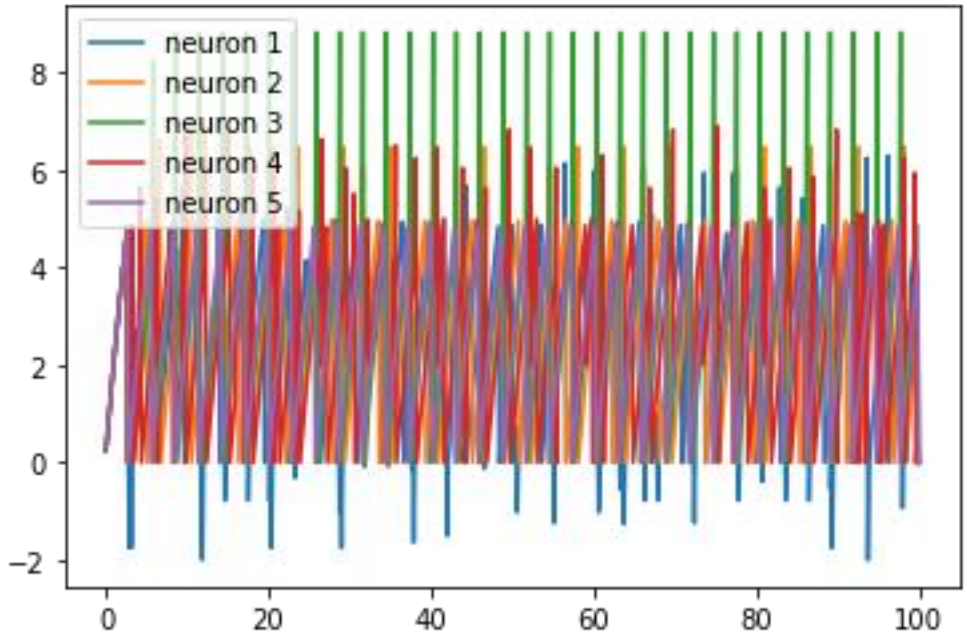
Create Neurons Group

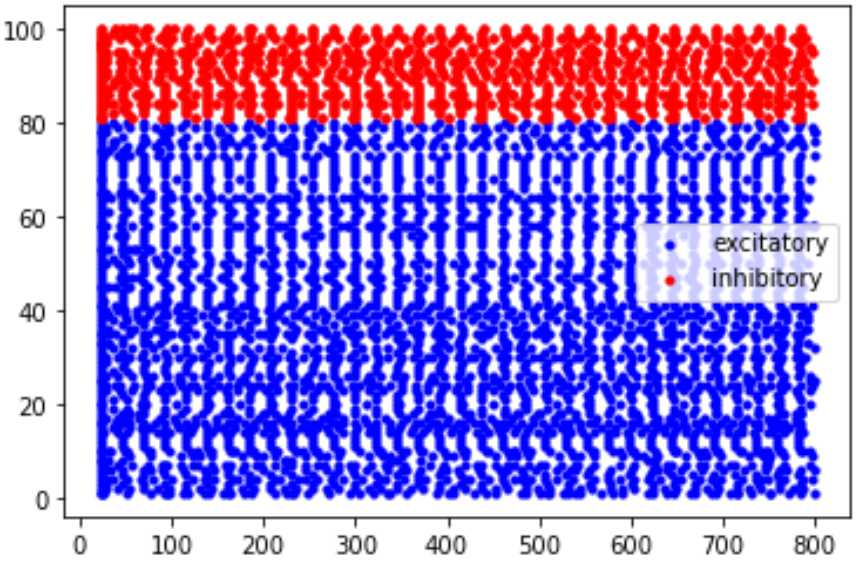
```
def create_neuron_group(neurons_count, excitatory_count, inhibitory_count, excitatory_prob, inhibitory_prob, I, **kwargs):
  neurons = []
  connections = []
  excitatory_neuron_conn_count = math.ceil(neurons_count * excitatory_prob)
  inhibitory_neuron_conn_count = math.ceil(neurons_count * inhibitory_prob)

  for i in range(excitatory_count):
    args = {}
    if 'excitatory' in kwargs.keys():
      for arg in kwargs['excitatory']:
        args[arg] = kwargs[arg][i]
    neuron = LIF(I[i], neuron_type='excitatory', **args)
    neurons.append(neuron)
    connections.append(random.sample(range(neurons_count), excitatory_neuron_conn_count))
  for i in range(inhibitory_count):
    args = {}
    if 'Inhibitory' in kwargs.keys():
      for arg in kwargs['inhibitory']:
        args[arg] = kwargs[arg][i]
    neuron = LIF(I[i], neuron_type='inhibitory', **args)
    neurons.append(neuron)
    connections.append(random.sample(range(neurons_count), inhibitory_neuron_conn_count))

  neurons_group = NeuronsGroup(neurons, connections, **kwargs)
  return neurons_group
```

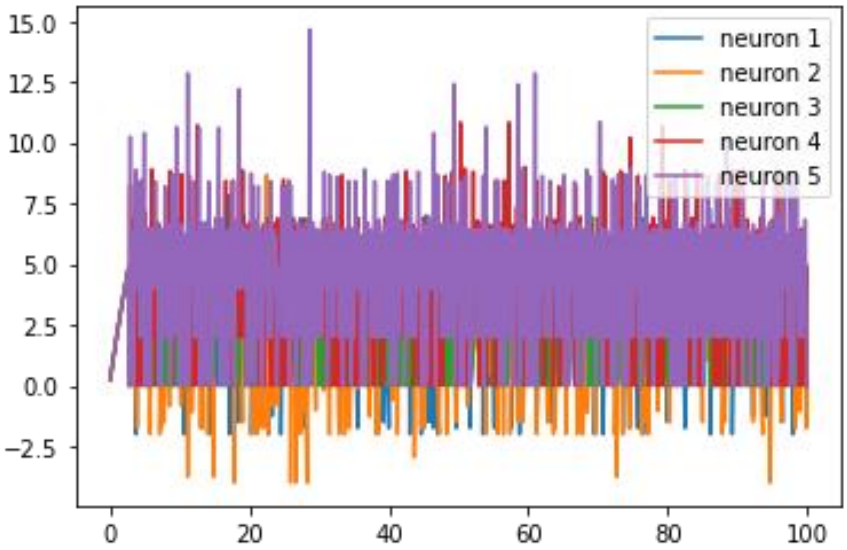
در ادامه تمرین یک جمعیت نورونی 100 تایی با 80 نورون تحریکی و 20 نورون مهاری میسازیم

