Results02

November 9, 2021

Matteo Zortea, Alessandro Rizzi

Brain Inspired Computing, Sheet 2

Group G3A8

Exercises solved

Ex 1: a) b) c)
Ex 2: a) b)
Ex 3: a)
Ex 4: a)

Exercise 1

a)

At time t the probability of a channel beeing open is $x(t)$, hence the number of expected open channels at time t is $s(t) = Nx(t)$, where N is the total number of channels.
At time $t + \Delta t$, some channels switch from opened to closed and vice-versa. This happen with the transition rates

$$r_{cl\to op} = \alpha_x(u) \qquad\qquad r_{op\to cl} = \beta_x(u)$$

so that the expected number of transitions from closed to opened and from opened to closed at time t in a time intervals δT are respectively

$$N_{cl\to op} = (N - s(t))\,\alpha(u)\,\Delta t = N(1 - x(t))\,\alpha(u)\,\Delta t$$
$$N_{op\to cl} = s(t)\,\beta(u)\,\Delta t = Nx(t)\,\beta(u)\,\Delta t$$

This means that

$$s(t + \Delta t) = s(t) + N_{cl\to op} - N_{op\to cl} = s(t) + [N(1 - x(t))\,\alpha(u) - Nx(t)\,\beta(u)]\,\Delta t$$

so that

$$\frac{s(t + \Delta t) - s(t)}{\Delta t} = N\,[(1 - x(t))\,\alpha(u) - x(t)\,\beta(u)]$$

Taking the limit $\Delta t \to 0$, remembering that $s(t) = N\,x(t)$ one gets the equation for the evolution of the p.d.f.

$$\frac{dx(t)}{dt} = (1 - x(t))\,\alpha(u) - x(t)\,\beta(u) = \alpha(u) - x(t)\;(\alpha(u) + \beta(u))$$

or in short form

$$\dot{x} = \alpha - x\,(\alpha + \beta)$$

b)

One can simply take apart $\alpha + \beta$ from the last expression

$$\dot{x} = (\alpha + \beta)\left(\frac{\alpha}{\alpha + \beta} - x\right) \equiv \frac{1}{\tau}(x_0 - x)$$

so that the required transformations are $\tau = \frac{1}{\alpha+\beta}$ and $x_0 = \frac{\alpha}{\alpha+\beta}$

c)

$$\tau(u) = \alpha(u) + \beta(u) = \frac{1}{1 + e^{-\frac{u+a}{b}}} + \frac{1}{1 + e^{\frac{u+a}{b}}} = \frac{e^{\frac{u+a}{b}}}{1 + e^{\frac{u+a}{b}}} + \frac{1}{1 + e^{\frac{u+a}{b}}} = 1$$

$$x_0(u) = \frac{\alpha(u)}{\alpha(u) + \beta(u)} = \frac{1}{1 + e^{-\frac{u+a}{b}}}$$

Now let us note that

$$1 + \tanh(x) = 1 + \frac{e^{2x} - 1}{e^{2x} + 1} = \frac{2e^{2x}}{e^{2x} + 1} = \frac{2}{1 + e^{-2x}}$$

Hence we can work out the equation for $x_0(u)$ to obtain the desired expression

$$x_0(u) = \frac{1}{1 + e^{-2\frac{u+a}{2b}}} = \frac{1}{2}\left[1 + \tanh\left(\frac{1}{2b}(u - (-a))\right)\right] \equiv \frac{1}{2}\left[1 + \tanh\left(\beta(u - \Theta)\right)\right]$$

where we set $\Theta = -a, \beta = 1/2b$

Exercise 2

a)

Definition of various parameters and importation of libraries. From the way our parameters appear in the differential equations, we can see that our scaling using mV, ms, muS, and nF is correct.

