PraticoNeuro

Parte 2

Exercício 05

Este exercício irá explorar o mecanismo de sinapses. Tente se lembrar dos exercícios do dia anterior em Brian e crie 3 neurônios integra-dispara. Os neurônios devem ter o mesmo potencial de repouso vrest = -70 mV, mesmo threshold = -60 mV, mesmo reset = -70 mV, tempo refratário = 5 ms, R=100 Mohm e inicialmente corrente de entrada nula I = 0 pA.

Escolha potenciais iniciais distintos em -65 mV, - 80 mV e - 50 mV.

a)

Rode uma simulação por 200 ms e plote os resultados. O que você observa?

b)

Para conectar os neurônios, comece alterando a entrada I para 120 pA em todos os neurônios. Depois, adicione as seguintes linhas de código onde for mais conveniente:

```
J = 1*mV
syn = Synapses(neurons, neurons, on_pre='v += J', delay=10*ms)
syn.connect(i=[0, 1], j=[1, 2])
```

Observe a resposta dos neurônios e tente explicar o que foi feito nas linhas de código acima.

c) Em seguida, faça as seguintes alterações na parte do código onde as sinapses são definidas:

```
J = 1*mV
g = 4
syn_ex = Synapses(neurons, neurons, on_pre='v += J', delay=10*ms)
syn_in = Synapses(neurons, neurons, on_pre='v -= g*J', delay=10*ms)

syn_ex.connect(i=0, j=2)
syn_in.connect(i=1, j=2)
```

Observe o resultado e tende explicar. Tente trocar a linha de conexão de sinapses excitatórias para:

```
syn_ex.connect(i=[0,2], j=[2,0])
```

Neste caso, tente imaginar qual seria o resultado antes de rodar a simulação. Rode e verifique se sua previsão está correta.

Obs: Em python, o exemplar [5:7] equivale ao conjunto [5,7[. Sendo assim, o elemento 7 não está dentro deste conjunto.

d) Você pode também fazer alterações no valor de corrente de cada neurônio, volte no primeiro circuito com sinapses apresentado no exercício **c)** e altere as correntes de forma que

```
neurons.I = 0*pA
neurons[0:2].I = 120*pA
```

Verifique o resultado e tente explicar o que acontece.

Exercício 06

Este exercício irá demonstrar algumas maneiras de visualizar redes de neurônios.

Iniciaremos introduzindo um raster plot.

a)

Utilize o último código que você possui e adicione as seguintes alterações:

- Ao invés de gravar o valor da voltagem em cada momento, recorde somente o valor dos disparos. Para isto, utilize:

```
spike_mon=SpikeMonitor(neurons)
```

- Depois de rodar este código, plote os resultados com:

```
plot(spike_mon.t/ms,spike_mon.i,'x',markersize=1.5)
xlabel('time (ms)')
```

```
ylabel('neuron index')
```

Tente compreender o significado deste gráfico. O raster plot indica o momento em que cada neurônio disparou. Verifique se isto está correto ao plotar o gráfico em conjunto com as voltagens, para tal, utilize:

```
subplot(2,1,1)
plot(spike_mon.t/ms,spike_mon.i,'x',markersize=1.5)
ylabel('neuron index')
xlim(-5,200)
subplot(2,1,2)
plot(state_mon.t/ms,state_mon.v[0]/mV,label="0")
plot(state_mon.t/ms,state_mon.v[1]/mV,label="1")
plot(state_mon.t/ms,state_mon.v[2]/mV,label="2")
xlabel('time (ms)')
xlim(-5,200)
legend()
```

Ao verificar a correlação entre disparos e raster plot, faça todos os neurônios receberem I=120 pA e verifique a mudança no raster plot.

b) Se tudo estiver correto até aqui, você deve ter 3 conexões no total:

neurônio 0 -> neurônio 1

neurônio 1 -> neurônio 2

Tente agora capturar a frequência instantânea da rede. Para tal, você deverá utilizar um outro monitor:

```
rate_mon = PopulationRateMonitor(neurons)
```

Plote o resultado em uma terceira linha da figura que você já está plotando. Use algo como:

```
subplot(3,1,3)
plot(rate_mon.t/ms,rate_mon.rate/Hz)
```

Exercício 07

Este exercício irá guiá-lo a criar uma pequena rede. Faça as adaptações no código anterior. As simulações aqui devem ser de 1000 ms.

