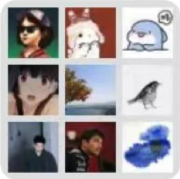


群聊: 24神经建模基础

该二维码7天内(10月2日前)有效，重新进入将更新

Programming of Hodgkin-Huxley (HH) model

彭相源

2024.09.26

The mathematic expression of the HH model

$$
\begin{cases}
c\frac{\mathrm{d}V}{\mathrm{d}t} = -\bar{g}_{\mathrm{Na}} m^3 h\left(V - E_{\mathrm{Na}}\right) - \bar{g}_{\mathrm{K}} n^4\left(V - E_{\mathrm{K}}\right) - \bar{g}_{\mathrm{L}}\left(V - E_{\mathrm{L}}\right) + I_{\mathrm{ext}}, \\
\frac{\mathrm{d}n}{\mathrm{d}t} = \phi\left[\alpha_n(V)(1-n) - \beta_n(V)n\right] \\
\frac{\mathrm{d}m}{\mathrm{d}t} = \phi\left[\alpha_m(V)(1-m) - \beta_m(V)m\right], \\
\frac{\mathrm{d}h}{\mathrm{d}t} = \phi\left[\alpha_h(V)(1-h) - \beta_h(V)h\right],
\end{cases}
$$

V: the membrane potential

n: activation variable of the K^+ channel

m: activation variable of the Na^+ channel

h: inactivation variable of the Na^+ channel

$$
\alpha_n(V) = \frac{0.01(V+55)}{1-\exp\left(-\frac{V+55}{10}\right)}, \quad \beta_n(V) = 0.125\exp\left(-\frac{V+65}{80}\right),
$$

$$
\alpha_h(V) = 0.07\exp\left(-\frac{V+65}{20}\right), \quad \beta_h(V) = \frac{1}{\left(\exp\left(-\frac{V+35}{10}\right)+1\right)},
$$

$$
\alpha_m(V) = \frac{0.1\left(V+40\right)}{1-\exp\left(-(V+40)/10\right)}, \quad \beta_m(V) = 4\exp\left(-(V+65)/18\right).
$$

α_x and β_x: voltage-dependent transition rates

$$
\phi = Q_{10}^{(T-T_{\mathrm{base}})/10}
$$

Programming of HH model in BrainPy

1.Define HH Model `class`

2. Initialization
   - parameters
   - variables
   - integral function

3. Define the derivative function

4. Complete the `update()` function

Programming of HH model in BrainPy

- Define HH Model `class`
  - Inherit bp.dyn.NeuGroup

```
class NeuGroup(DynamicalSystem):
  """Base class to model neuronal groups.
```

```
import brainpy as bp
import brainpy.math as bm

class HH_neurons(bp.dyn.NeuGroup):
```

Programming of HH model in BrainPy

- Define HH Model `class`
  - Inherit bp.dyn.NeuGroup

- Initialization

```
class HH_neurons(bp.dyn.NeuGroup):

    #初始化
    def __init__(self, size,
                 ENa=50., gNa=120.,
                 EK=-77., gK=36.,
                 EL=-54.387, gL=0.03,
                 V_th=20., C=1., T=6.3):
        super().__init__(size=size)
```

Programming of HH model in BrainPy

- Define HH Model `class`
  - Inherit bp.dyn.NeuGroup

- Initialization
  - parameters

```python
class HH_neurons(bp.dyn.NeuGroup):

    #初始化
    def __init__(self, size,
                 ENa=50., gNa=120.,
                 EK=-77., gK=36.,
                 EL=-54.387, gL=0.03,
                 V_th=20., C=1., T=6.3):
        super().__init__(size=size)

        #定义神经元参数
        self.ENa = ENa
        self.gNa = gNa
        self.EK = EK
        self.gK = gK
        self.EL = EL
        self.gL = gL
        self.V_th = V_th
        self.C = C
        self.T_base = 6.3
        self.Q10 = 3.
        self.phi = self.Q10 ** ((T - self.T_base)/10)
```