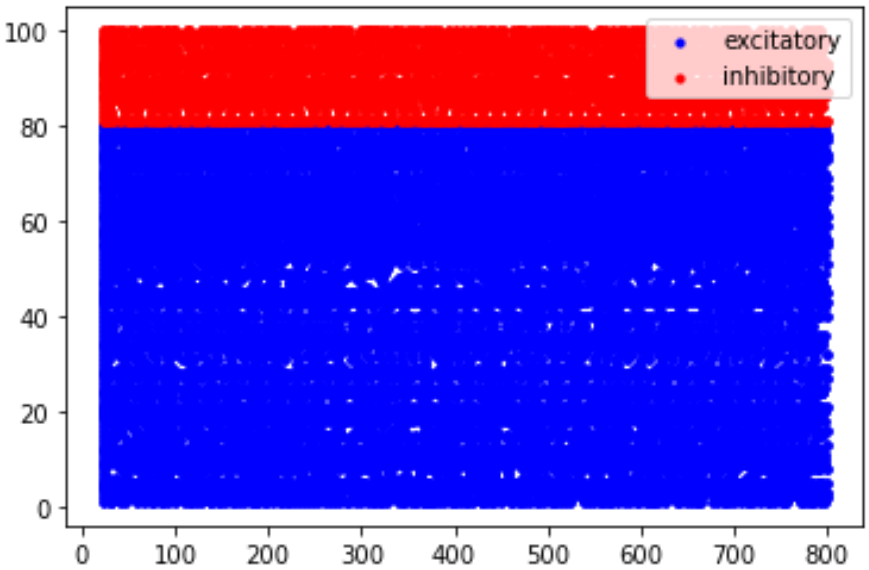
ورودی هارا به دلخواه وارد کرده و با تغییرات جریان ورودی و تغییرات احتمال کانکشن نورون های مهاری و تحریکی تاثیرات نورون هارا در یک جمعیت نورونی بررسی میکنیم به وضوح پیداست که با افزایش تعداد نورون های مهاری تعداد اسپایک نورون های تحریکی کاهش میابد و به عکس :