[57]:
```
import numpy as np
from matplotlib import pyplot as plt

# Parameters
dt = 0.01 # ms
T_run = 100 # ms
n0 = 0.3
m0 = 0.1
h0 = 0.6
u0 = -65 # mV

# Resting potentials
E_Na = 50 # mV
E_K = -77 # mV
```

```
E_l = -54.4 # mV
# Conductances
g_Na = 120 # muS
g_K = 36 # muS
g_l = 0.3 # muS

Cm = 1 # nF

npoints = int(T_run/dt)
```

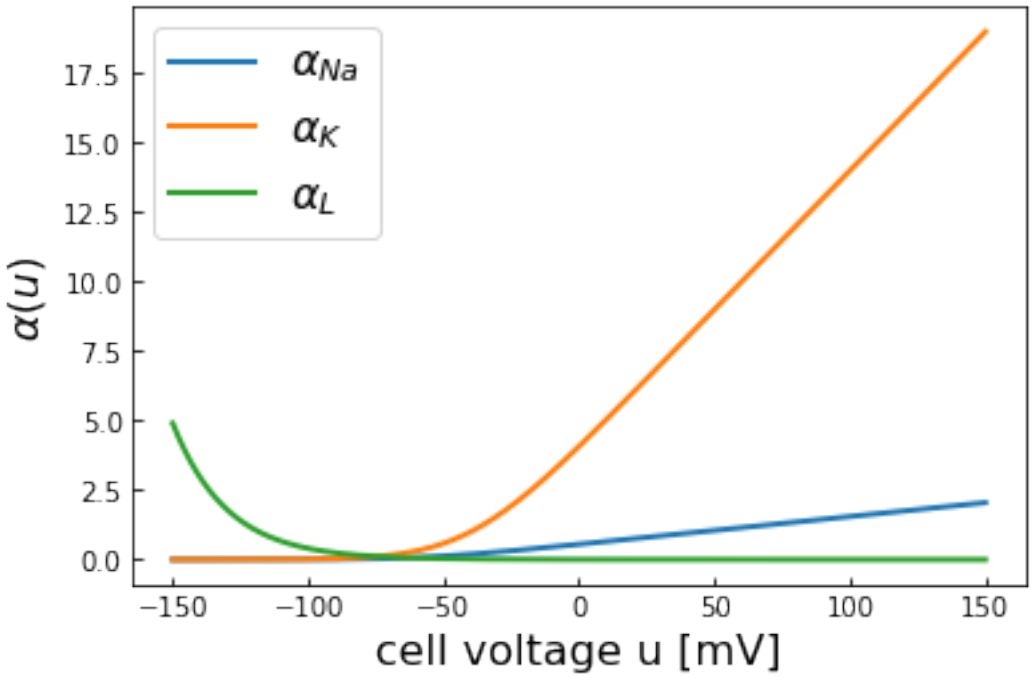
We then define the functions that calculate the transition rates. The plots follow.

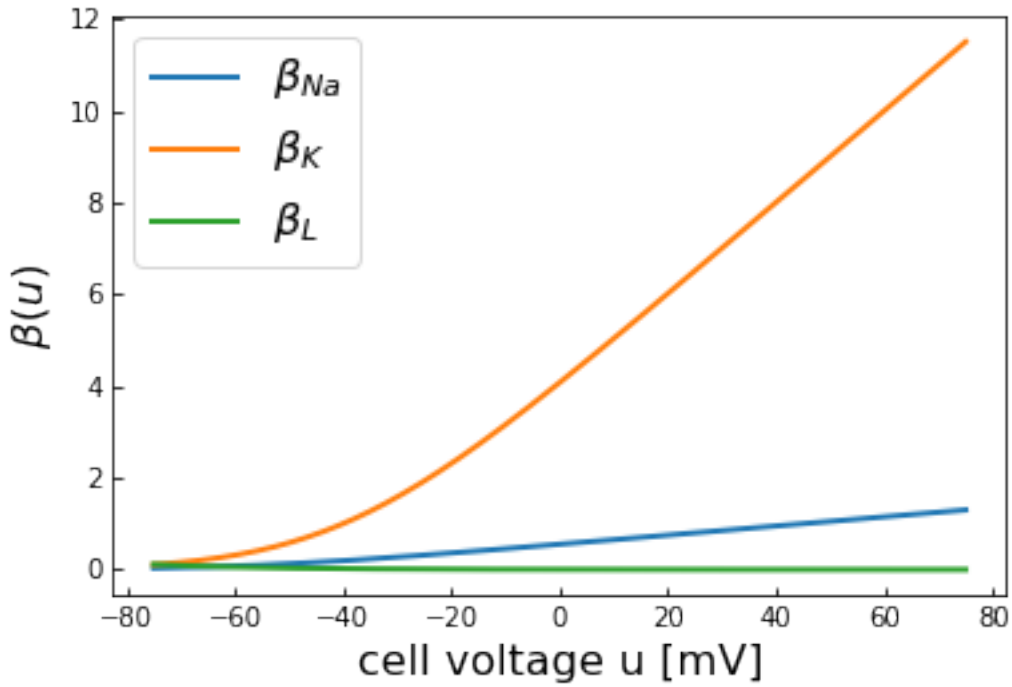
[58]:
```
# Transition rates

# closed to opened
cl_to_op = lambda u: np.array([0.01*(-55-u)/(np.exp(-5.5 - 0.1*u) - 1), 0.
 ↪1*(-40-u)/(np.exp(-4 - 0.1*u) - 1), 0.07*np.exp(-(u+65)/20)])
# opened to closed
op_to_cl = lambda u: np.array([0.125*np.exp(-(u+65)/80), 4.0*np.exp(-(u+65)/18),␣
 ↪1/(np.exp(-3.5-0.1*u) + 1)])

# x0, tau
get_x0 = lambda u: cl_to_op(u)/(cl_to_op(u)+op_to_cl(u))
get_tau = lambda u: 1./(cl_to_op(u) + op_to_cl(u))
```