Programming of HH model in BrainPy

- Define HH Model `class`
  - Inherit bp.dyn.NeuGroup

- Initialization
  - parameters
  - variables

```python
class HH_neurons(bp.dyn.NeuGroup):

    #初始化
    def __init__(self, size,
                 ENa=50., gNa=120.,
                 EK=-77., gK=36.,
                 EL=-54.387, gL=0.03,
                 V_th=20., C=1., T=6.3):
        super().__init__(size=size)

        #定义神经元参数
        self.ENa = ENa
        self.gNa = gNa
        self.EK = EK
```

```python
#定义神经元变量
self.V = bm.Variable(-70.68 * bm.ones(self.num))
self.m = bm.Variable(0.0266 * bm.ones(self.num))
self.h = bm.Variable(0.772 * bm.ones(self.num))
self.n = bm.Variable(0.235 * bm.ones(self.num))
self.input = bm.Variable(bm.zeros(self.num))
self.spike = bm.Variable(bm.zeros(self.num, dtype=bool))
self.t_last_spike = bm.Variable(bm.ones(self.num) * -1e7)
```

Programming of HH model in BrainPy

- Define HH Model `class`
  - Inherit bp.dyn.NeuGroup

- Initialization
  - parameters
  - variables
  - integral function

```python
class HH_neurons(bp.dyn.NeuGroup):

    #初始化
    def __init__(self, size,
                      ENa=50., gNa=120.,
                      EK=-77., gK=36.,
                      EL=-54.387, gL=0.03,
                      V_th=20., C=1., T=6.3):
        super().__init__(size=size)

        #定义神经元参数
        self.ENa = ENa
        self.gNa = gNa
        self.EK = EK
```

```python
#定义神经元变量
self.V = bm.Variable(-70.68 * bm.ones(self.num))
self.m = bm.Variable(0.0266 * bm.ones(self.num))
self.h = bm.Variable(0.772 * bm.ones(self.num))
self.n = bm.Variable(0.235 * bm.ones(self.num))
self.input = bm.Variable(bm.zeros(self.num))
self.spike = bm.Variable(bm.zeros(self.num, dtype=bool))
self.t_last_spike = bm.Variable(bm.ones(self.num) * -1e7)
```

```python
#定义积分函数
self.integral = bp.odeint(f=self.derivative,method='exp_auto')
```

Programming of HH model in BrainPy

- Define HH Model `class`
  - Inherit bp.dyn.NeuGroup

- Initialization
  - parameters
  - variables
  - Integral function

- Derivative function

$$
\begin{cases}
c\dfrac{\mathrm{d}V}{\mathrm{d}t} = -\bar{g}_{\mathrm{Na}}m^3h\left(V-E_{\mathrm{Na}}\right) - \bar{g}_{\mathrm{K}}n^4\left(V-E_{\mathrm{K}}\right) - \bar{g}_{\mathrm{L}}\left(V-E_{\mathrm{L}}\right) + I_{\mathrm{ext}}, \\
\dfrac{\mathrm{d}n}{\mathrm{d}t} = \phi\left[\alpha_n(V)(1-n) - \beta_n(V)n\right] \\
\dfrac{\mathrm{d}m}{\mathrm{d}t} = \phi\left[\alpha_m(V)(1-m) - \beta_m(V)m\right], \\
\dfrac{\mathrm{d}h}{\mathrm{d}t} = \phi\left[\alpha_h(V)(1-h) - \beta_h(V)h\right],
\end{cases}
$$

```python
@property
def derivative(self):
    return bp.JointEq(self.dV,self.dm,self.dh,self.dn)

def dm(self,m,t,V):
    alpha = 0.1 * ( V + 40) / (1 - bm.exp(-(V + 40) /  10))
    beta = 4.0 * bm.exp(-(V + 65) / 18)
    dmdt = alpha * (1 - m) - beta * m
    return self.phi * dmdt

def dh(self,h,t,V):
    alpha = 0.07 * bm.exp(-(V + 65) / 20)
    beta = 1 / (1 + bm.exp(-(V + 35) / 10))
    dhdt = alpha * (1 - h) - beta * h
    return self.phi * dhdt


def dn(self, n, t, V):
    alpha = 0.01 * (V + 55) / (1 - bm.exp(-(V + 55) / 10))
    beta = 0.125 * bm.exp(-(V + 65) / 80)
    dndt = alpha * (1 - n) - beta * n
    return self.phi * dndt

def dV(self, V, t, m, h, n):
    I_Na = (self.gNa * m**3 * h) * (V - self.ENa)
    I_K = (self.gK * n**4) * (V - self.EK)
    I_leak = self.gL * (V - self.EL)
    dVdt = (-I_Na - I_K - I_leak + self.input) / self.C
    return dVdt
```

