Question 1. Simulation of a multi-compartment model of a passive neurite.

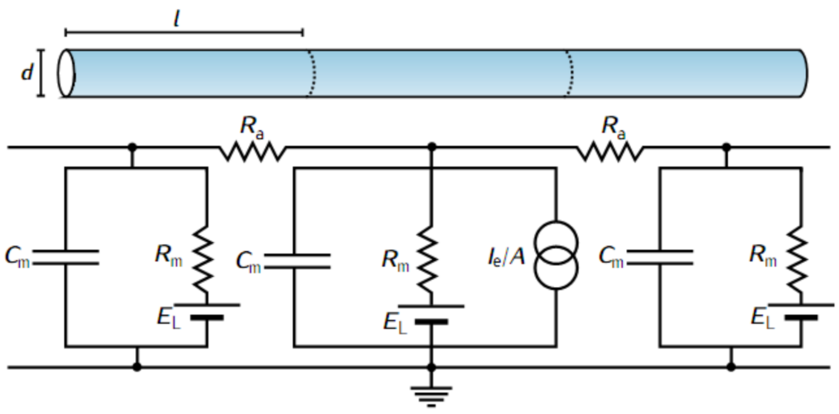

Figure 1: Multi-compartment model.

1.1 In the previous homework we solved for the voltage responses of a two-compartment model. Recall that

$$C_m \frac{dV_1(t)}{dt} = \frac{E_m - V_1(t)}{R_m} + \frac{V_2(t) - V_1(t)}{R_a} + I_e(t)$$
$$C_m \frac{dV_2(t)}{dt} = \frac{V_1(t) - V_2(t)}{R_a} + \frac{E_m - V_2(t)}{R_m}$$

We need to generalise this to an n-compartment model where $n \geqslant 2$ and arbitrary current $I_e(t)$ can be injected into any of the n compartments.

Following the diagram in Figure 1, we observe that

$$I_{e,2}(t) = C_m \frac{dV_2(t)}{dt} + \frac{V_2(t) - E_L}{R_m} + \frac{V_2(t) - V_3(t)}{R_a} + \frac{V_2(t) - V_1(t)}{R_a}$$

Where the compartments are labelled 1-3 starting from the right hand side.

Rearranging yeilds

$$C_m \frac{dV_2(t)}{dt} = \frac{E_L - V_2(t)}{R_m} + \frac{V_3(t) - V_2(t)}{R_a} + \frac{V_1(t) - V_2(t)}{R_a} + I_{e,2}(t)$$

Note that we have the middle compartment 2 expressed in terms of its neighbouring compartments 1 & 3. This can thus be generalised as

$$\frac{dV_j(t)}{dt} = \frac{E_L - V_j(t)}{\tau_m} + \frac{1}{C_m}\left(\frac{V_{j+1}(t) - V_j(t)}{R_a} + \frac{V_{j-1}(t) - V_j(t)}{R_a}\right) + \frac{1}{C_m} I_{e,j}(t)$$

The above equation is known as the fundamental equation of a compartmental model. For the compartments without any current injected the response will look the same but lacking the current term and dependent on the direction of the current flow. Thus, if we use the same numbering scheme, then for compartment i where $0 < i < j$, we have

$$\frac{dV_i(t)}{dt} = \frac{E_L - V_i(t)}{\tau_m} + \frac{1}{C_m}\left(\frac{V_{i-1}(t) - 2V_i(t) + V_{i+1}(t)}{R_a}\right)$$

Similarly, for compartment i where $j < i < N$ where N is the number of compartments, we have

$$\frac{dV_i(t)}{dt} = \frac{E_L - V_i(t)}{\tau_m} + \frac{1}{C_m}\left(\frac{V_{i+1}(t) - 2V_i(t) + V_{i-1}(t)}{R_a}\right)$$

Nota bene for the terminal compartments (i.e. when $i = 0$ or $i = N$ we have to make use of the boundary conditions as shown in Figure 2. For the killed end (Dirichlet)