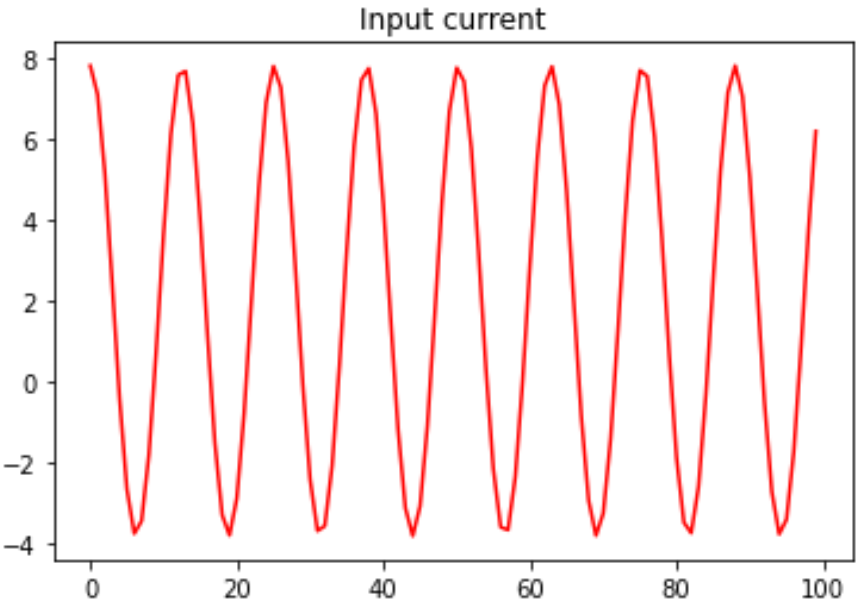


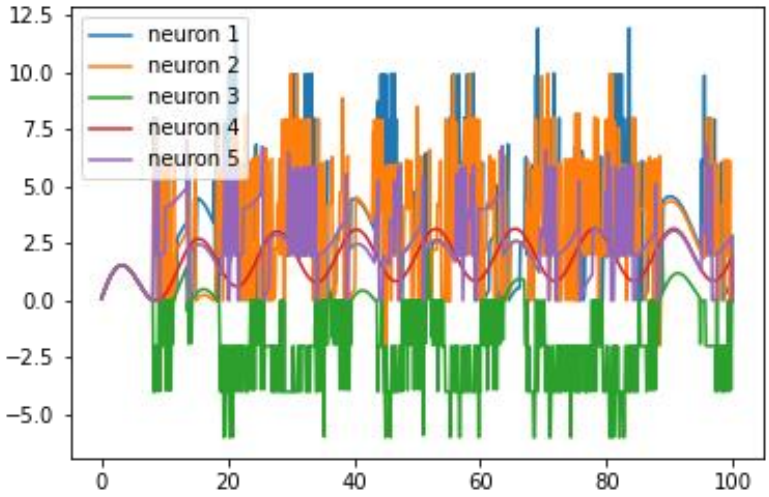


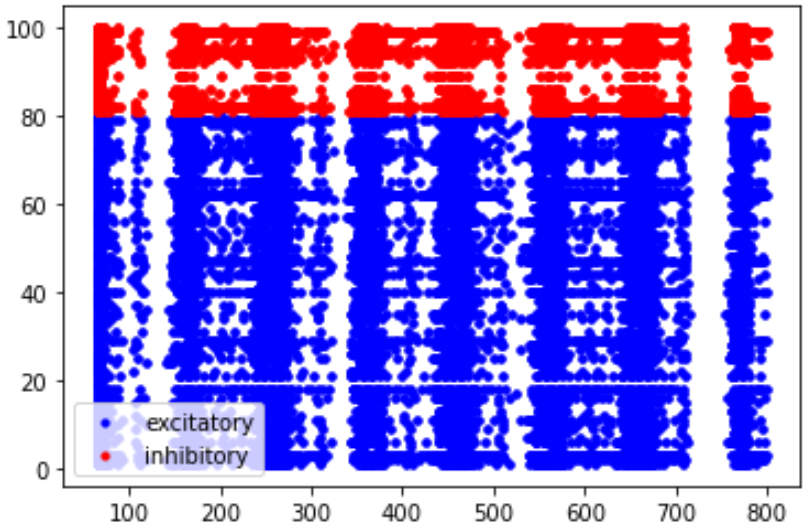

excitatory
inhibitory

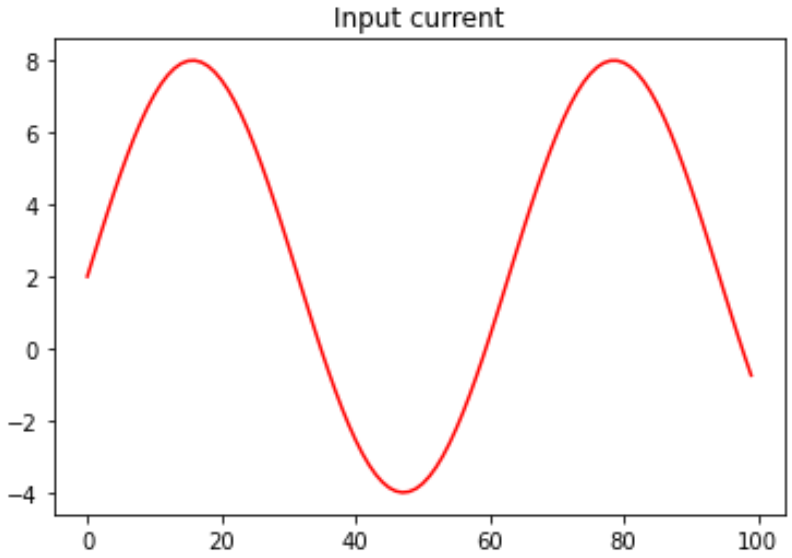

neuron 1
neuron 2
neuron 3
neuron 4
neuron 5

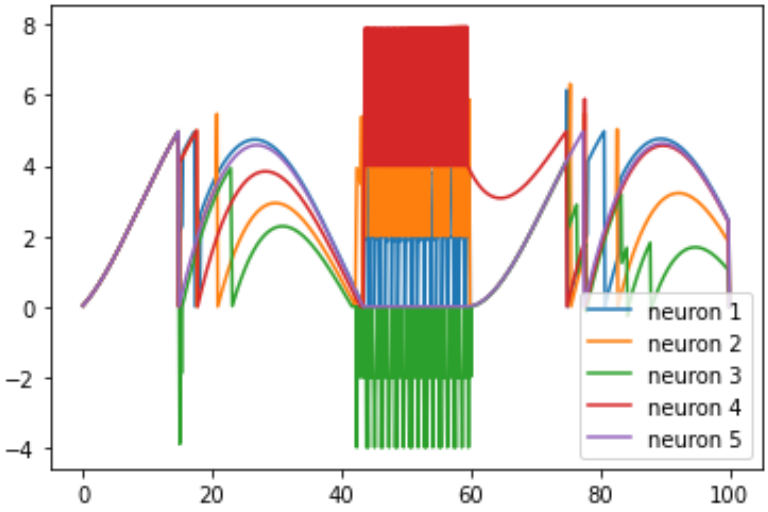