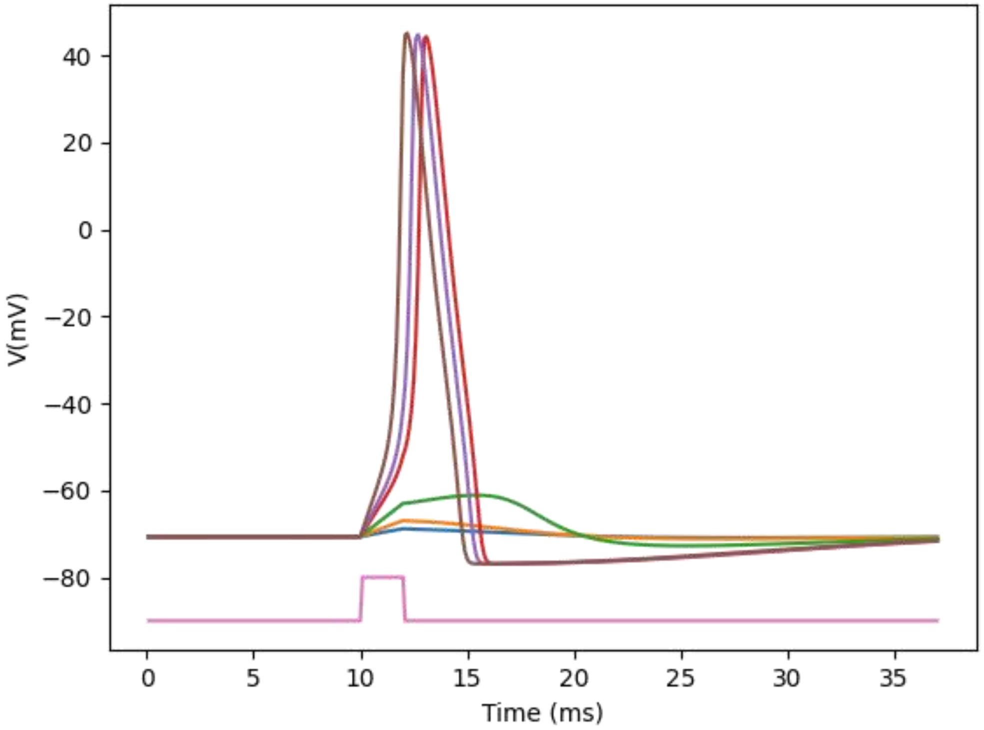
Programming of HH model in BrainPy

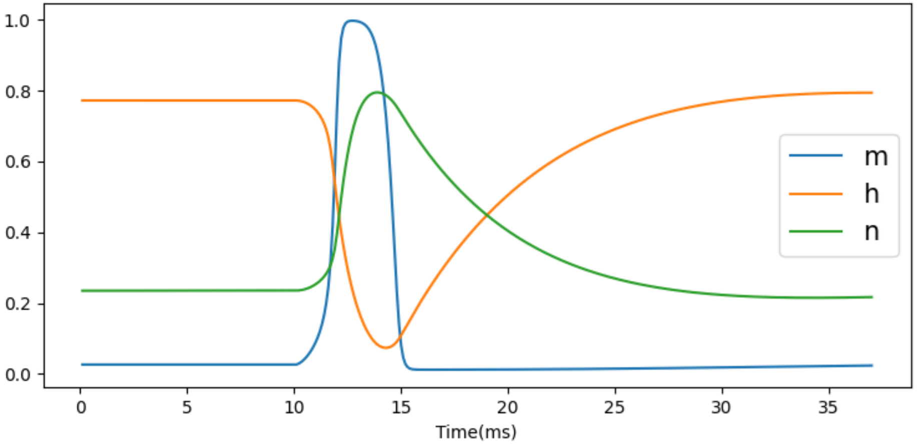
- Define HH Model `class`
  - Inherit bp.dyn.NeuGroup
- Initialization
  - parameters
  - variables
  - Integral function
- Derivative function
- update() function

```python
#更新函数
def update(self):
    #tdi----shared parameters :time t,dt, iteration i
    t = bp.share['t']
    dt = bp.share['dt']

    #计算当前时刻变量的值
    V,m,h,n = self.integral(self.V, self.m, self.h, self.n, t, dt=dt)

    #判断是否产生动作电位
    self.spike.value = bm.logical_and(self.V<self.V_th, V>=self.V_th)

    #发放时间
    self.t_last_spike.value = bm.where(self.spike, t, self.t_last_spike)

    #更新变量的值
    self.V.value = V
    self.m.value = m
    self.h.value = h
    self.n.value = n

    #重置输入
    self.input[:]= 0.
```

Simulations