Adapte o programa do exercício anterior de forma a ter 10 neurônios. Faremos 5 neurônios excitatórios, 5 inibitórios. Para tal, após criar o grupo de neurônios você pode dividir a rede escrevendo:

```
neuronsE = neurons[:6]

neuronsI = neurons[6:]
```

Para diferenciar os potenciais iniciais, distribua os potenciais aleatoriamente entre -70 mV e -60 mV:

```
neuronsE.v = 'rand()*(threshold-vrest) + vrest'

neuronsI.v ='rand()*(threshold-vrest) + vrest'
```

Para conectar sua pequena rede de 10 neurônios, você deverá escrever as sinapses da seguinte forma:

```
syn_ex = Synapses(neuronsE, neurons, on_pre='v += J', delay=10*ms)

syn_in = Synapses(neuronsI, neurons, on_pre='v -= g*J', delay=10*ms)


syn_ex.connect(condition='i!=j', p = 0.1)

syn_in.connect(condition='i!=j', p = 0.1)
```

Isso faz com que os neurônios de ambas populações se conectem com probabilidade p=0.1, mas evita auto-conexões (autapses).

Em seguida, calcule a frequência de cada neurônio e a frequência média da rede:

```
freq_single = spike_mon.count

freq_global = mean(freq_single)
```

Em qual unidade está o seu resultado?

Você acredita que frequência de disparos dos neurônios se altera por interferência das sinapses? Retire as conexões e execute o programa novamente. Compare a frequência de disparos dos neurônios com e sem as conexões. E a da rede?

Exercício 08

a)

Os próximos exercícios serão um pouco mais desafiantes computacionalmente. Vamos implementar uma rede muito próxima a rede de Brunel 2000. Ele construiu uma rede com 10.000 neurônios excitatórios, 2.500 inibitórios. Utilize o código anterior e altere seus parâmetros. Além disto, os seguintes parâmetros deverão ser alterados para adaptar com a rede de Brunel 2000:

```
tau = 20*ms
n = 12500
vrest = 0*mV
delay = 1.5*ms
J = 0.1*mV
g = 4
refractory = 2*ms
threshold = 10*mV
vext = 2*(threshold)
```

Assim colocando todas as outras definições de parâmetros no começo da simulação (boas práticas de programação).

Observe que o eixo-y na figura de seu artigo (mostrada na apresentação) utiliza vext ao invés da definição R*I que estivemos utilizando. Aqui definimos vext como função do threshold como ele fez.

Altere o restante do codigo, rode uma simulação de 1000 ms e plote somente os 800 ms finais em ambos raster plot e atividade da rede:

```
subplot(2,1,1)
plot(spike_mon.t/ms,spike_mon.i,'.',markersize=0.5)
ylabel('neuron index')
ylim(0,100)
xlim(200,1000)
subplot(2,1,2)
plot(rate_mon.t/ms,rate_mon.rate/Hz)
xlim(200,1000)
```

b)

Tendo a rede implementada, você deve agora escolher parâmetros que criem as atividades do tipo AI, SR e SI slow osc. Observe que esta rede é uma adaptação da rede Brunel 2000 e não uma exata réplica.

Para cada comportamento, calcule também a frequência global da rede e observe o que acontece com ela.

Exercício 09

Ostojic 2014 re-utilizou a rede de Brunel 2000 e explorou um novo comportamento assíncrono ao variar o valor de J. Ele cria uma rede homogênea e faz com que os

disparos sejam heterogêneos. O comportamento é dominado por inibição (g=5).
Altere seu código para que ele crie os dois tipos de comportamentos assíncronos para J=0.1 mV e J=0.8 mV. Você deverá plotar raster plot e atividade da rede para ambos na mesma figura. Isto significa que você fará duas simulações. Utilize o seu próprio julgamento para encontrar a melhor solução para este problema.

Avaliação do Curso:

1- Você tem experiência prévia com programação? Qual linguagem?

2- Você tem experiência em programação orientada à objeto?

3- Já programava em Python?

4- Como você avalia a progressão do grau de dificuldade entre os exercícios? Retiraria algum exercício? Colocaria algum exercício extra? Qual?

5- Como você avalia o grau de dificuldade do curso?

6- Como você avalia a utilidade do curso para você?

7- O que você mudaria no curso/apresentações/apostila?