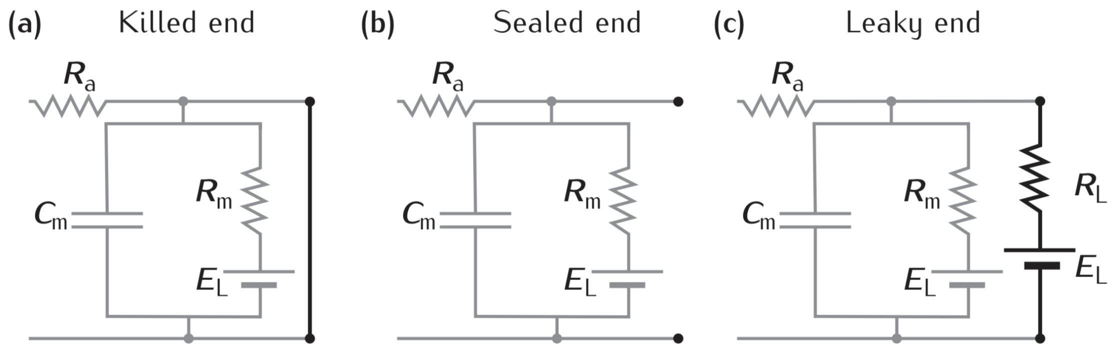

Figure 2: Boundary conditions for the terminal compartments.

boundary condition, the neurite's intracellular environment is in contact with the extracellular medium, and hence the potential V_i for that compartment is set to 0, for it is grounded. For the sealed end (Neumann) boundary condition, we assume that the resistance is so high that no current can pass through the end. Since the axial current is proportional to the gradient of the membrane potential, no current implies zero potential gradient.

$$\frac{V_{i+1} - V_i}{R_a} = C_m\frac{dV_i}{dt} + \frac{V_i - E_L}{R_m}$$
$$\implies \frac{dV_i}{dt} = \frac{E_L - V_i}{\tau_m} + \frac{V_{i+1} - Vi}{C_m R_a}$$

1.2 Equation 1 can be approximated with the forward Euler method as shown below

$$\frac{V_j(t+\Delta t) - V_j(t)}{\Delta t} = \frac{E_L - V_j(t)}{\tau_m} + \frac{1}{C_m}\left(\frac{V_{j+1}(t) - V_j(t)}{R_a} + \frac{V_{j-1}(t) - V_j(t)}{R_a}\right) + \frac{1}{C_m}I_{e,j}(t)$$

$$\implies V_j(t+\Delta t) = V_j(t) + \Delta t\left(\frac{E_L - V_j(t)}{\tau_m} + \frac{1}{C_m}\left(\frac{V_{j+1}(t) - V_j(t)}{R_a} + \frac{V_{j-1}(t) - V_j(t)}{R_a}\right)\right) + \frac{\Delta t}{C_m}I_{e,j}(t)$$

Now, we simulate this with the step current given below

$$I_e(j,t) = \begin{cases} 0, & (t < t_e) \vee (j \neq j_e) \\ 10\,\mathrm{pA}, & (t_e \leqslant t) \wedge (j = j_e) \end{cases} \tag{1}$$

We assume $N = 50$ compartments where the first compartment is terminated as a 'sealed end' and the last compartment is terminated as a 'killed end'. We also assume the following electrical properties

a) Membrane capacitance $C_m = 62.8\,\mathrm{pF}$

b) Membrane resistance $R_m = 1.59\,\mathrm{G\Omega}$

c) Axial resistance $R_a = 0.0318\,\mathrm{G\Omega}$

d) $E_L = E_m = 0\,\mathrm{V}$

The results of the scenario mentioned above appear in Figure 3.