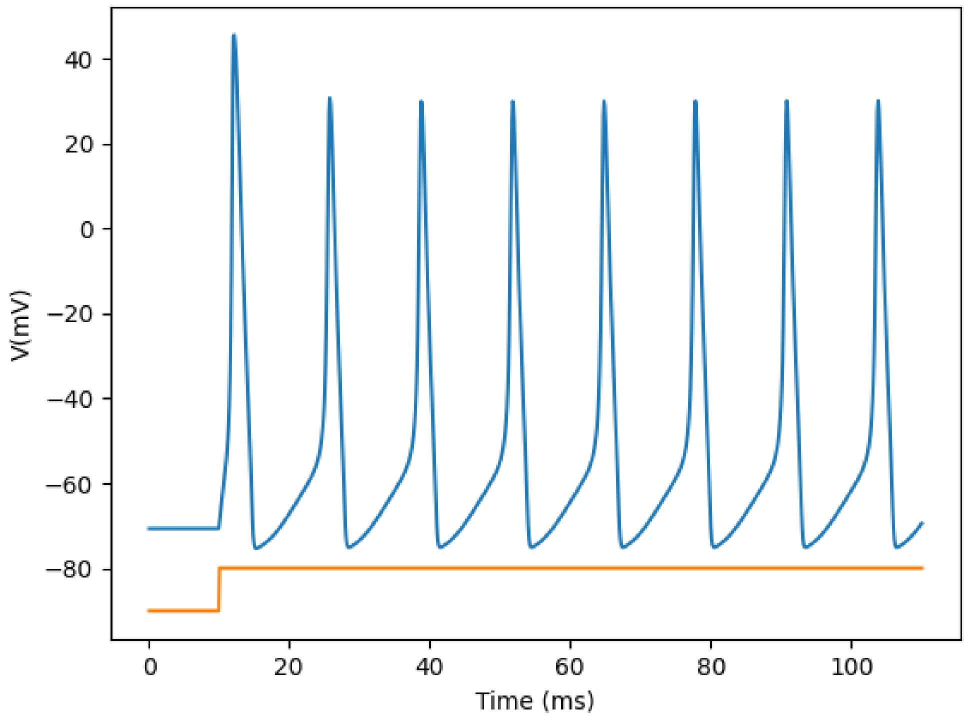
```
current , length = bp.inputs.section_input(values=[0.,bm.asarray([1.,2.,4.,8.,10.,15.]),0.],
                                            durations=[10,2,25],
                                            return_length=True)

hh_neurons = HH_neurons(current.shape[1])

runner = bp.dyn.DSRunner(hh_neurons,monitors=['V','m','h','n'],inputs=['input',current,'iter'])

runner.run(length)

import numpy as np
import matplotlib.pyplot as plt

bp.visualize.line_plot(runner.mon.ts,runner.mon.V,ylabel='V(mV)',plot_ids=np.arange(current.shape[1]))

plt.plot(runner.mon.ts, bm.where(current[:,-1]>0,10,0) - 90)  #

plt.figure()

plt.plot(runner.mon.ts,runner.mon.m[:,-1])
plt.plot(runner.mon.ts,runner.mon.h[:,-1])
plt.plot(runner.mon.ts,runner.mon.n[:,-1])
plt.legend(['m','h','n'])
plt.xlabel('Time(ms)')
```