Let us plot the transition rate functions

[59]:
```
uline = np.linspace(-150, 150, 10000) # x ticks [mV] to plot the transition rates
a = np.zeros((3, len(uline)))
for i in range(len(uline)):
    a[:,i] = cl_to_op(uline[i])
plt.plot(uline, a[0,:], label=r"$\alpha_{Na}$", lw=2)
plt.plot(uline, a[1,:], label=r"$\alpha_{K}$", lw=2)
plt.plot(uline, a[2,:], label=r"$\alpha_{L}$", lw=2)
plt.xlabel("cell voltage u [mV]", fontsize=16)
plt.ylabel(r"$\alpha(u)$", fontsize=16)
plt.legend(fontsize=16)
ax = plt.gca()
ax.tick_params(direction='in')
plt.show()

uline = np.linspace(-75, 75, 10000) # x ticks [mV] to plot the transition rates
b = np.zeros((3, len(uline)))
for i in range(len(uline)):
    b[:,i] = cl_to_op(uline[i])
plt.plot(uline, b[0,:], label=r"$\beta_{Na}$", lw=2)
plt.plot(uline, b[1,:], label=r"$\beta_{K}$", lw=2)
plt.plot(uline, b[2,:], label=r"$\beta_{L}$", lw=2)
```

```
plt.xlabel("cell voltage u [mV]", fontsize=16)
plt.ylabel(r"$\beta(u)$", fontsize=16)
plt.legend(fontsize=16)
ax=plt.gca()
ax.tick_params(direction='in')
plt.show()
```





Input current function

[60]:
```
def I_ext(t):
    if t > 0 and t < 50:
        return 7.5
    else:
        return 0
```

Right hand sides of the differential equations

[61]:
```
def get_xdot(x, u):
    return np.power(tau, -1) * (x0 - x)
def get_udot(u, t, n, m, h):
    return (g_l*(E_l - u) + g_Na*m**3*h*(E_Na - u) + g_K*n**4*(E_K - u) +␣
 ↪I_ext(t))/Cm
```

We have to solve the differential equations

$$\begin{cases} \frac{dn}{dt} = \frac{1}{\tau_n(u)}\left(n_0(u) - n(u)\right) \\ \frac{dm}{dt} = \frac{1}{\tau_m(u)}\left(m_0(u) - m(u)\right) \\ \frac{dh}{dt} = \frac{1}{\tau_h(u)}\left(h_0(u) - h(u)\right) \\ C_m \frac{du}{dt} = g_L\left(E_L - u\right) + g_{Na} m^3 h\left(E_{Na} - u\right) + g_K n^4\left(E_K - u\right) + I_{ext(t)} \end{cases}$$