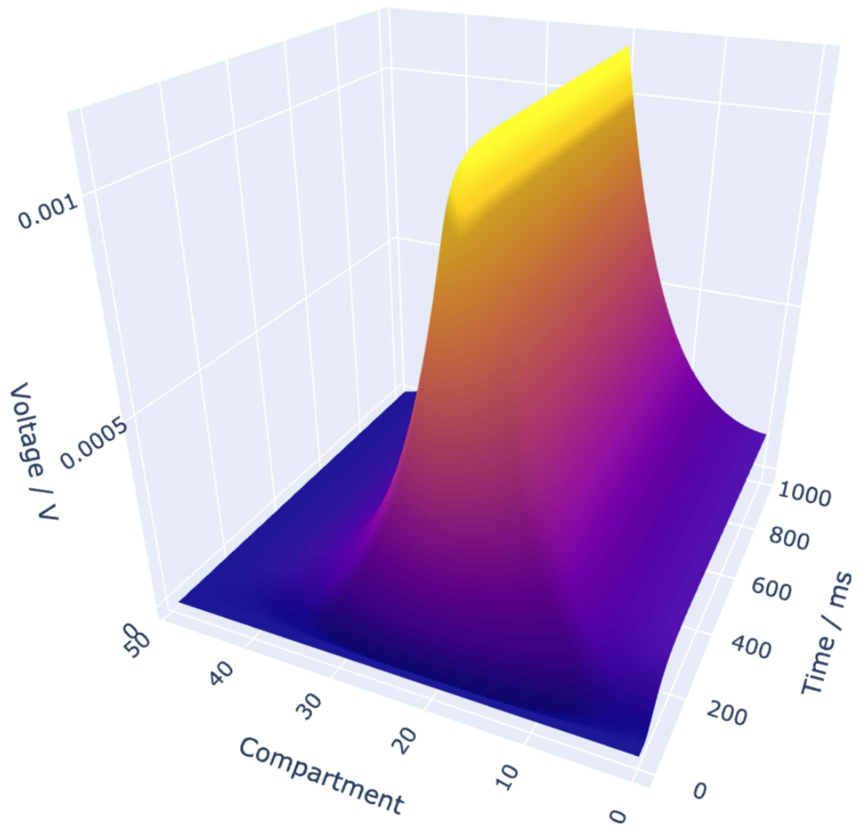

Figure 3: Voltage response over time for 50 compartments.

1.3 As regards the functional form of the spatially-dependent stationary solution $V(j,\infty)$, it decays exponentially while travelling through each compartment away from the current injection site. The voltage decay constant is larger for the current travelling towards the 'sealed end', and hence the voltage decreases slower, as opposed to the other side where it effectively reaches zero.

1.4 Next, the same multi-compartment model was simulated in response to the rectangular current input

$$I_e(j,t) = \begin{cases} 0, & (t < t_e) \vee (t_s \leqslant t) \vee (j \neq j_e) \\ I_0, & (t_e \leqslant t < t_s) \wedge (j = j_e) \end{cases} \tag{2}$$

with $I_0 = 100\,\text{pA}$, $j_e = 14$, $t_e = 20\,\text{ms}$, and $t_s = 200\,\text{ms}$. The results can be seen in Figure 4.