excitatory
inhibitory

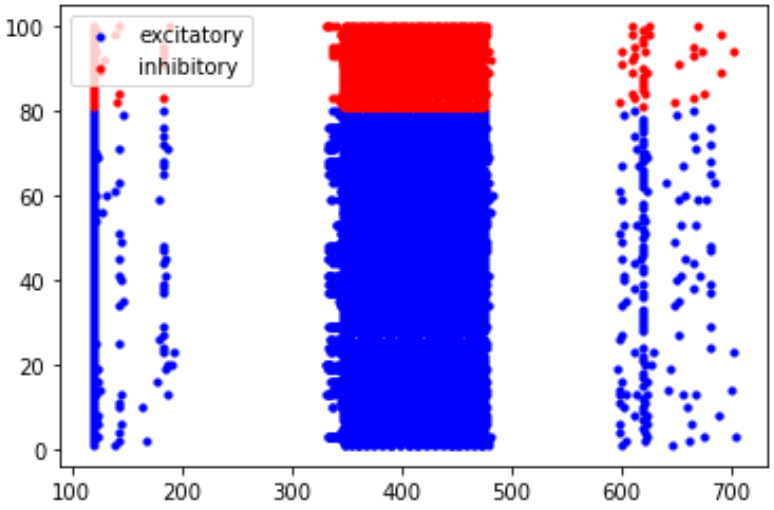

Input current
8
6
4
2
0
−2
−4
0
20
40
60
80
100

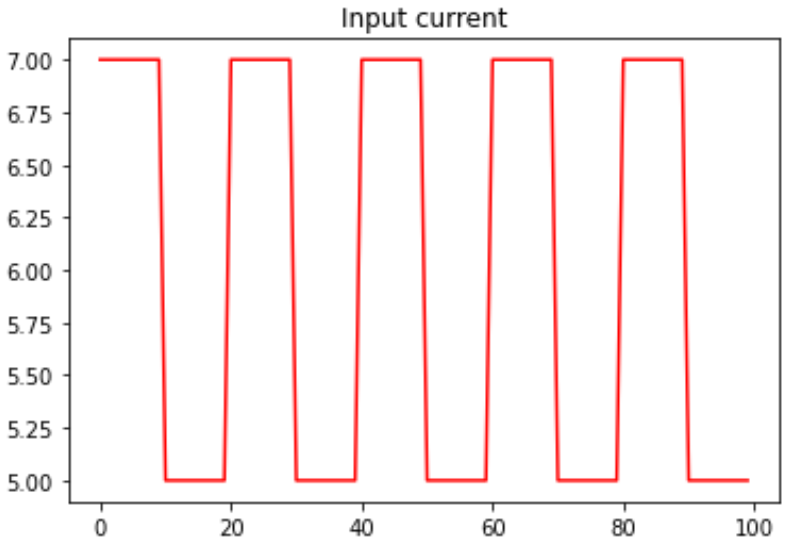

12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
neuron 1
neuron 2
neuron 3
neuron 4
neuron 5
0
20
40
60
80
100