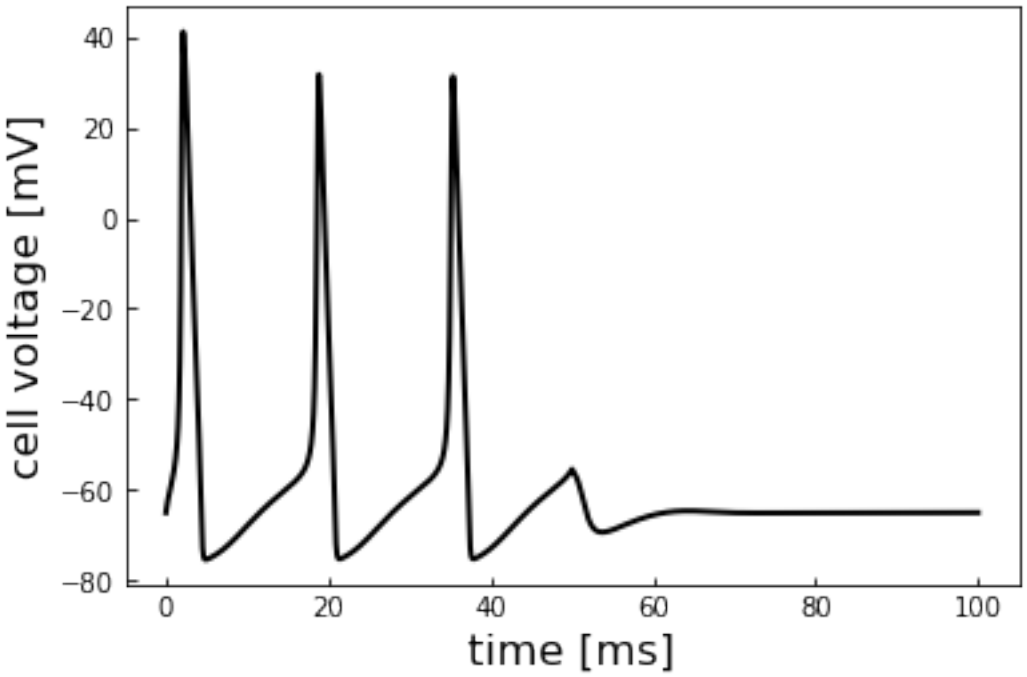
The system of equations can be efficiently integrated using the leapfrog method obtaining a numerical accuracy of order 2, using the same number of evaluations of the Euler method

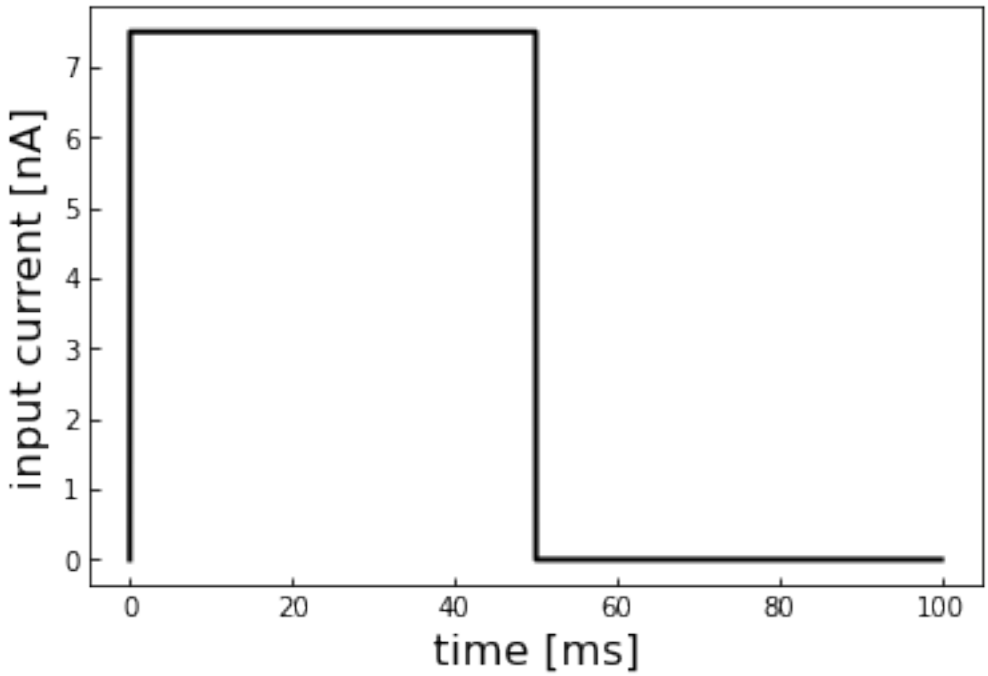
[62]:
```
t = np.zeros(npoints)
x = np.zeros((npoints, 3))
u = np.zeros(npoints)
t[0] = 0
x[0,:] = [n0, m0, h0]
u[0] = u0
```