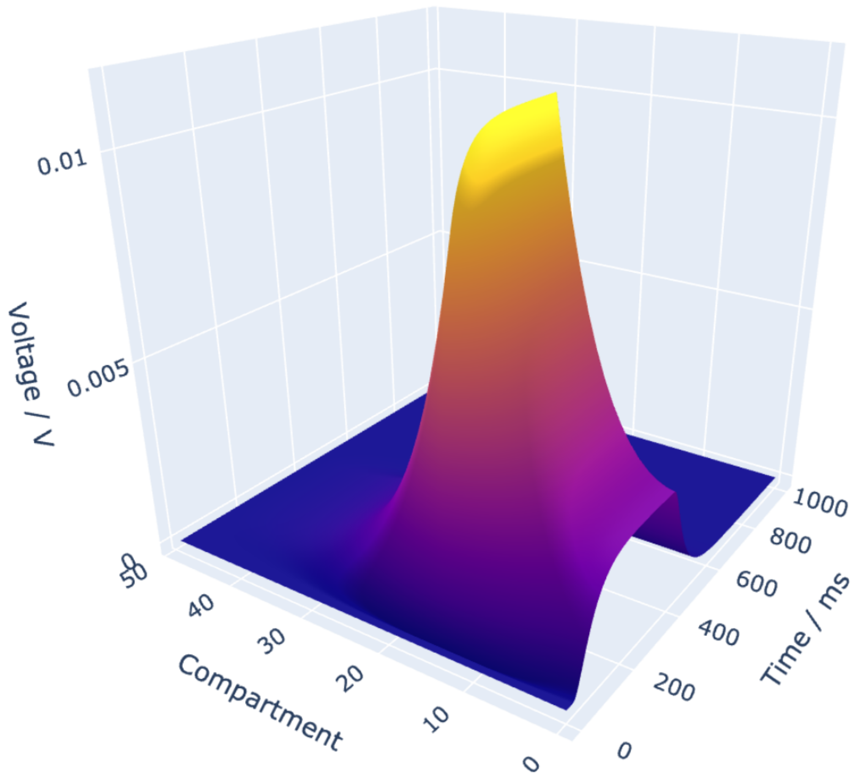

Figure 4: Voltage response over time for 50 compartments.

Question 2. Simulation of a single-compartment Hodgkin-Huxley model of an active neurite.

2.1 The equations (3-12) give the Hodgkin-Huxley model of an imaginary single-compartment (i.e. no axial current) active neurite with input current $I_e(t)$. The electric properties for this neurite are

- $C_m = 1\,\mu\text{F}$, membrane capacitance
- $E_{Na} = 115\,\text{mV}$, sodium equilibrium potential
- $E_K = -12\,\text{mV}$, potassium equilibrium potential
- $E_L = 10.6\,\text{mV}$, leak equilibrium potential

$$C_m \frac{\mathrm{d}V}{\mathrm{d}t} = I_e(t) - \bar{g}_L (V - E_L) - \bar{g}_{\mathrm{Na}} m^3 h (V - E_{\mathrm{Na}}) - \bar{g}_{\mathrm{K}} n^4 (V - E_{\mathrm{K}}) \tag{3}$$

$$\frac{\mathrm{d}m}{\mathrm{d}t} = \alpha_m (1-m) - \beta_m m \tag{4}$$

$$\alpha_m = \begin{cases} 0.1 \frac{V-25}{1-\exp\left(-\frac{V-25}{10}\right)} & V \neq 25mV \\ 1 & V = 25mV \end{cases} \tag{5}$$

$$\beta_m = 4\exp\left(-\frac{V}{18}\right) \tag{6}$$

$$\frac{\mathrm{d}h}{\mathrm{d}t} = \alpha_h (1-h) - \beta_h h \tag{7}$$

$$\alpha_h = 0.07\exp\left(-\frac{V}{20}\right) \tag{8}$$

$$\beta_h = \frac{1}{1+\exp\left(-\frac{V-30}{10}\right)} \tag{9}$$

$$\frac{\mathrm{d}n}{\mathrm{d}t} = \alpha_n (1-n) - \beta_n n \tag{10}$$

$$\alpha_n = \begin{cases} 0.01 \frac{V-10}{1-\exp\left(-\frac{V-10}{10}\right)} & V \neq 10mV \\ 0.1 & V = 10mV \end{cases} \tag{11}$$

$$\beta_n = 0.125\exp\left(-\frac{V}{80}\right) \tag{12}$$

- $V(0) = 0\,\mathrm{mV}$, initial membrane potential
- $\bar{g}_{Na} = 120\,\mathrm{ms}$, maximum sodium channel conductance
- $\bar{g}_K = 36\,\mathrm{ms}$, maximum potassium channel conductance
- $\bar{g}_L = 0.3\,\mathrm{ms}$, maximum leak conductance