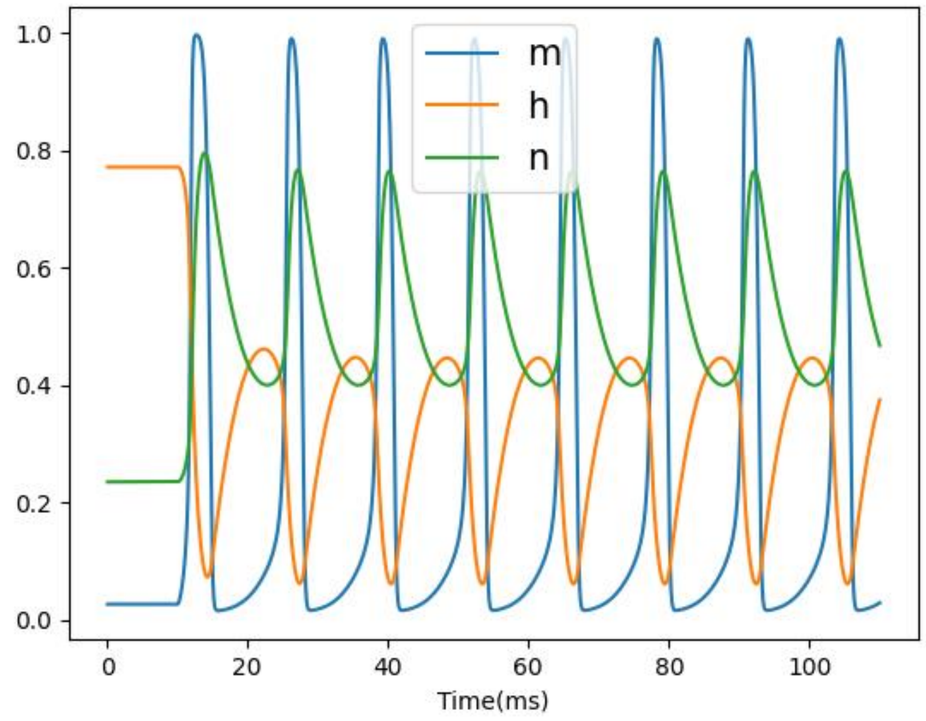




Simulations

Periodic firing





NIP LAB

Homework 1

- 作业内容
  - 根据授课内容，编写HH模型代码
  - 任选一个HH模型中的参数，观察该参数的变化对HH模型的动作电位有何影响

- 提交期限
  - 10.16日24:00前

- 提交方式
  - 将代码和报告打包在同一个以姓名命名的文件夹内，上传至教学网。