[63]:
```
# Run the simulation
for i in range(npoints - 1):
    tau = np.array(get_tau(u[i]))
    x0 = np.array(get_x0(u[i]))
    u[i+1] = u[i] + dt*get_udot(u[i], t[i], x[i,0], x[i,1], x[i,2])
    x[i+1] = x[i] + dt*get_xdot(x[i].flatten(), u[i+1])
    t[i+1] = t[i] + dt
```

Plotting the potential and the input current

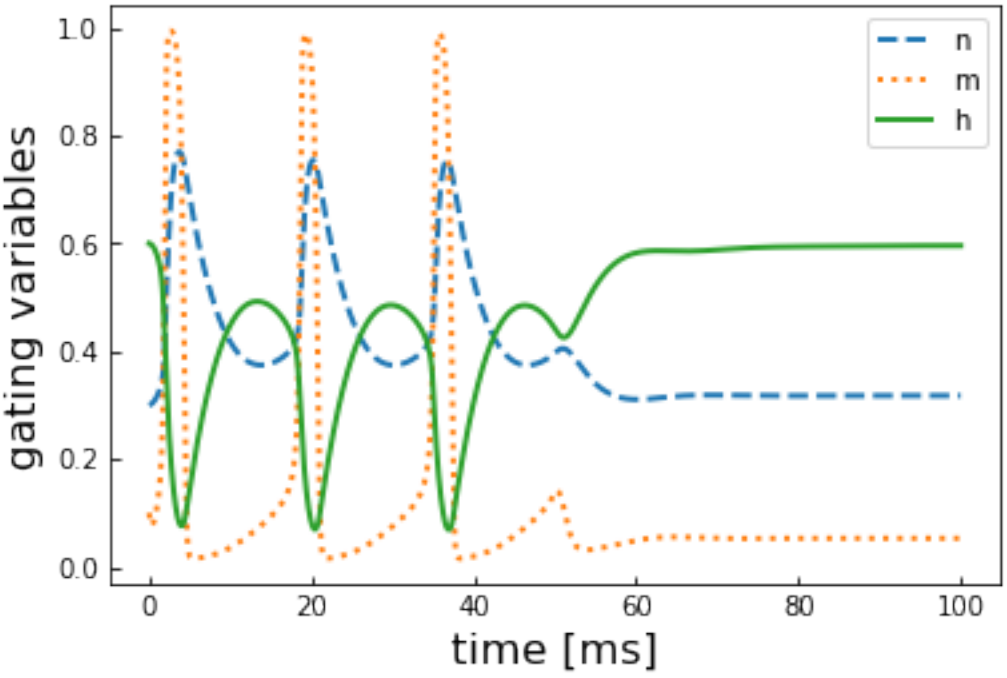
[64]:
```
plt.plot(t, u, lw=2, color='black')
plt.xlabel('time [ms]', fontsize=16)
plt.ylabel('cell voltage [mV]', fontsize=16)
ax = plt.gca()
ax.tick_params(direction='in')
plt.show()

I = []
for tt in t:
    I.append(I_ext(tt))
plt.plot(t, I, lw=2, color='black')
plt.xlabel('time [ms]', fontsize=16)
plt.ylabel('input current [nA]', fontsize=16)
ax = plt.gca()
ax.tick_params(direction='in')
plt.show()
```





Plotting the gating variables

[65]:
```
plt.plot(t, x[:,0], lw=2, label='n', linestyle='dashed')
plt.plot(t, x[:,1], lw=2, label='m', linestyle='dotted')
plt.plot(t, x[:,2], lw=2, label='h')
plt.legend()
plt.xlabel('time [ms]', fontsize=16)
plt.ylabel('gating variables', fontsize=16)
ax = plt.gca()
ax.tick_params(direction='in')
plt.show()
```



b)

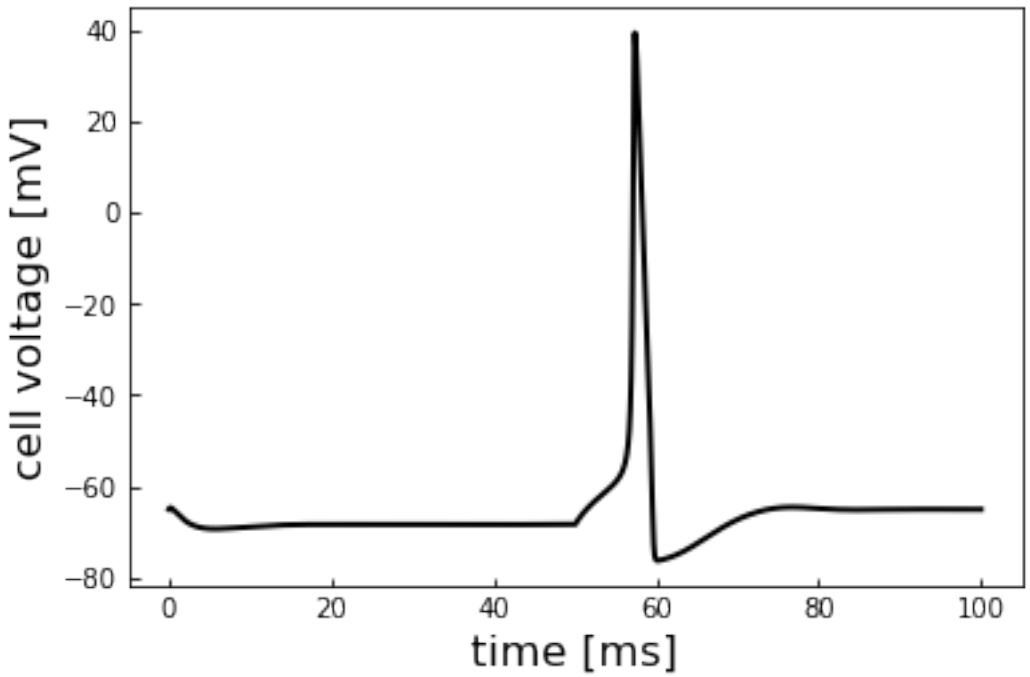
Let us now use the other input current

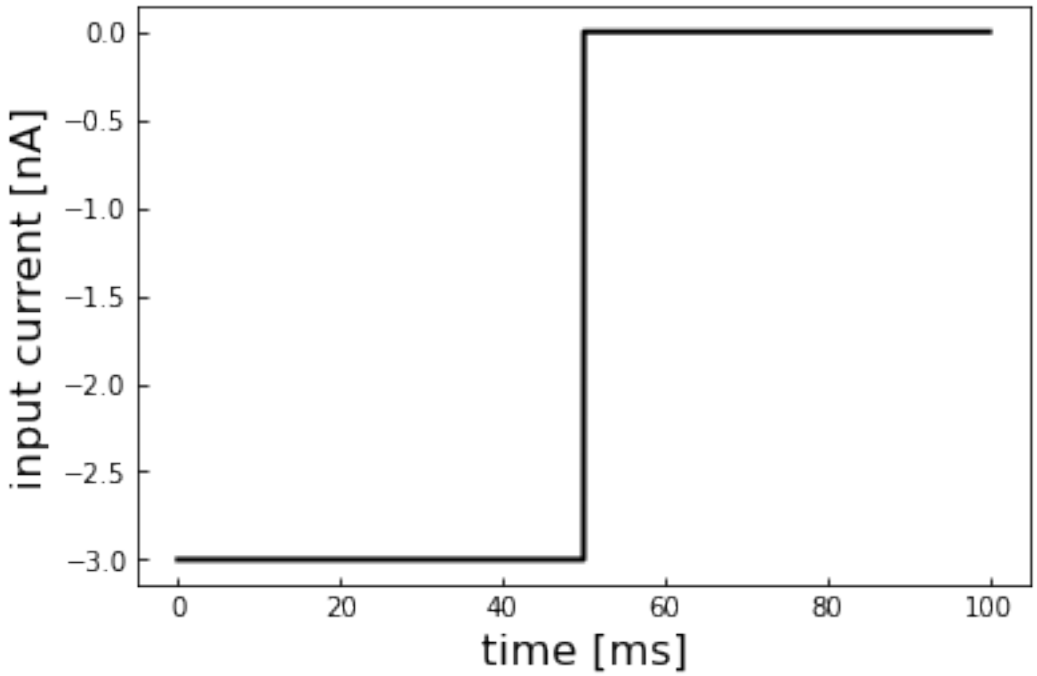
[66]:
```
def I_ext(t):
    if t < 50:
        return -3
    else:
        return 0
```

[67]:
```
# Run the simulation
t = np.zeros(npoints)
x = np.zeros((npoints, 3))
u = np.zeros(npoints)
```

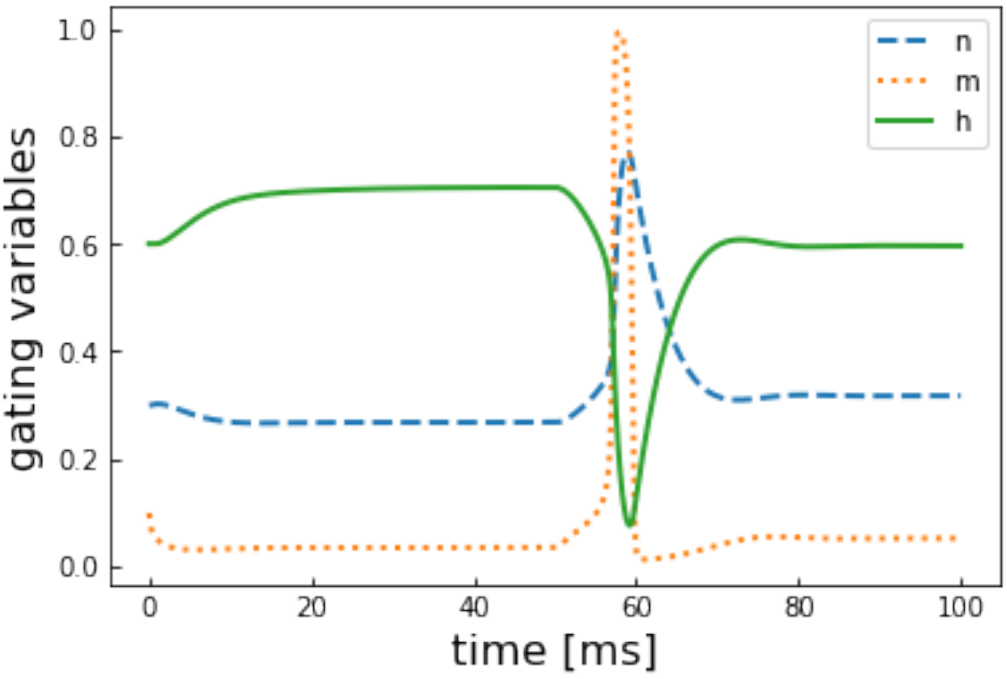
```
t[0] = 0
x[0,:] = [n0, m0, h0]
u[0] = u0
for i in range(npoints - 1):
    tau = np.array(get_tau(u[i]))
    x0 = np.array(get_x0(u[i]))
    u[i+1] = u[i] + dt*get_udot(u[i], t[i], x[i,0], x[i,1], x[i,2])
    x[i+1] = x[i] + dt*get_xdot(x[i].flatten(), u[i+1])
    t[i+1] = t[i] + dt
```

[68]:
```
plt.plot(t, u, lw=2, color='black')
plt.xlabel('time [ms]', fontsize=16)
plt.ylabel('cell voltage [mV]', fontsize=16)
ax = plt.gca()
ax.tick_params(direction='in')
plt.show()

I = []
for tt in t:
    I.append(I_ext(tt))
plt.plot(t, I, lw=2, color='black')
plt.xlabel('time [ms]', fontsize=16)
plt.ylabel('input current [nA]', fontsize=16)
ax = plt.gca()
ax.tick_params(direction='in')
plt.show()
```





[69]:
```
plt.plot(t, x[:,0], lw=2, label='n', linestyle='dashed')
plt.plot(t, x[:,1], lw=2, label='m', linestyle='dotted')
plt.plot(t, x[:,2], lw=2, label='h')
plt.legend()
plt.xlabel('time [ms]', fontsize=16)
plt.ylabel('gating variables', fontsize=16)
ax = plt.gca()
ax.tick_params(direction='in')
plt.show()
```



Exercise 3

With a constant value of the external current, we can solve the differential equation for the membrane potential u:

$$C_m \frac{du}{dt} = g_l(E_l - u) + I_{\text{ext}}.$$

We proceed separing the variables

$$\frac{du}{u - E_l - I_{\text{ext}}/g_l} = -\frac{g_l}{C_m} dt,$$

and then integrating both sides we obtain the solution $u(t)$:

$$u(t) = E_l + \frac{I_{\text{ext}}}{g_l} + Ae^{-\frac{t}{\tau_m}},$$

where A is a constant to be determined by the initial conditions, and $\tau_m = C_m/g_l$ is the membrane time constant.

We want to calculate the firing rate, i.e. the number of spikes per time unit of the neuron. To simplify our task, we choose to set the origin of time $t = 0$ immeadeatly after a spike, i.e. we apply the initial condition $u(0) = E_{\text{reset}}$. With this condition we find that $A = E_{\text{reset}} - E_l - I_{\text{ext}}/g_l$.

We now want to compute the time τ_{spike} needed to elicit a spike, i.e. to reach the treshold value Θ.

$$u(\tau_{\text{spike}}) = \Theta \rightarrow \tau_{\text{spike}} = -\tau_m \ln\left(\frac{\Theta - E_l - I_{\text{ext}}/g_l}{E_{\text{reset}} - E_l - I_{\text{ext}}/g_l}\right)$$

In the LIF neuron, after having reached the treshold, it follows an absolute refractary period τ_{ref}, and then we are again at $u = E_{\text{reset}}$, and the mechanism repeat. Thus, the firing rate is:

$$\nu(I_{\text{ext}}) = \frac{1}{\tau_{\text{ref}} + \tau_{\text{spike}}} = \left[\tau_{\text{ref}} + \tau_m \ln\left(\frac{\Theta - E_l - I_{\text{ext}}/g_l}{E_{\text{reset}} - E_l - I_{\text{ext}}/g_l} \right) \right]^{-1}$$

Exercise 4

Setting $\tau_{\text{ref}} = 0$ and expanding the logarithm at the second order, we obtain:

$$\nu_{\text{asy}}(I_{\text{ext}}) \sim \frac{1}{\tau_m} \left[\frac{g_l}{I_{\text{ext}}}(\Theta - E_{\text{reset}}) - \frac{g_l^2}{I_{\text{ext}}^2} \left(E_{\text{reset}}^2 - \Theta^2 + 2E_l(\Theta - E_{\text{reset}}) \right) + o\left(\frac{1}{I_{\text{ext}}^2} \right) \right]^{-1} =$$

$$= \frac{1}{\tau_m(\Theta - E_{\text{reset}})} \frac{I_{\text{ext}}}{g_l} \left[1 - \frac{g_l}{I_{\text{ext}}} \frac{E_{\text{reset}}^2 - \Theta^2 + 2E_l(\Theta - E_{\text{reset}})}{\Theta - E_{\text{reset}}} + o\left(\frac{1}{I_{\text{ext}}} \right) \right]^{-1}$$

Expanding now the denominator at the first order we find the result:

$$\nu_{\text{asy}}(I_{\text{ext}}) \sim \frac{1}{\tau_m(\Theta - E_{\text{reset}})} \frac{I_{\text{ext}}}{g_l} + \frac{E_{\text{reset}}^2 - \Theta^2 + 2E_l(\Theta - E_{\text{reset}})}{\tau_m(\Theta - E_{\text{reset}})^2}$$