To approximate the given DEQs we will use the Forward Euler method

$$V(t+\Delta t) = V(t) + \frac{\Delta t}{C_m}\left(I_e(t) - \bar{g}_L (V(t) - E_L) - \bar{g}_{Na} m^3 h (V(t) - E_{Na}) - \bar{g}_K n^4 (V(t) - E_K)\right)$$
$$m(t+\Delta t) = m(t) + \Delta t\,(\alpha_m (1-m) - \beta_m m)$$
$$h(t+\Delta t) = h(t) + \Delta t\,(\alpha_h (1-h) - \beta_h h)$$
$$n(t+\Delta t) = n(t) + \Delta t\,(\alpha_n (1-n) - \beta_n n)$$

2.2 Now, we implement this model with the input current $I_e(t)$ given by equation 2 with $t_e = 50\,\mathrm{ms}$, $t_s = 300\,\mathrm{ms}$ for different amploitudes $I_0 = 0\,\mu\mathrm{A}$, $I_0 = 3\,\mu\mathrm{A}$, $I_0 = 6\,\mu\mathrm{A}$, $I_0 = 8\,\mu\mathrm{A}$. The resulting voltage responses appear in Figures 5, 6, 7, and 8. The code used to generate the following figures appears at the end of the script.

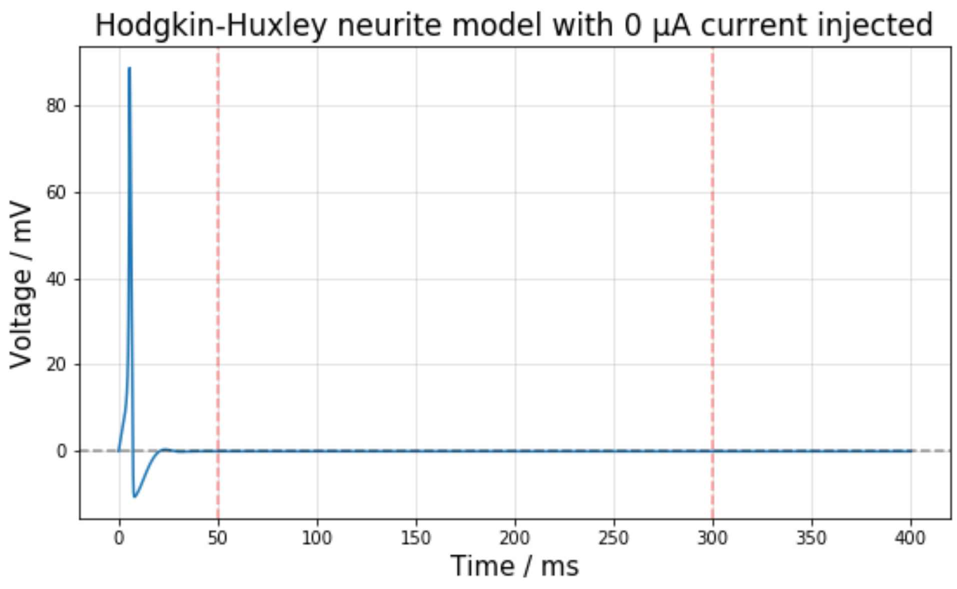

Figure 5: No current injected.

2.3 Next, we simulate the model with different amplitudes of input current (from $I_0 = 0\,\mu$A to $I_0 = 20\,\mu$A) in steps of $0.5\,\mu$A. The plot of firing rate in the stationary state (200 ms after current injection) plotted against applied current is shown in Figure 9.