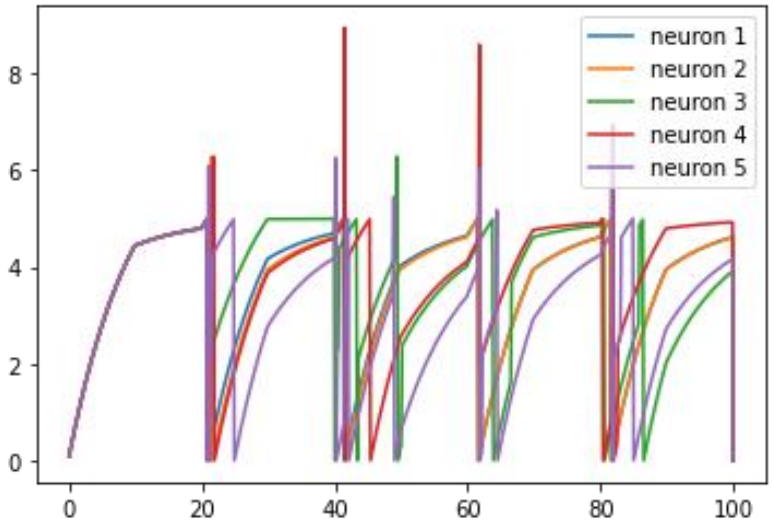

100
80
60
40
20
0
excitatory
inhibitory
100
200
300
400
500
600
700
800

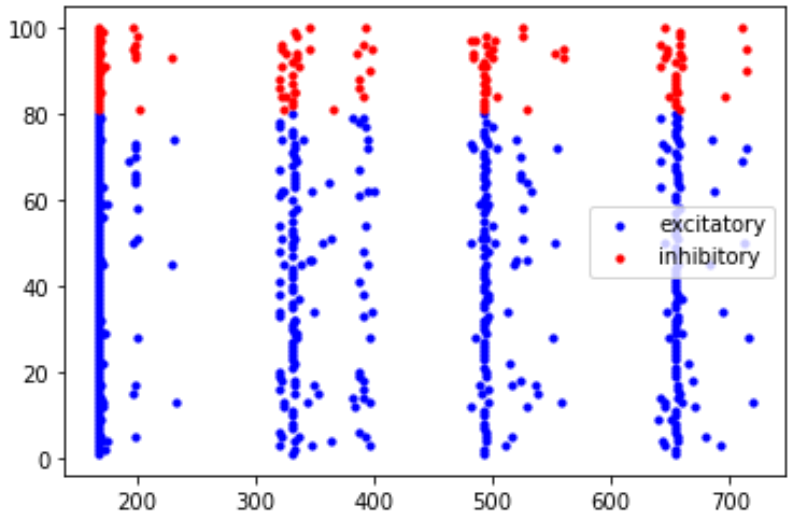

Input current
8
6
4
2
0
−2
−4
0
20
40
60
80
100

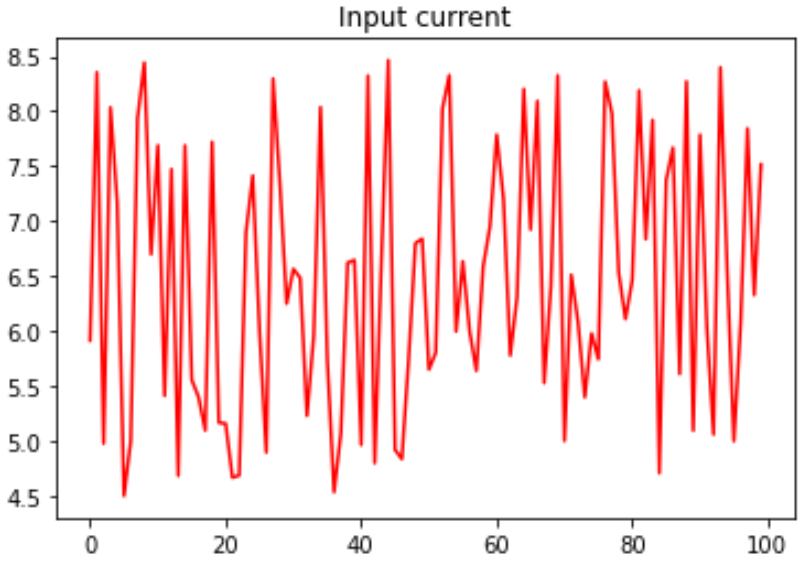

8
6
4
2
0
−2
−4
0
20
40
60
80
100
neuron 1
neuron 2
neuron 3
neuron 4
neuron 5

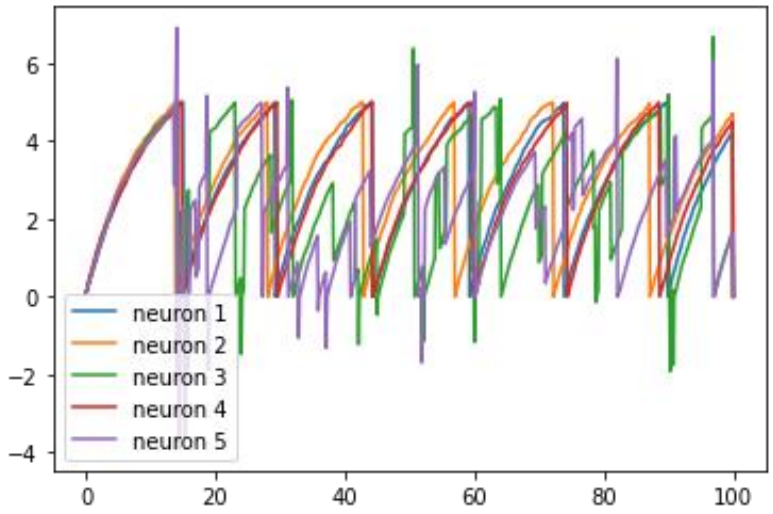

100
80
60
40
20
0
100
200
300
400
500
600
700
excitatory
inhibitory

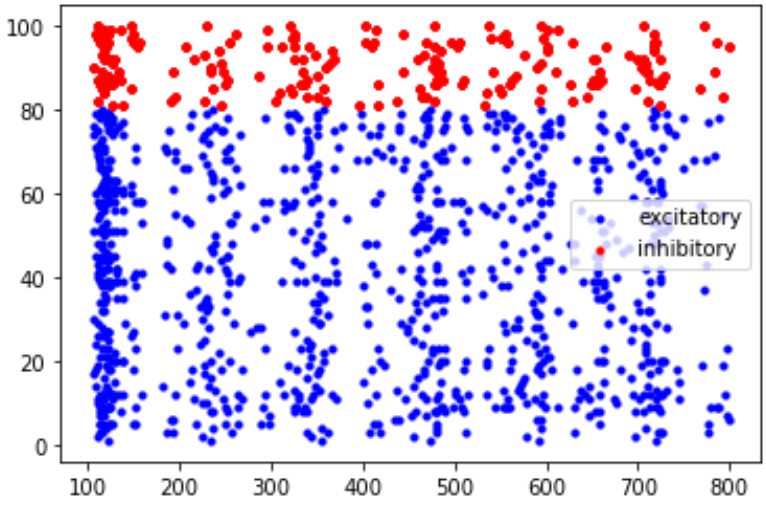

Input current
7.00
6.75
6.50
6.25
6.00
5.75
5.50
5.25
5.00
0
20
40
60
80
100


neuron 1
neuron 2
neuron 3
neuron 4
neuron 5
8
6
4
2
0
0
20
40
60
80
100


excitatory
inhibitory
100
80
60
40
20
0
200
300
400
500
600
700

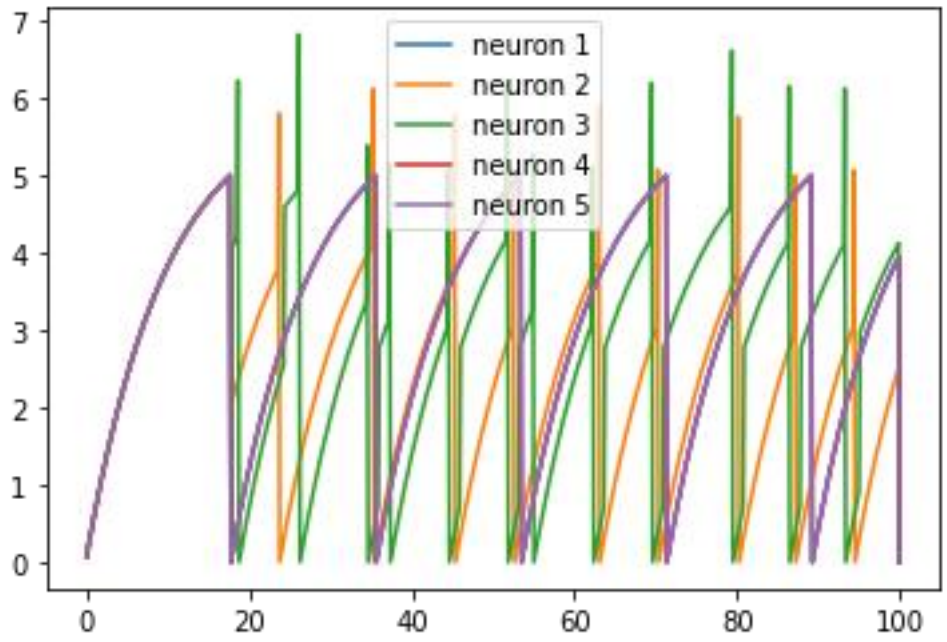

Input current
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
0
20
40
60
80
100


6
4
2
0
−2
−4
0
20
40
60
80
100
neuron 1
neuron 2
neuron 3
neuron 4
neuron 5

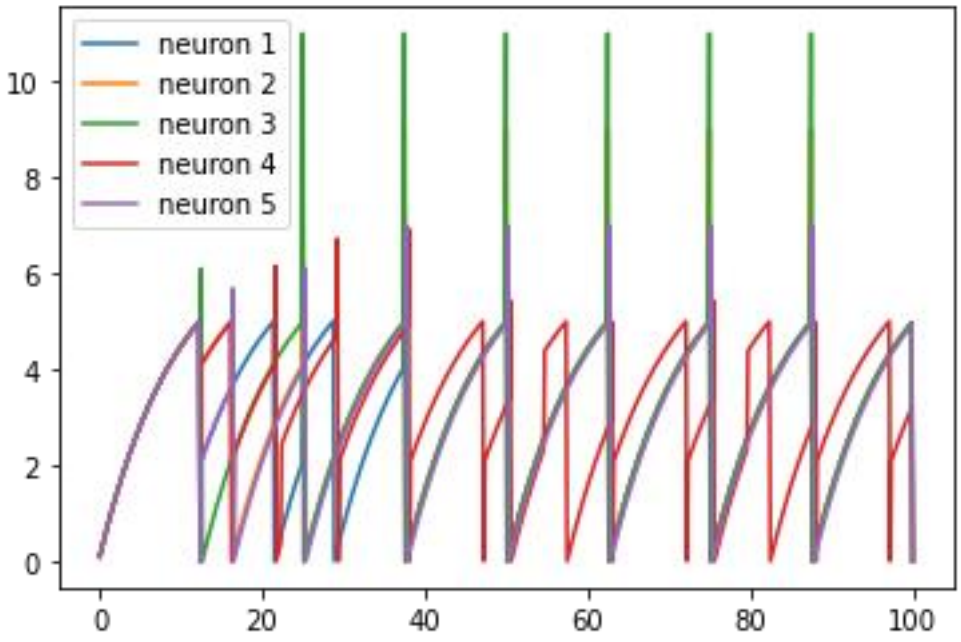

100
80
60
40
20
0
100
200
300
400
500
600
700
800
excitatory
inhibitory

در آخر یک تابع connect اضافه میکنیم تا بتونیم مشخص کنیم که کدام جمعیت های نورونی را میخواهیم به هم دیگر متصل کنیم

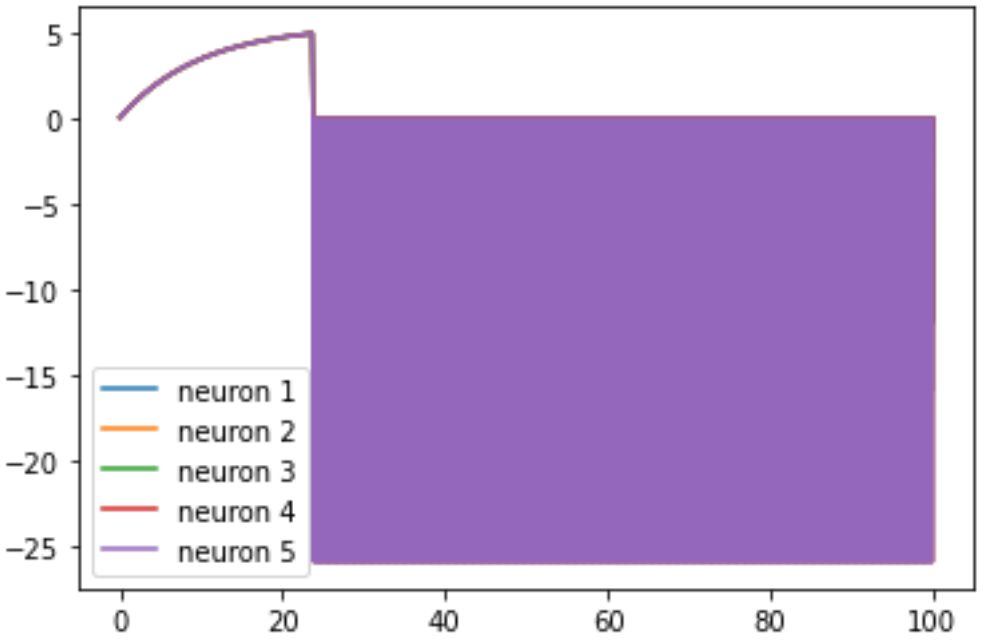
سه جمعیت نورونی خواهیم داشت هر کدام با تعداد 100 نورون که دوتا از این جمعیت های نورونی تحریکی و یکی از آنها مهاری باشد و سپس تاثیرات این جمعیت های نورونی روی یکدیگر را مشاهده میکنیم :



نمودار تغییرات پتانسیل در جمعیت نورونی اول

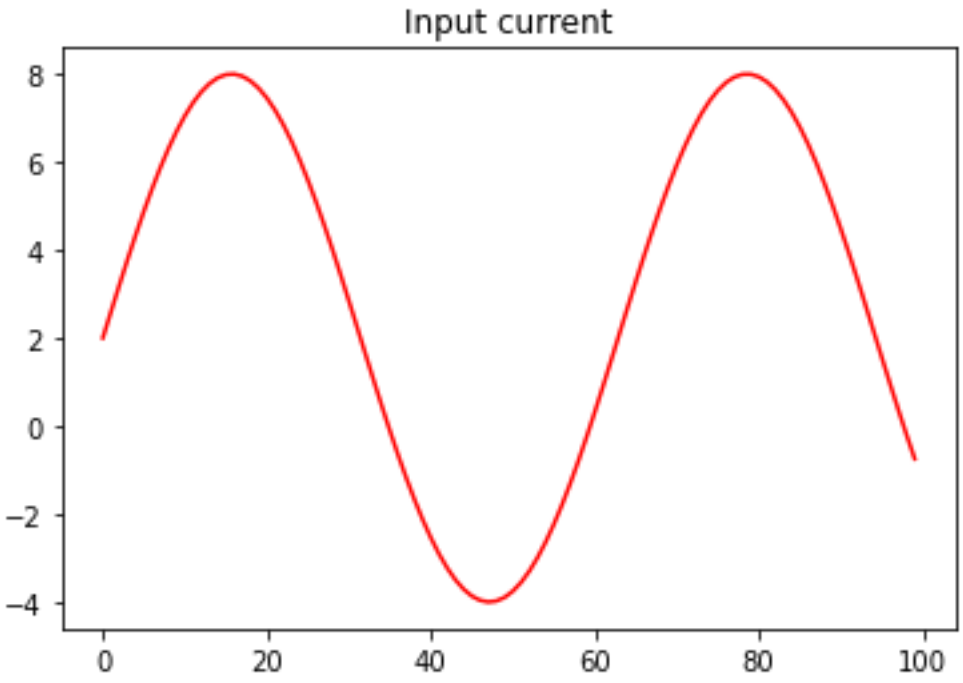


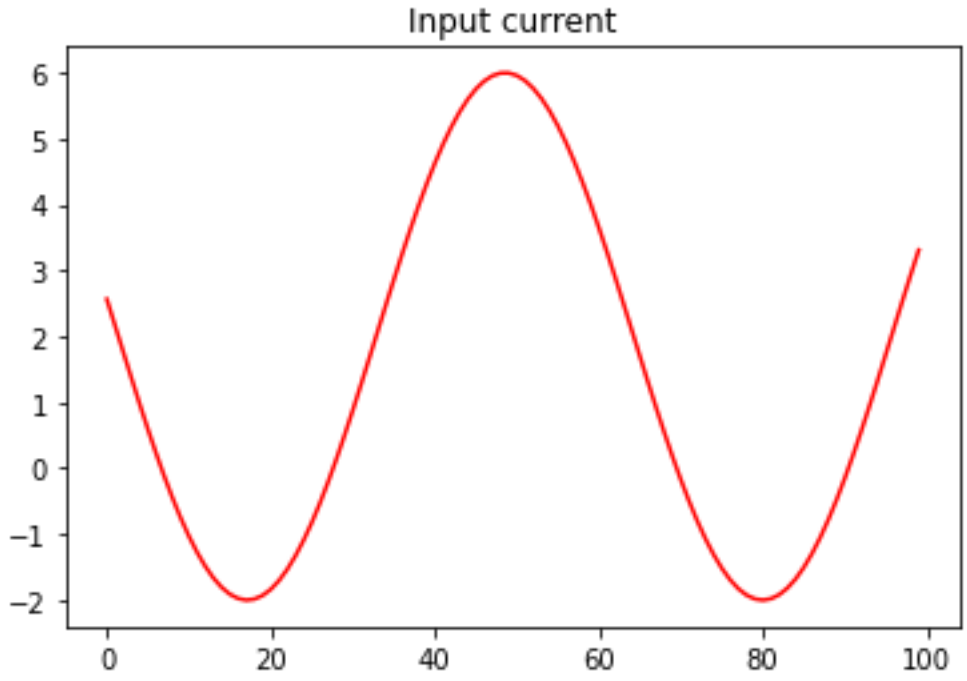
نمودار تغییرات پتاسیل در جمعیت نورونی دوم

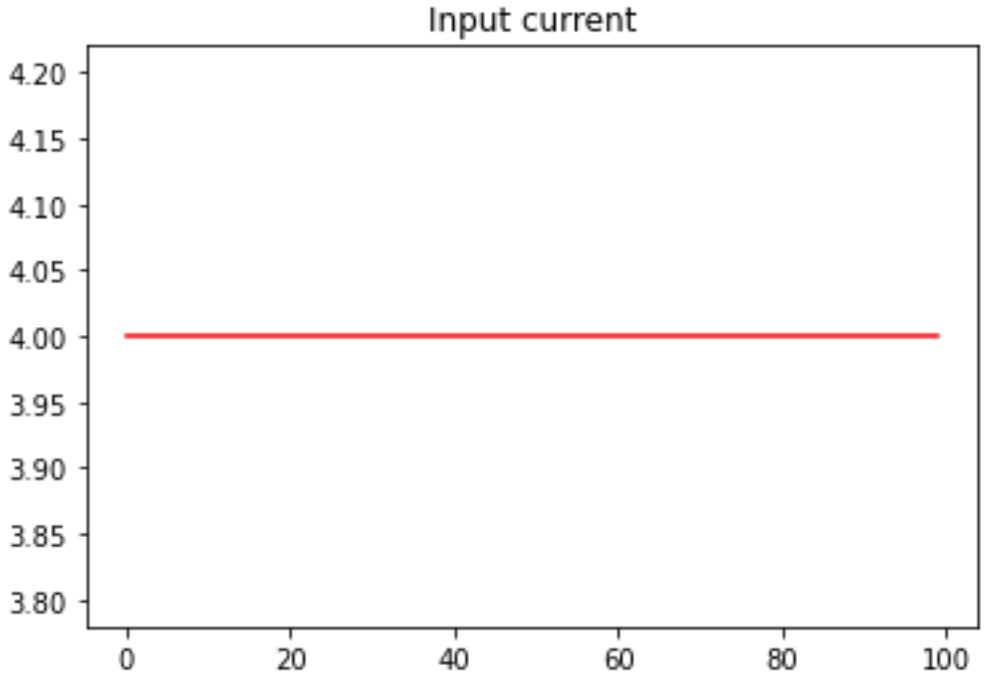


نمودار تغییرات پتانسیل جمعیت نورونی سوم

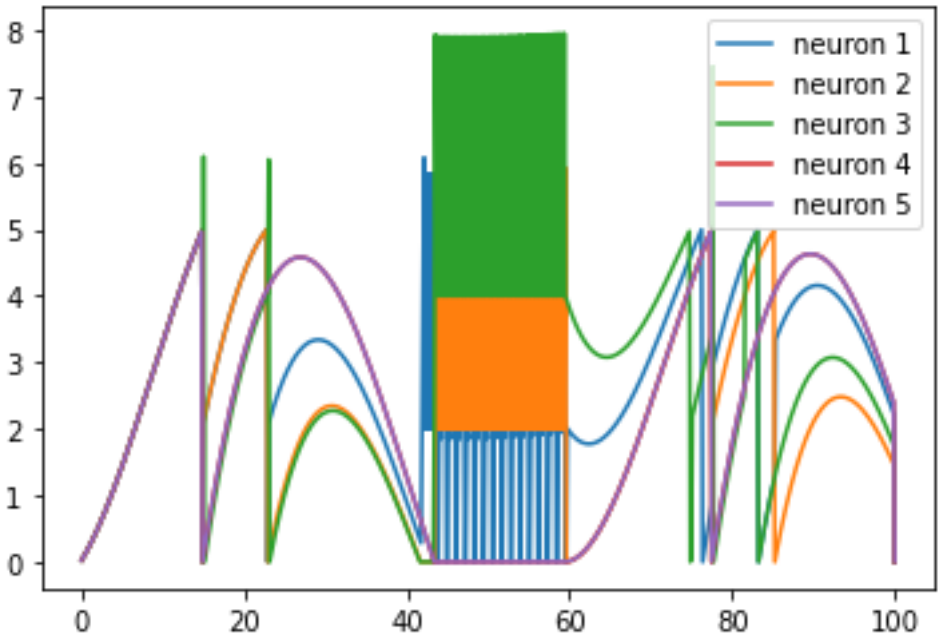
جریانات ورودی به هر یک از جمعیت های نورونی به ترتیب :

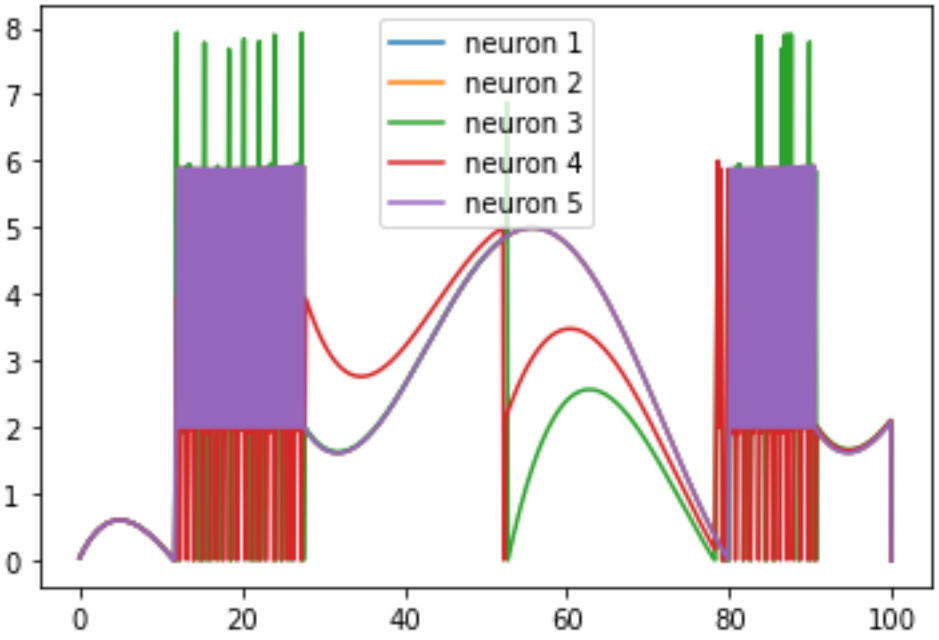


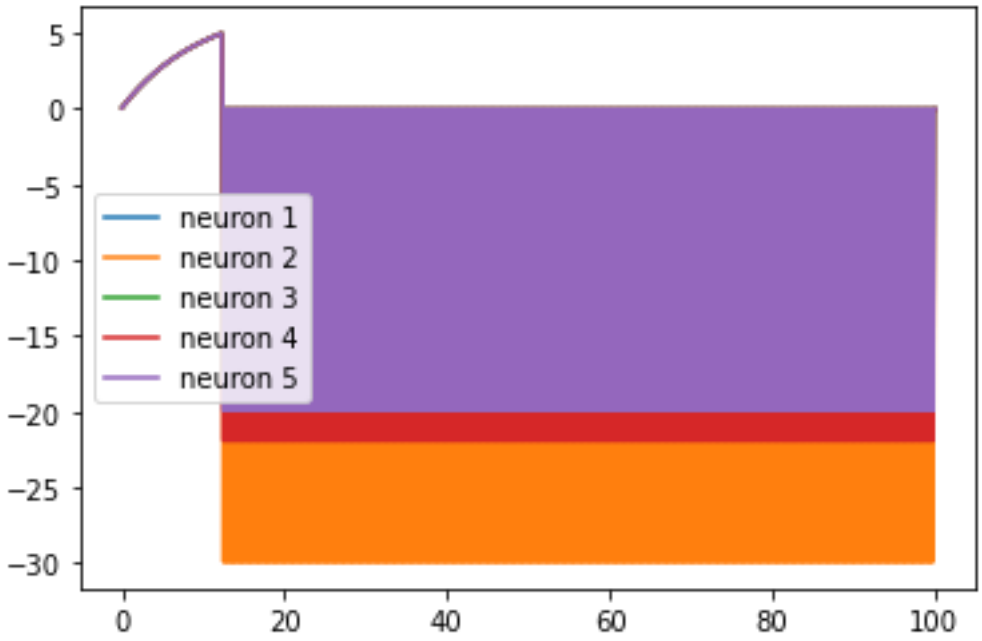




تغییرات آنها به ترتیب :


neuron 1
neuron 2
neuron 3
neuron 4
neuron 5
0
1
2
3
4
5
6
7
8
0
20
40
60
80
100


neuron 1
neuron 2
neuron 3
neuron 4
neuron 5
0
1
2
3
4
5
6
7
8
0
20
40
60
80
100


5
0
−5
−10
−15
−20
−25
−30
0
20
40
60
80
100
neuron 1
neuron 2
neuron 3
neuron 4
neuron 5