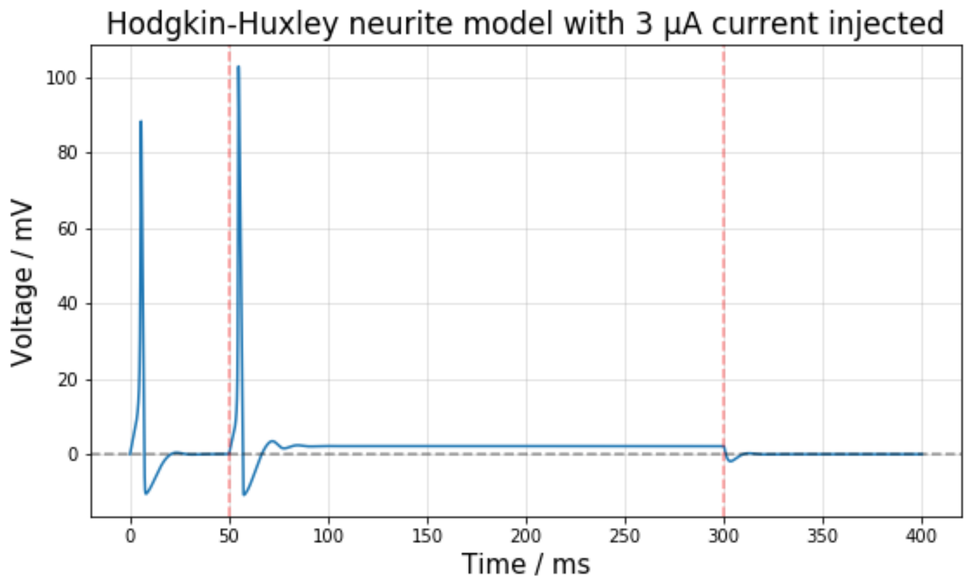

Figure 6: 3 μA injected.

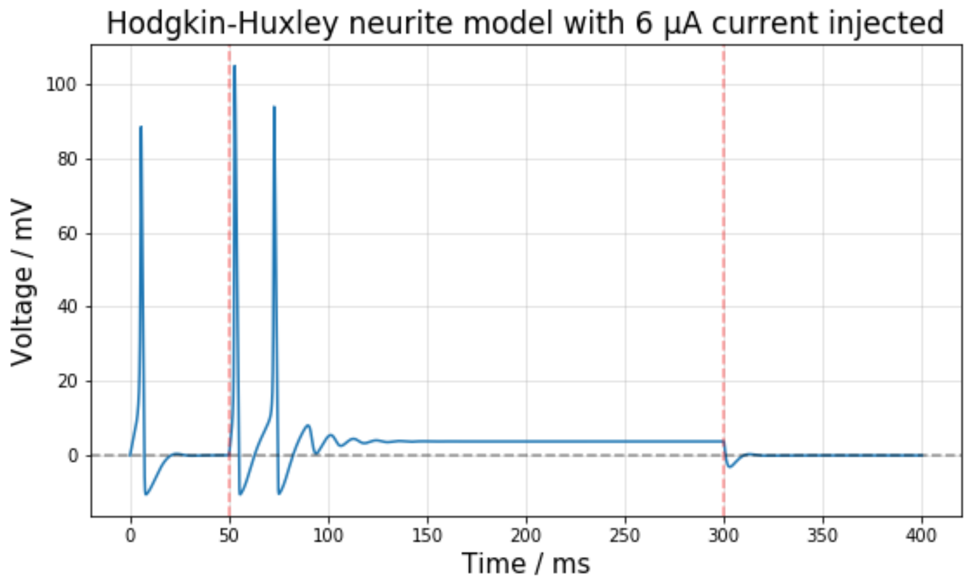

Figure 7: 6 μA injected.

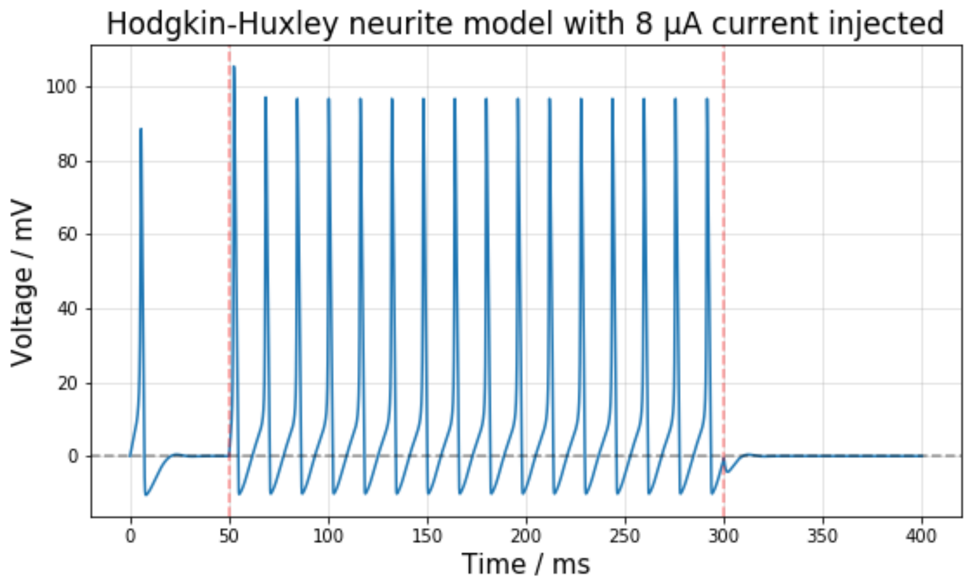

Figure 8: 8 μA injected.

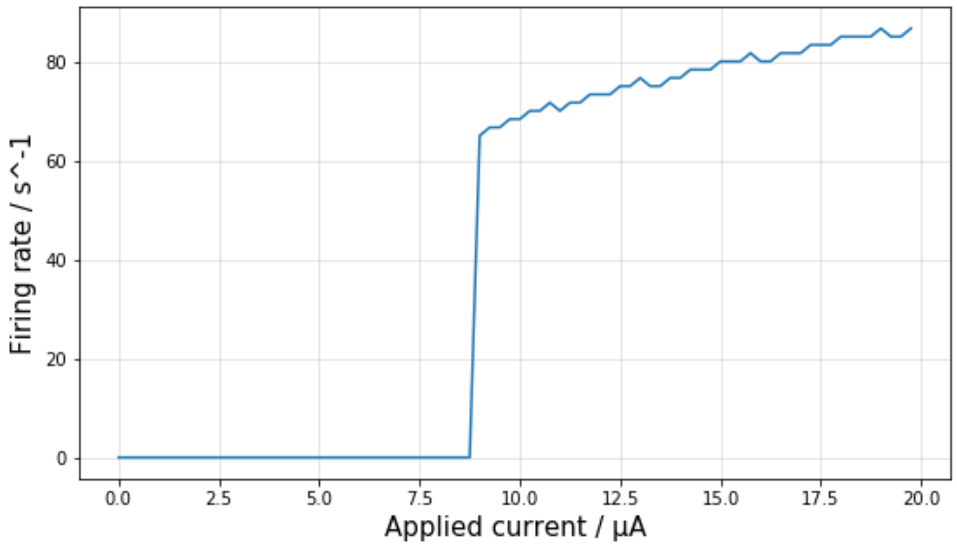

Figure 9: Elicited firing rate in response to applied currenr of varying amplitude.

HH

November 27, 2019

```
[ ]: import numpy as np
     import matplotlib.pyplot as plt
     from scipy.signal import find_peaks
     %matplotlib inline
```

```
[ ]: class Neuron:

         def __init__(self, E_m, C_m, E_na, E_k, E_l, g_na, g_k, g_l):
             self.E_m = E_m
             self.C_m = C_m
             self.E_na = E_na
             self.E_k = E_k
             self.E_l = E_l
             self.g_na = g_na
             self.g_k = g_k
             self.g_l = g_l

         def simulate(self, I_t, dt, time, t_e=None, t_s=None, plot=True,␣
      ↪save=False):

             num_time_bins = int(time/dt)

             V_t  = 0
             V_course = np.zeros(num_time_bins)

             m_t  = 0
             m_t1 = np.zeros(num_time_bins)

             h_t  = 0
             h_t1 = np.zeros(num_time_bins)

             n_t  = 0
             n_t1 = np.zeros(num_time_bins)

             alpha_m = 0
             alpha_h = 0
             alpha_n = 0
```

```
        beta_m = 0
        beta_h = 0
        beta_n = 0

        for i in np.arange(0, int(time/dt), 1):

            if t_e:
                if (i*dt) < t_e:
                    I = I_t[0]
                elif t_e <= (i*dt) and (i*dt) < t_s:
                    I = I_t[1]
                else:
                    I = I_t[2]
            else:
                I = I_t

            beta_m = 4*np.exp(-V_t/18)
            beta_h = 1/(1 + np.exp(-(V_t - 30)/10))
            beta_n = 0.125*np.exp(-V_t/80)

            if V_t == 25:
                alpha_m = 1
            else:
                alpha_m = 0.1*((V_t - 25)/(1 - np.exp(-(V_t - 25)/10)))
            alpha_h = 0.07*np.exp(-V_t/20)
            if V_t == 10:
                alpha_n = 0.1
            else:
                alpha_n = 0.01*((V_t - 10)/(1 - np.exp(-(V_t - 10)/10)))

            m_t1[i] = m_t + dt*(alpha_m*(1 - m_t) - beta_m*m_t)
            m_t = m_t1[i]
            h_t1[i] = h_t + dt*(alpha_h*(1 - h_t) - beta_h*h_t)
            h_t = h_t1[i]
            n_t1[i] = n_t + dt*(alpha_n*(1 - n_t) - beta_n*n_t)
            n_t = n_t1[i]

            leak = self.g_l*(V_t - self.E_l)
            sod  = self.g_na*(m_t**3)*h_t*(V_t - self.E_na)
            pot  = self.g_k*(n_t**4)*(V_t - self.E_k)
            V_course[i] = V_t + (dt/self.C_m)*(I - leak - sod - pot)
            V_t = V_course[i]

        if plot:
            figure = plt.figure(figsize=(9, 5))
            plt.grid(alpha=0.4)
            xi = np.arange(0, len(V_course), 1)*dt
            plt.plot(xi, V_course)
            plt.axhline(y=0, color='k', linestyle='--', alpha=0.4)
#             plt.axvline(x=t_e, color='r', linestyle='--', alpha=0.4)
#             plt.axvline(x=t_s, color='r', linestyle='--', alpha=0.4)
            plt.xlabel('Time / ms', fontsize=15)
            plt.ylabel('Voltage / mV', fontsize=15)
            plt.xticks(fontsize = 10)
            plt.yticks(fontsize = 10)
            plt.title('Hodgkin-Huxley neurite model with %.2f ţA current␣
 ↪injected'%I, fontsize=17)
            if save:
                plt.savefig('Current_%dA.png'%I_t[1])

        return V_course
```

[ ]:
```
E_m = 0
C_m = 1
E_na = 115
E_k = -12
E_l = 10.6
g_na = 120
g_k = 36
g_l = 0.3

a = Neuron(E_m, C_m, E_na, E_k, E_l, g_na, g_k, g_l)
```

[ ]:
```
I = [0, 8, 0]
t_e = 50
t_s = 300
dt = 0.01 #ms
time = t_s+100 #ms

v = a.simulate(I, dt, time, t_e, t_s, save=False)
```

[ ]:
```
def threshold_detect(signal):
    peaks, _ = find_peaks(signal, prominence=1)
    return peaks
```

[ ]:
```
I = np.arange(0, 20, 0.25)
dt = 0.01 #ms
time = 800 #ms

v = np.zeros((len(I), int(time/dt)))

for c in range(len(I)):
    v[c, :] = a.simulate(I[c], dt, time, plot=False, save=False)
```

[ ]:
```
r = []
for i in range(v.shape[0]):
    trace = v[i, 20000:]
    x = threshold_detect(trace)
    r.append(np.float(len(x))/np.float(0.6))
```

[ ]:
```
figure = plt.figure(figsize=(9, 5))
plt.grid(alpha=0.4)
plt.xlabel('Applied current / ţA', fontsize=15)
plt.ylabel('Firing rate / s^-1', fontsize=15)
plt.plot(I, r)
# plt.scatter(I, r, marker='x', alpha=0.6)
# plt.savefig('Firing_rate.png')
```

[ ]: