Lab Session #1

Computational Neurophysiology [E010620A]

Dept of Electronics and Informatics (VUB) and Dept of Information Technology (UGent)

Jorne Laton, Matthias Inghels, Talis Vertriest, Jeroen Van Schependom, Sarah Verhulst

Student names and IDs: Constantijn Coppers (02010771)
Academic Year: 2023-2024

General Introduction

In all the practical sessions of this course we will use python 3 and jupyter notebooks. Install Anaconda on your computer and open jupyter notebook by typing "jupyter notebook" or "jupyter lab" in the command line. Your browser will open a file explorer, from where you can navigate to the exercise. It is advised to create a python environment in anaconda before you start working on the notebooks. This way a lot of problems will be solved.

The lab sessions consist of a jupyter notebook in which the different steps are described and explained, together with the tasks you are asked to complete.

You will form groups of two and submit one report per group. Reports should be formatted according to the guiding document. Make sure your answers stand out in the final submitted document!

Deadline: usually 2 weeks after lecture, dates available on Ufora. Reports submitted after the deadline will not be graded.

Context and Goals

This lab session is focused on the Hodgkin-Huxley (HH) model, following and reproducing the theoretical chapter that can be found here: https://neuronaldynamics.epfl.ch/online/Ch2.S2.html. You will be asked to complete code scripts, make observations and explain the results of the different simulations, and contextualise the analyses with the HH model theoretical background.

Questions

Part 1: Hodgkin-Huxley model equations

Q1.1 HH equations

We are coding the HH equations by a single or by multiple functions, to reproduce the behaviour of a human pyramidal neuron when excited. This will be the function to be made to perform the HH model is the following:

`m, h, n, V, INa, IK, alpha_m, alpha_n, alpha_h, beta_m, beta_n, beta_h = HH_model(T, I_input, dt)`

The inputs are:

1. `T` -> Time window of simulation [ms]
2. `I_input` -> input current [μA]
3. `dt` -> the rate of update [ms]

The outputs are:

1. `V` -> the voltage of neuron [mV]
2. `m` -> activation variable for Na-current [u/cm^2]
3. `h` -> inactivation variable for Na-current [u/cm^2]
4. `n` -> activation variable for K-current [u/cm^2]
5. `t` -> the time axis of the simulation (useful for plotting) [ms]
6. `I_Na` -> Na current [μA/cm^2]
7. `I_K` -> K current [μA/cm^2]
8. `alpha_m, beta_m, alpha_n, beta_n, alpha_h, beta_h` -> gating parameters [1/ms]

You can use the functions that are already provided to retrieve some parameters. To complete the code, you can find the equations and parameter values in https://neuronaldynamics.epfl.ch/online/Ch2.S2.html.

Q1.1a Implement the update equations of the gating variables m, n and h as described in Table 2.1 of the online version of the book and reproduce Figure 2.3.

Q1.1b Make the plots and describe in your own words what you have plotted.

Q1.1c Unfortunately, when simulating the HH model with these parameters, you will not get a K-current. Please adjust (u-25) by (u+25) in the update equations for gating variable n.

- Import these modules

- Fill in answers here

Q1.2 Simulate the response to an impulse current

To try out whether the designed functions work, design a step function (A1.2a) that can be used to model the current input (A1.2b). Consider the following design parameters:

1. `I_input function` -> step function
2. `T` -> time of simulation: 100 ms
3. `dt` -> update time: 0.01 ms
4. Current impulse `I_input` : 22 µA between 1 and 2 ms.

- Fill in answers here

Q1.3. Plot V(t)

Plot the first 25 ms of $V(t)$.

Describe the dynamics of the neural voltage V. Does it make sense?

- Fill in answers here

Q1.4 Plot the model parameters

Plot the model parameters n, m and h in function of t and again limit the plots to the first 25 ms. Describe the dynamics of the model parameters m, n, and h. Does it make sense? Describe how the gates swing open and closed during the dynamics of a spike.

- Fill in answers here

Q1.5. Plot I_Na and I_K

Plot I_Na and I_K in function of t (again only the first 25 ms).

Describe the dynamics of the currents. Does it make sense? Describe the currents flows during a spike.

- Fill in answers here

Q1.6. Plot the conductances g_Na and g_K

Plot the conductances g_Na, g_K in function of t (again only the first 25 ms).

How are the conductances evolving during a spike?

- Fill in answers here

Part 2: Package BRIAN

In the second part of the practical, you are going to use the Brian library to simulate the dynamics of a squid neuron when excited. To learn more about this module, read this paper: https://pubmed.ncbi.nlm.nih.gov/19115011/ . Before starting, the Brian module must be installed, together with the neurodynex3 module. Open the anaconda prompt, and install the brian2 package:

```
conda install -c conda-forge brian2
```

and then the neurodynex3 package:

```
pip install neurodynex3
```

Note

If you are working on Linux or one of the newest mac OS systems, you might check whether your pip command is actually pip3.

After installing, the packages are ready to use! If you need more information about the modules, you can find it here https://briansimulator.org/.

- Import these modules

In [1]:
```
#!conda install -c conda-forge brian2
```

Q2.1 Step current response

We study the response of a Hodgkin-Huxley squid neuron to different input currents.

Have a look at the documentation of the functions `HH.simulate_HH_neuron()` and `HH.plot_data()` and the module `neurodynex3.tools.input_factory`.

By using the mentioned functions, code the following steps:

1. Step function for the input current
2. Run the HH simulation (for 300ms)
3. Plot the results of this simulation

Vary the amplitude of the input current between 0.1 and 50 µA between 50 and 250ms. Describe the different dynamics of the spiking neuron.

What is the lowest step current amplitude to generate a spike or to generate repetitive firing? Discuss the difference between the two regimes.

- Fill in answer here

Q2.2 Slow and fast ramp current

The minimal current to elicit a spike does not just depend on the amplitude I or on the total charge Q of the current, but on the "shape" of the current. Let's investigate why.

Q2.2.1 Slow ramp current

Inject a slow ramp current into a HH neuron. The current is 0 µA at t = 0 and starts at 5 ms, linearly increasing to an amplitude of 14.0 µA at t = t_ramp_end. At t > t_ramp_end, the current is set back to 0 µA. A slow ramp duration could be between 30-100 ms.

Experiment with different t_ramp_end values to discover the maximal duration of a ramp, such that the neuron does not spike. Make sure you simulate for at least 20ms after t_ramp_end.

Q2.2.1a Use the function `HH.plot_data()` to visualize the dynamics of the system.

Q2.2.1b What is the membrane voltage at the time when the current injection stops (t=slow_ramp_t_end)?

- Fill in answers here

Q2.2.2 Fast ramp current

Q2.2.2a Do the same as before but this time for a fast ramp current. Start the linearly increasing input current again at t = 5 ms. The amplitude at t = t_ramp_end is 5.0 µA. Start with a duration of 5 ms and then increase the ramp duration it until no spike occurs. Use the function `HH.plot_data()` to visualize the dynamics of the system.

Q2.2.2b What is the membrane voltage at the time when the current injection stops (t=slow_ramp_t_end)?

- Fill in answers here

Q2.2.3. Differences

Discuss the differences between the two situations. Why are the two “threshold” voltages different? Link your observation to the gating variables m, n and h.

Hint: have a look at Chapter 2, Figure 2.3.

- Fill in answers here

Q2.3 Rebound Spike

A HH neuron can spike not only if it receives a sufficiently strong depolarizing input current but also after a hyperpolarizing current. Such a spike is called a rebound spike.

Inject a hyperpolarizing step current I_input = -1 µA for a duration of 25 ms into the HH neuron. Simulate the neuron for 50 ms and plot the voltage trace and the gating variables. Repeat the simulation with I_input = -5 µA. What is happening here? To which gating variable do you attribute this rebound spike?

It may be difficult to to see which gating parameter is responsible for depolarisation (and a possible consequent rebound spike) after a negative current injection. Therefore, also plot the real ratio of n and m, together with the effect of h and use this plot to answer the above question.

- Fill in answers here

Answers

Part 1: Hodgkin-Huxley model equations

Imports

In [2]:
```python
# Import and add all the libraries you need throughout the code
import math as m
import numpy as np
import matplotlib.pyplot as plt

# Make your graphs colorblind-friendly
plt.style.use('tableau-colorblind10')
```

A1.1: HH Equations

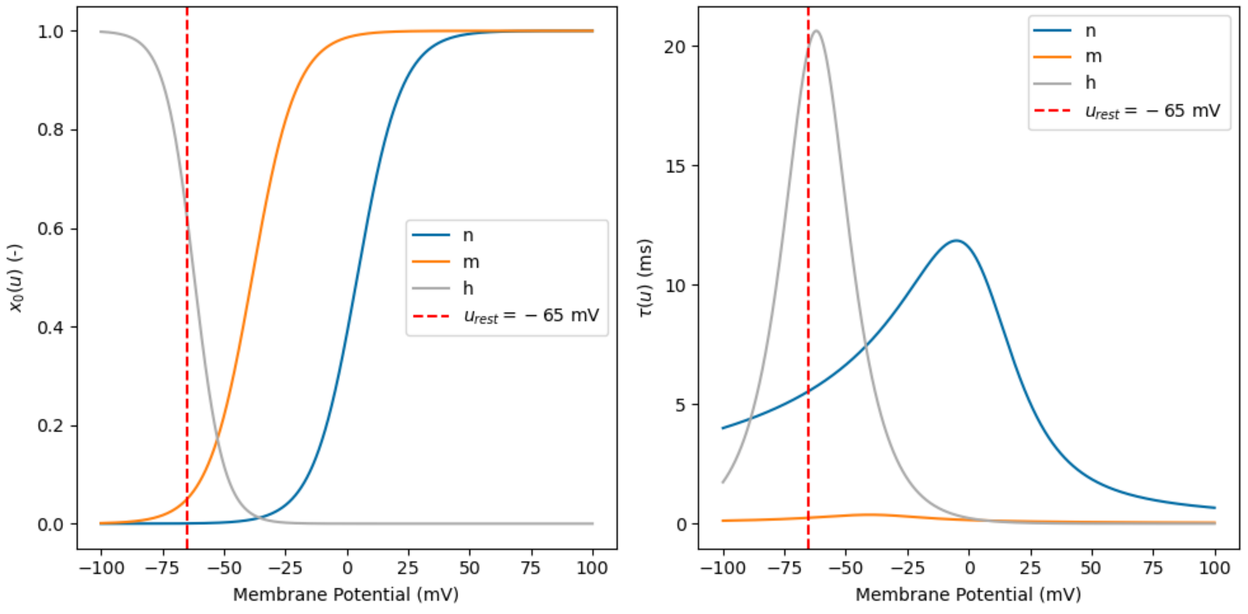
- Go back to Q1.1

In [3]:
```
# A1.1a Enter your code at the bottom of this cell

# Use the following variable names:
#T
#I_input
#dt
#alpha_m
#beta_m
#alpha_n
#beta_n
#alpha_h
#beta_h
#m
#h
#n
#V
#t
#I_Na
#I_K
#I_L

# Hints:
# Reversal potentials have unit mV
# Constant conductances have unit mS/cm2

# Hint: complete the following functions:

def gating_variable_m(u):
    beta_m = -0.124 * (u + 35) / (1 - np.exp((u + 35)/9))
    alpha_m = 0.182 * (u + 35) / (1 - np.exp(-(u + 35)/9))
    return alpha_m, beta_m

def gating_variable_n(u, sign = 1):
    beta_n = -0.002 * (u - sign*25) / (1 - np.exp((u - sign*25)/9))
    alpha_n = 0.02 * (u - sign*25) / (1 - np.exp(-(u - sign*25)/9))
    return alpha_n, beta_n

def gating_variable_h(u):
    beta_h = 0.25 * np.exp((u + 62)/6) / np.exp((u + 90)/12)
    alpha_h = 0.25 * np.exp(-(u + 90)/12)
    return alpha_h, beta_h

def get_alpha_beta(u, variable, sign = 1):
    if variable == 'n':
        return gating_variable_n(u, sign = sign)

    if variable == 'm':
        return  gating_variable_m(u)

    if variable == 'h':
        return gating_variable_h(u)
```

```
def get_x0(u, variable, sign = 1):
    alpha, beta = get_alpha_beta(u, variable, sign = sign)
    return alpha / (alpha + beta)

def get_tau(u, variable, sign = 1):
    alpha, beta = get_alpha_beta(u, variable, sign = sign)
    return 1 / (alpha + beta)

def HH_model(T, I_input, dt, sign = 1):
    ## VARIABLES ##
    g_Na_0 = 40 # mS/cm^2
    E_Na = 55   # mV
    g_K_0 = 35  # mS/cm^2
    E_K = -77   # mV
    g_L_0 = 0.3 # mS/cm^2
    E_L = -65   # mV
    C_m = 1     # uF/cm^2
    #u0 = -65    # initial condition

    ## UPDATE FUNCTIONS ##
    # currents
    I_Na_f = lambda u, m, h: g_Na_0 * m**3 * h * (u - E_Na)
    I_K_f = lambda u, n: g_K_0 * n**4 * (u - E_K)
    I_L_f = lambda u: g_L_0 * (u - E_L)

    # gating variables
    dh = lambda alpha, beta, h: alpha*(1 - h) - beta*h
    dn = lambda alpha, beta, n: alpha*(1 - n) - beta*n
    dm = lambda alpha, beta, m: alpha*(1 - m) - beta*m
    du = lambda u, I_input, m, h, n : 1/C_m * (I_input - I_Na_f(u, m, h)

    ## INITIALIZATION ##
    # Amount of iterations
    N = int(T // dt)

    # time vector
    t = np.arange(N) * dt

    # input function vector
    I_in = I_input(t)

    # Transition rates
    alpha_h, beta_h = np.zeros(N), np.zeros(N)
    alpha_m, beta_m = np.zeros(N), np.zeros(N)
    alpha_n, beta_n = np.zeros(N), np.zeros(N)

    # Membrane potential
    V = np.zeros(N)

    # Gating variables
    m, n, h = np.zeros(N), np.zeros(N), np.zeros(N)
    ## INITIAL CONDITIONS ##
    V[0] = E_L
    m[0] = get_x0(V[0], 'm')
    n[0] = get_x0(V[0], 'n', sign = -1)
    h[0] = get_x0(V[0], 'h')
    #print(f"Initial gating parameters:\nm_0: {m[0]:.4f}\nh_0: {h[0]:.4f}

    ## NUMERICAL IMPLEMENTATION ##
    for k in range(N-1):
        # update the gating parameters
        alpha_h[k], beta_h[k] = gating_variable_h(V[k])
        alpha_m[k], beta_m[k] = gating_variable_m(V[k])
        alpha_n[k], beta_n[k] = gating_variable_n(V[k], sign = -1)

        m[k+1] = m[k] + dt * dm(alpha_m[k], beta_m[k], m[k])
        n[k+1] = n[k] + dt * dn(alpha_n[k], beta_n[k], n[k])
        h[k+1] = h[k] + dt * dh(alpha_h[k], beta_h[k], h[k])

        # update the membrane potential
        V[k+1] = V[k] + dt*du(V[k], I_in[k], m[k], h[k], n[k])

    I_Na = I_Na_f(V, m, h)
    I_K = I_K_f(V, n)
    I_L = I_L_f(V)
    return m, h, n, V, t, I_Na, I_K, I_L, alpha_m, alpha_n, alpha_h, beta
```

In [4]:
```
# A1.1b Plot your graph below

# Hint: the first graph plots rate constants (n, m and h in function of m
# Hint: the second graph plots the voltage dependent time constants (tau_

# What is your conclusion?

# Step Voltage
dV = 0.01
V = np.arange(-100, 100, dV)

# Gating Parameters
g_params = ['n', 'm', 'h']

# x0
x0 = {var : get_x0(V, var) for var in g_params}

# tau
tau = {var : get_tau(V, var) for var in g_params}

# plotting
fig, ax = plt.subplots(1, 2, figsize = (10, 5))

for var in g_params:
    ax[0].plot(V, x0[var], label = var)
    ax[1].plot(V, tau[var], label = var)
```

```
ax[0].set_ylabel('$x_0(u)$ (-)')
ax[1].set_ylabel(r'$\tau(u)$ (ms)')


for axi in ax:
    axi.axvline(x = - 65, ls = '--', color = 'red', label = '$u_{rest}=-6
    axi.legend()
    axi.set_xlabel('Membrane Potential (mV)')


plt.tight_layout()
plt.show()
```



A1.1b conclusion

In order to get a feel for what the time constant τ_x and equilibrium function x_0 ($\forall x \in \{n, m, h\}$) mean, we rely on the theory of Systems and Signals. Differential equations are interpretet as *systems* and the signals x as *states* (state equations).

The gating variables m, n, h describe the state (the probability of being open/closed) of the **voltage gated** ion channels on the neuronal membrane. The combined action of m (activation) and h (inactivating) controls the Na$^+$ channels while the K$^+$ gates are controlled by n (activation). The dynamics of each parameter is described by the first-order differential equation

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -\frac{1}{\tau_x(u)}(x - x_0(u)) \,, \qquad x \in \{n, m, h\}. \tag{1}$$

For $u(t) = u_0\mathbb{H}(t - t_0), u_0 \in \mathbb{R}_0$, the general solution for $t > t_0$ to (1) is

$$\begin{aligned} x(t) &= e^{-\frac{t}{\tau x(u_0)}} x(t_0) + \frac{1}{\tau_x(u_0)} \int_{t_0}^{t} e^{-\frac{1}{\tau x(u_0)}(t-\sigma)} x_0 \mathrm{d}\sigma && (1) \\ &= \underbrace{x_0(u_0)}_{\text{equilibrium state}} + \underbrace{(x_0(u(t_0)) - x_0(u_0))}_{\text{distance from equilibrium}} e^{-\frac{t-t_0}{\tau x(u_0)}} && (2) \end{aligned}$$

Interpretation of the equilibrium function
From equation (2), it's evident that as t tends towards infinity:

$$x(t) \to x_0(u_0).$$

This indicates that each gating variable will approach its equilibrium state $x_0(u_0)$. At this equilibrium, the gating variable stabilizes: when $x(t) = x_0(u_0)$, equation (1) resolves to zero. For every membrane potential, there exists a distinct equilibrium state toward which each gating parameter progresses. The left plot illustrates the equilibrium value for each gating variable as a function of the (time-independent^2) membrane potential.

Interpreatiton of the time constant
The time constant τ_x elucidates how (and whether) the system transitions towards the equilibrium state. Assuming $\tau_x > 0$, the system remains stable (eigenvalues, i.e. roots of equation (1)'s characteristic equation lie to the left of the imaginary axis). As depicted in equation (2), when $\tau_x^{(1)} > \tau_x^{(2)}$, system 1's evolution towards equilibrium is slower compared to system 2. For every membrane potential, each gating variable reacts uniquely to alterations in the parameter state. In other words, the speed at which equilibrium is attained varies depending on the membrane potential. Alternatively, τ_x can be interpreted as determining the ease with which the equilibrium state x_0 is attained. The right plot illustrates the time constant in relation to the (time-independent) membrane potential.

Conclusion for the plot above

1. At the resting potential, the voltage gated sodium channels are almost closed (m_0 close to zero, while $h_0 \approx 0.6$), only very little sodium can enter the neuron. Increasing the voltage from the resting potential means that $h_0 \to 0$ and $m_0 \to 1$.
2. At the resting potential, the voltage gated potasium channels are closed ($n \approx 0$). Increasing the membrane potential from the resting potential opens the sodium channels $n_0 \to 1$.
3. τ_m is the smallest for all membrane voltages. This means that when the membrane voltage changes, the sodium channels will become activating. In

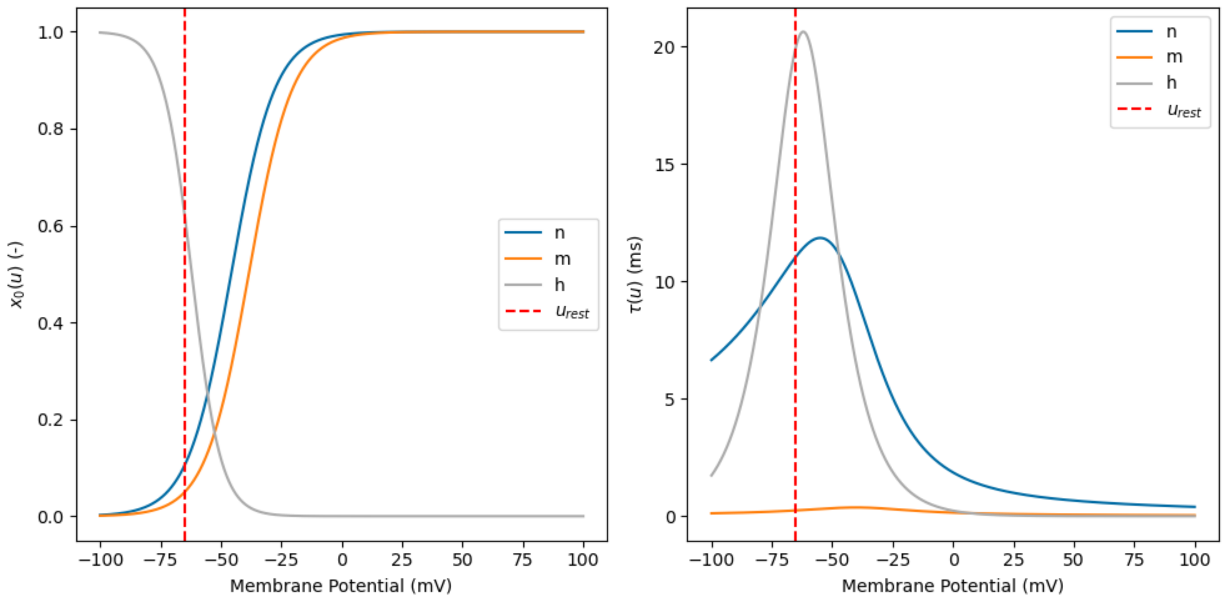
contrast n and m have a higher time constant and thus will respond slower to changes in membrane voltage.

[1]Physically, this implies that the gating parameters remain static over time (in the absence of external stimuli).
[2]These plots are applicable only for a step membrane potential: $u(t) = u_0\mathbb{H}(t - t_0)$.
[3]Physically, this assumption is reasonable, as negative time is not a valid concept.

In [5]:
```
# A1.1c Enter the updated HH model here

# Generally: set sign = -1 to get the adjusted gating parameter n

## the adjusted gating parameters is obtained by setting: sign = -1
adjusted_gating_variable_n = lambda u: gating_variable_n(u, sign = -1)

## the adjusted HH model is also obtained by setting: sign = -1
adjusted_HH_model = lambda T, I_input, dt: HH_model(T, I_input, dt, sigr
```

In [6]:
```
## make a plot of the adjusted gating variable
dV = 0.01
V = np.arange(-100, 100, dV)

# Gating Parameters
g_params = ['n', 'm', 'h']

# x0
x0 = {var : get_x0(V, var, sign = -1) for var in g_params}

# tau
tau = {var : get_tau(V, var, sign = -1) for var in g_params}

# plotting
fig, ax = plt.subplots(1, 2, figsize = (10, 5))

for var in g_params:
    ax[0].plot(V, x0[var], label = var)
    ax[1].plot(V, tau[var], label = var)


ax[0].set_ylabel('$x_0(u)$ (-)')
ax[1].set_ylabel(r'$\tau(u)$ (ms)')


for axi in ax:
    axi.axvline(x = - 65, ls = '--', color = 'red', label = '$u_{rest}$'
    axi.legend()
    axi.set_xlabel('Membrane Potential (mV)')
```

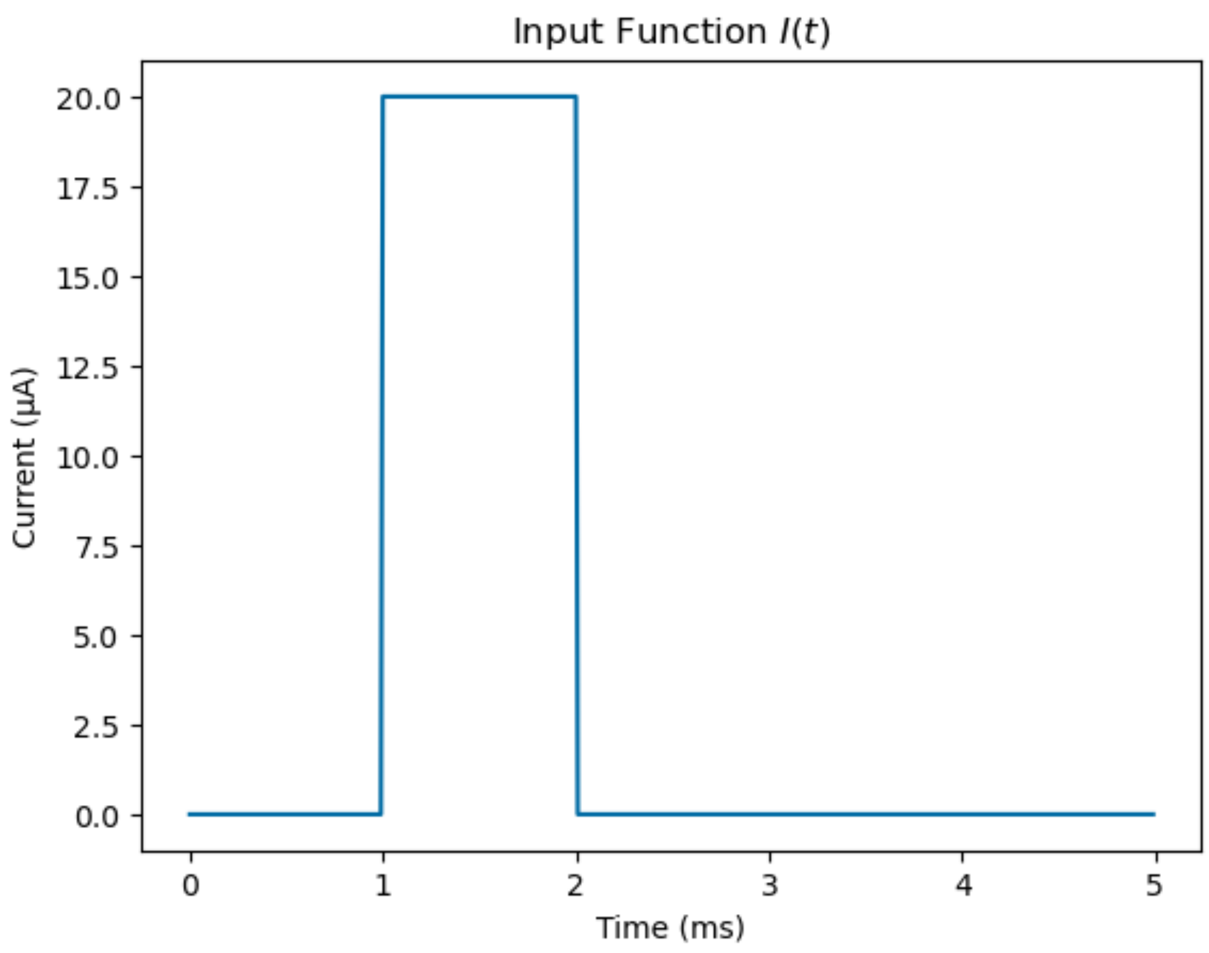
```
plt.tight_layout()
plt.show()
```



A1.2: Response simulation

- Go back to Q1.2

In [7]:
```
# A1.2a Set up and plot your input current

def block_function(t, t0 = 1, t1 = 2):
    return (t0 <= t) * (t <= t1) * 20
T = 100
dt = 0.01
t = np.arange(500) * dt

fig, ax = plt.subplots(1, 1)
ax.plot(t, block_function(t))

plt.title('Input Function $I(t)$')
ax.set_xlabel('Time (ms)')
ax.set_ylabel('Current (μA)')
plt.show()
```

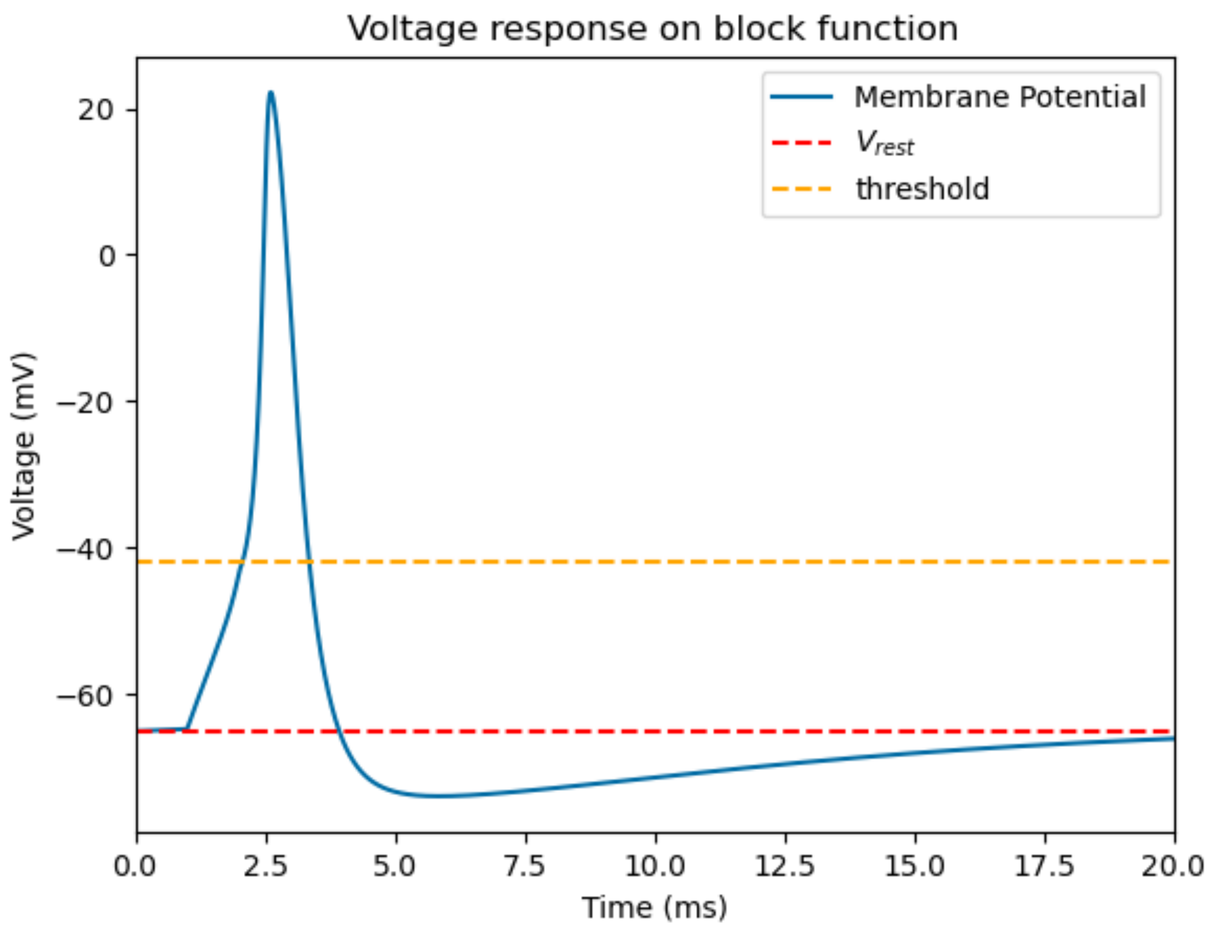


In [8]:
```
# A1.2b Insert the input current into your HH_model function

m, h, n, V, t, I_Na, I_K, I_L, alpha_m, alpha_n, alpha_h, beta_m, beta_r
```

A1.3: Plot V(t)

- Go back to Q1.3

In [9]:
```
# A1.3a Enter your answer below
fig, ax = plt.subplots(1, 1)
ax.plot(t, V, label = 'Membrane Potential')

plt.title('Voltage response on block function')
ax.set_xlabel('Time (ms)')
ax.set_ylabel('Voltage (mV)')
ax.axhline(y = -65, linestyle='--', color = 'red', label='$V_{rest}$')
ax.axhline(y = -42, linestyle='--', color = 'orange', label='threshold')
ax.set_xlim(0, 20)

plt.legend()
plt.show()
```



A1.3b conclusion

- When the neuron is at rest, before application of the input current, it is polarized at the rest potential^1 $V = -65$ mV.
- At $t = 1$ ms, a current is injected into the neuron and therefore the membrane potential moderately^2 increases (depolarization).
- When the membrane potential reaches about -42 mV (threshold) an action potential will be generated (rapid depolarization).
- After reaching a peak voltage (about 20 mV), the membrane potential drops to potentials below the resting potential (hyperpolarization) and progresses back towards the resting potential.

^{1}**Resting membrane potential:**
A neuron at rest (no current injection) is polarized as a consequence of

1. **differences in K$^+$ and Na$^+$ concentrations inside and outside the cells:** The Na$^+$ concentration is higher *outside* the cell while the K$^+$ concenration is higher *inside* the cell. The Na$^+$-K$^+$ pumps actively maintain the concentration gradient across the neuronal membrane. In the Hudgkin-Huxley model the Na$^+$-K$^+$ pumps represent the batteries E_{K^+} and E_{Na^+}.
2. **differences in permeability of the plasma membrane to K$^+$ and Na$^+$:**

The plasma membrane is more permeable (NOT the ***voltage gated*** channels, they are closed) to K$^+$, and thus these ions flow down its concentration gradient outside the neuron. A negative membrane potential develops as a result. However, the plasma membrane is less permeable to Na$^+$, which will slightly reduce negative membrane potential.

^{2} We know that the laplace transform of a Heaviside function is given by

$$\mathbb{H}(\cdot) \overset{\mathcal{L}}{\longleftrightarrow} \frac{1}{s}, \qquad \text{ROC} = \{s \in \mathbb{C} : \Re s > 0\}.$$

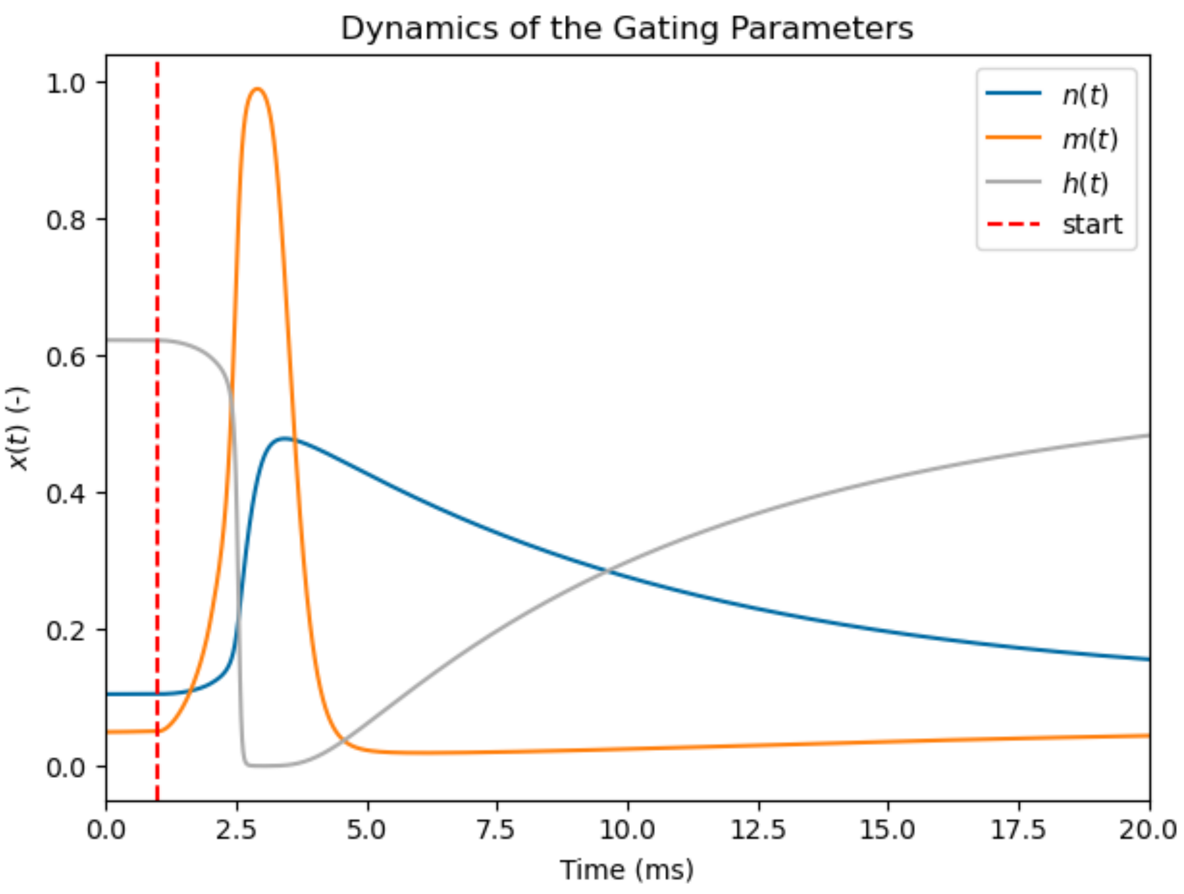
We can write any block function as a linear combination of (delayed) heaviside functions. This means that the linear part of the membrane potential (before the threshold) is due to the integrator-like behavior of our input funtion!

A1.4: Plot the model parameters

- Go back to Q1.4

In [10…

```
# A1.4a Plot your graph below

fig, ax = plt.subplots(1, 1)

ax.plot(t, n, label = '$n(t)$')
ax.plot(t, m, label = '$m(t)$')
ax.plot(t, h, label = '$h(t)$')
ax.axvline(x = 1, ls = '--', label = 'start', color = 'red')

plt.title('Dynamics of the Gating Parameters')
ax.set_xlabel('Time (ms)')
ax.set_ylabel('$x(t)$ (-)')
ax.set_xlim(0, 20)


plt.legend()
plt.tight_layout()
plt.show()
```



A1.4b conclusion

Voltage gated ion channels
An action potential is generated by the control of the in- and outflux of the sodium and potassium ions. This is regulated by the voltage gated ion channels:

1. **Voltage gated Na$^+$ channels**
   These channels have three different states:
   - *closed* at the resting state, so no Na$^+$ enters the cell through them.
   - *opened* by depolarization, therby allowing Na$^+$ influx.
   - *inactivated* Soon after the channel opens, it is automatically blocked by inactivation gates.
2. **Voltage gated K$^+$ channels**
   These channels have two different states:
   - *closed* at the resting state, so no K$^+$ enters the cell through them.
   - *opened* by depolarization, after a delay, therby allowing K$^+$ outflux.

The *closed*-*opened* state is modeled by the parameter m, and the *inactivation* is modeled by the parameter h. The state of the K$^+$ channels is moddeled by the parameter n.

Dynamics of the voltage gated ion channels during an AP

1. **Resting State**
   At the resting potential, all voltage gated Na$^+$ and K$^+$ are closed: m, n are close to zero (see A1.1b).
2. **Depolarization**
   When the membrane potential increases, the Na$^+$-channels open, allowing rapid Na$^+$ entry. As a consequence the neuron starts to depolarize rapidly. Recall from A1.1b that at the resting potential, $h \approx 6$ while m is close to zero: the channel is closed. When the current is injected into the neuron, m will react very fast (low τ_m) and will increase towards 1 (open the channel). However, h is reacting less fast to this change in potential and thus it will take more time for h to progress from 0.6 towards 0 (blocking the channel).
3. **Repolarization**
   Shortly after the Na$^+$ channels open the Na$^+$ channels become inactivating[1] ($h \rightarrow 0$ around 2.5 ms) so no Na$^+$ is able to enter anymore. Eventually, the K$^+$ channels open (n is increasing, τ_n is high causing the delayed response), allowing K$^+$ to exit. The neuron starts to repolarize.
4. **Hyperpolarization**
   Some K$^+$ channels remain open (n is slowly decreasing) and the Na$^+$ channels slowly de-inactivate (n, h both have a large τ_x around the resting membrane potential), causing the hyperpolarized neuron. As time evolves, all gating paremeters are reset i.e. n and h return to their equilibrium (refractoriness).

[1]From the plot of question 1.1c, you can see that τ_h is large around the resting potential and thus h reacts more slowely than m (reacts very fast), which is benficial. If h would react faster than n, no Na$^+$ current would flow into the neuron.

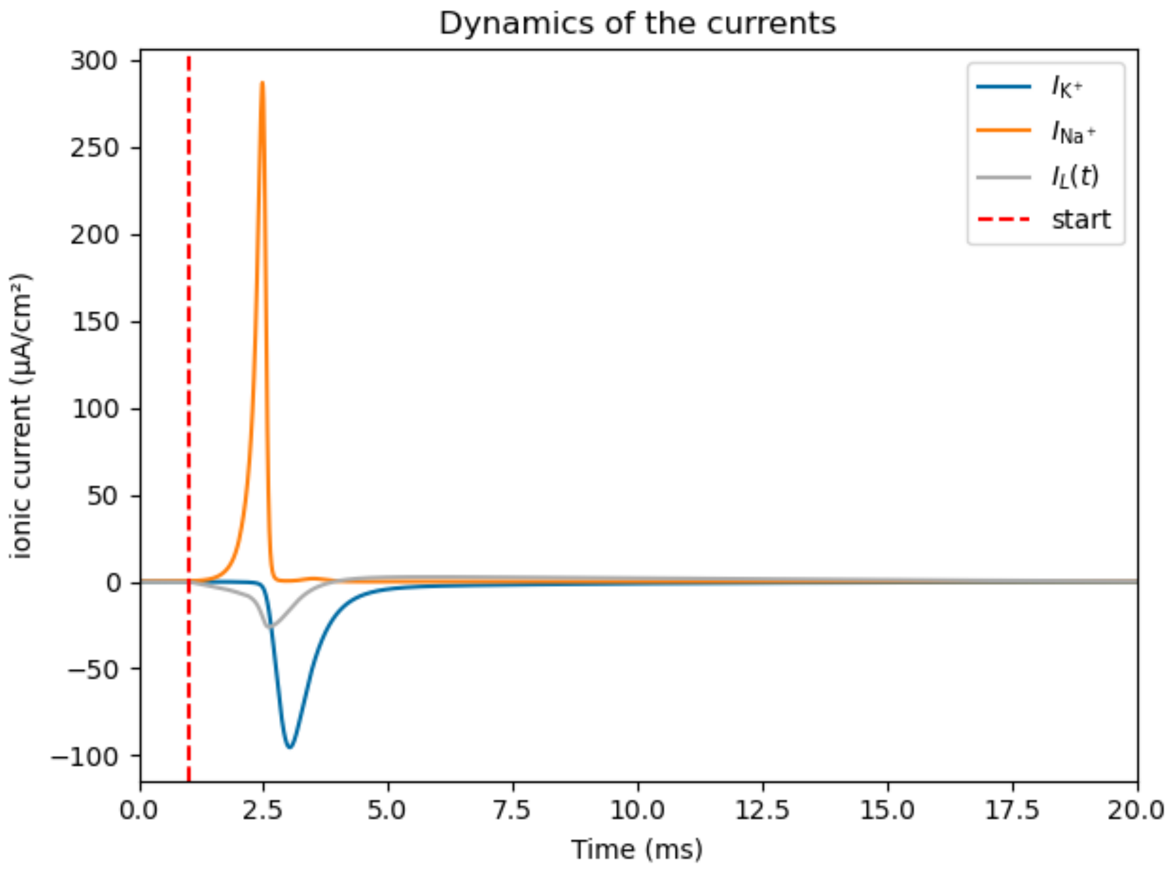
A1.5: Plot I_Na and I_K

- Go back to Q1.5

In [11…

```
# A1.5a Plot your graph here

fig, ax = plt.subplots(1, 1)

ax.plot(t, -I_K, label = '$I_{\mathrm{K}^+}$')
ax.plot(t, -I_Na, label = '$I_{\mathrm{Na}^+}$')
ax.plot(t, -I_L, label = '$I_L(t)$')
ax.axvline(x = 1, ls = '--', label = 'start', color = 'red')
```

```
plt.title('Dynamics of the currents')
ax.set_xlabel('Time (ms)')
ax.set_ylabel('ionic current (µA/cm²)')
ax.set_xlim(0, 20)


plt.legend()
plt.tight_layout()
plt.show()
```



A1.5b conclusion
In the plot above: a positive current is associated with an inflow, while negative currents indicate inflow into the neuron.

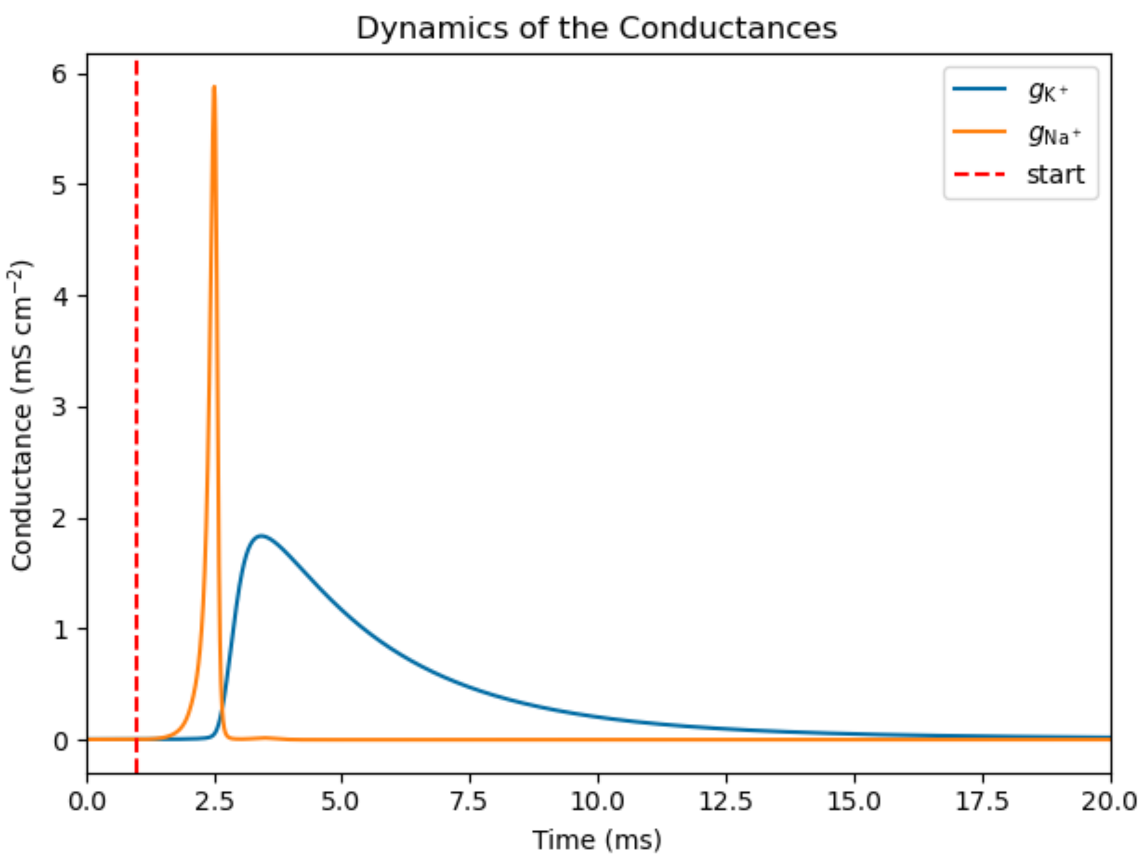
As previously mentioned, the sodium current is regulated by the gating variables m and h. As depicted in the preceding plot, the interplay between these parameters allows Na$^+$ to swiftly enter the neuron (current peak), indicating an open channel, until reaching $t = 2.5$ ms, when sodium influx abruptly ceases due to channel inactivation. Subsequently, the potassium (K$^+$) channels open (delayed) for an extended duration, facilitating the outward flow of K$^+$ ions from the cell. Eventually, all currents converge towards zero: all channels are closed.

A1.6: Plot the conductances g_Na and g_K

- Go back to Q1.6

In [12…

```
# A1.6a Enter your code and plot

g_Na = 40 * m**3 * h
g_K = 35 * n**4

fig, ax = plt.subplots(1, 1)

ax.plot(t, g_K, label = '$g_{\mathrm{K}^+}$')
ax.plot(t, g_Na, label = '$g_{\mathrm{Na}^+}$')
ax.axvline(x = 1, ls = '--', label = 'start', color = 'red')

plt.title('Dynamics of the Conductances')
ax.set_xlabel('Time (ms)')
ax.set_ylabel('Conductance (mS cm$^{-2}$)')
ax.set_xlim(0, 20)

plt.legend()
plt.tight_layout()
plt.show()
```



A1.6b conclusion

At the onset of the action potential, the neuron undergoes a rapid increase in Na$^+$ conductivity over a brief interval, while the conductivity of K$^+$ remains negligible. Around 2.5 ms, this pattern shifts: Na$^+$ conductivity diminishes to zero, while K$^+$ conductivity gradually rises and persists for an extended duration (compared to the short Na$^+$ conductivity duration), but eventually becomes zero again.

Part 2: Package BRIAN

Import

In [13…

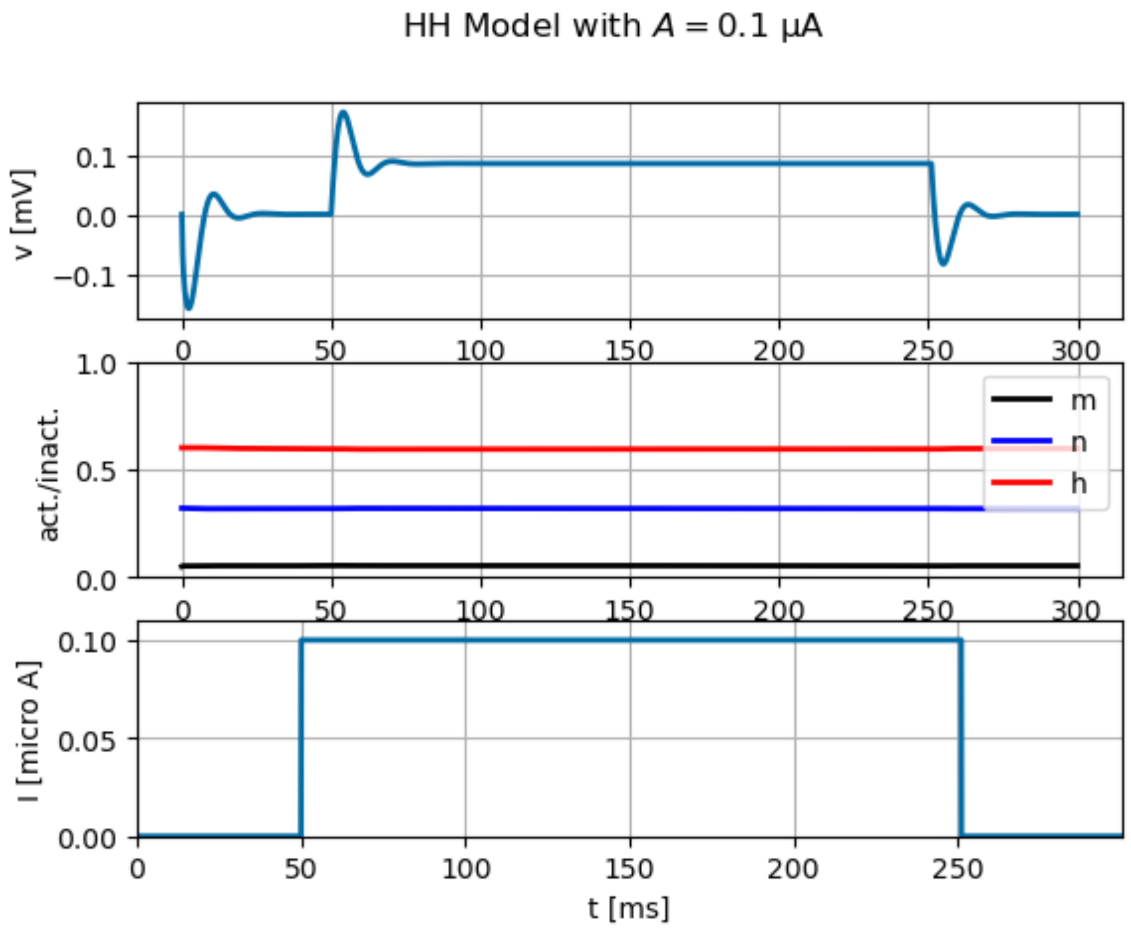
```
# The new libraries we need to add
%matplotlib inline
import brian2 as b2
import matplotlib.pyplot as plt
import numpy as np
from neurodynex3.hodgkin_huxley import HH
from neurodynex3.tools import input_factory
```

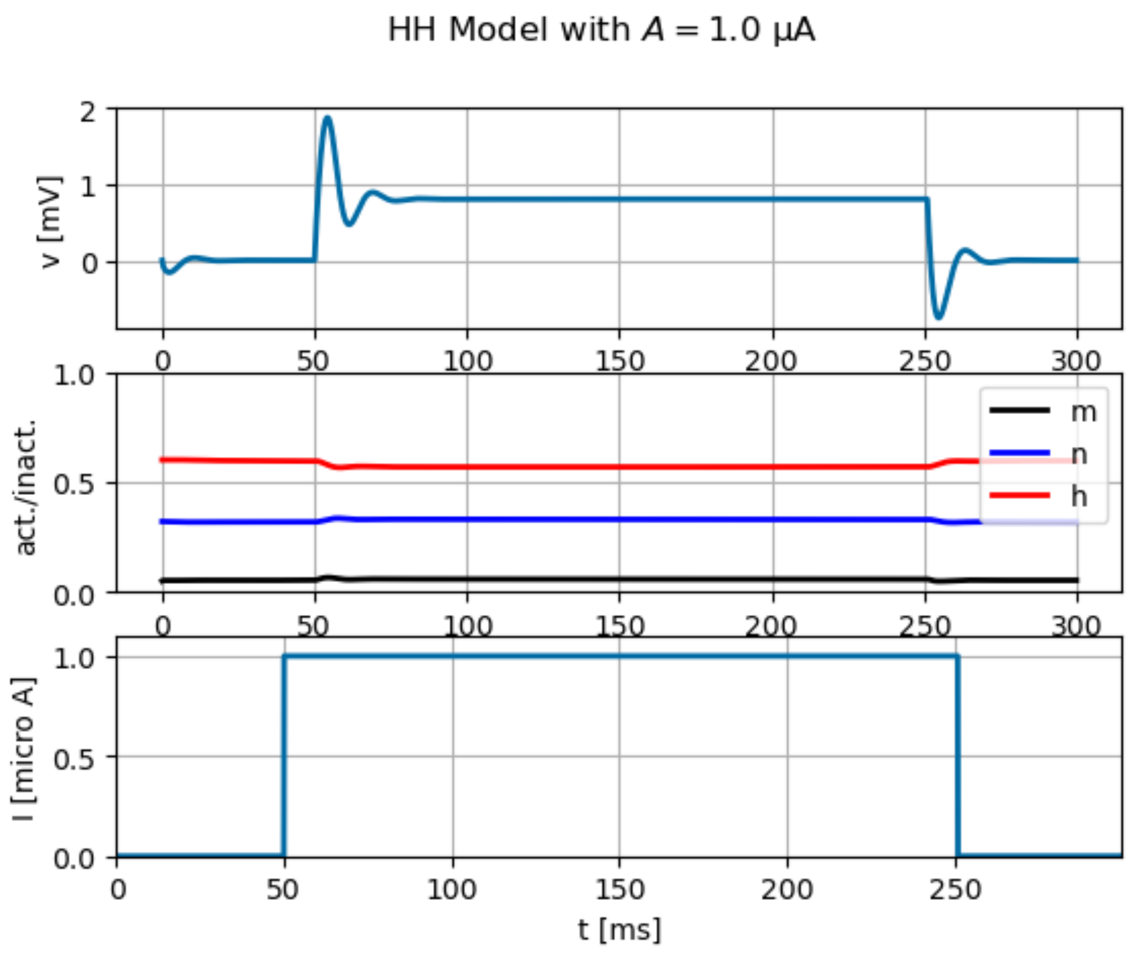
A2.1 Step current response

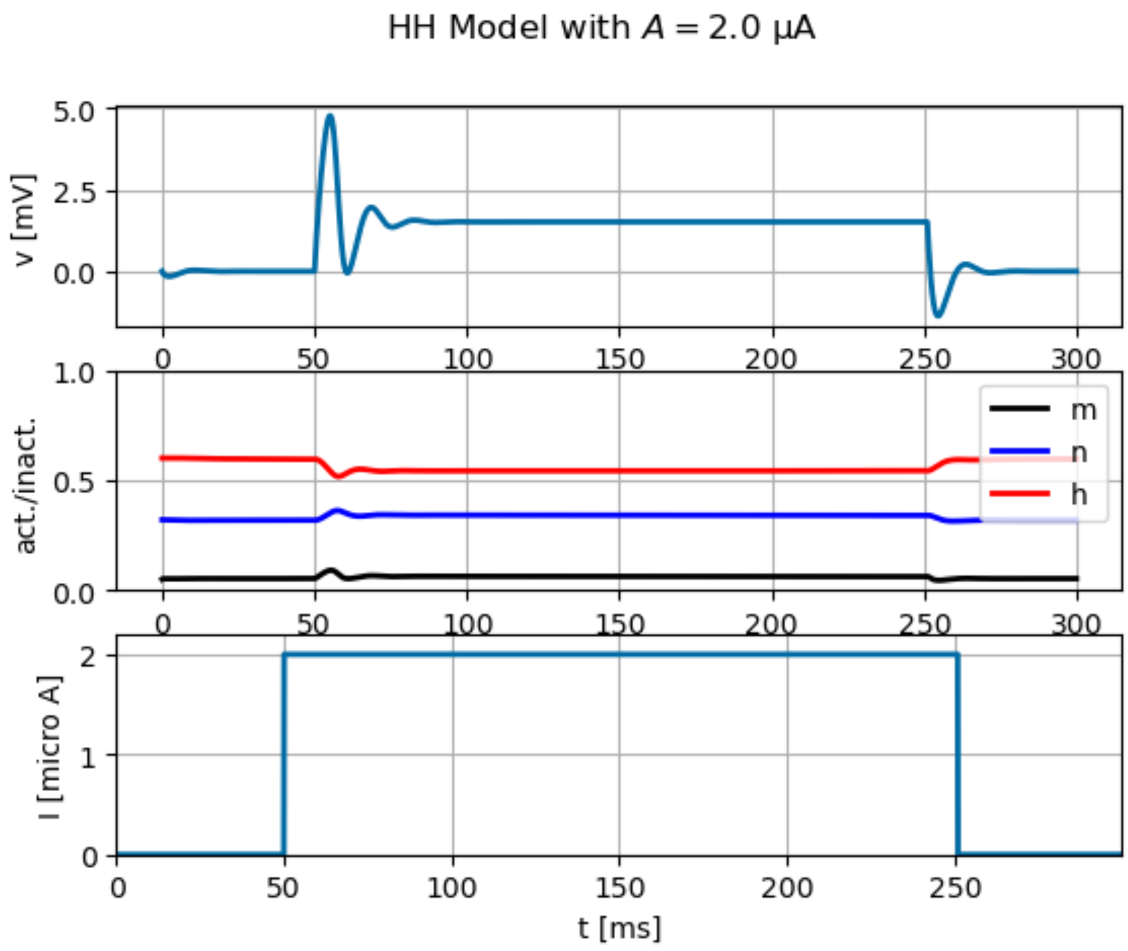
- Go back to Q2.1

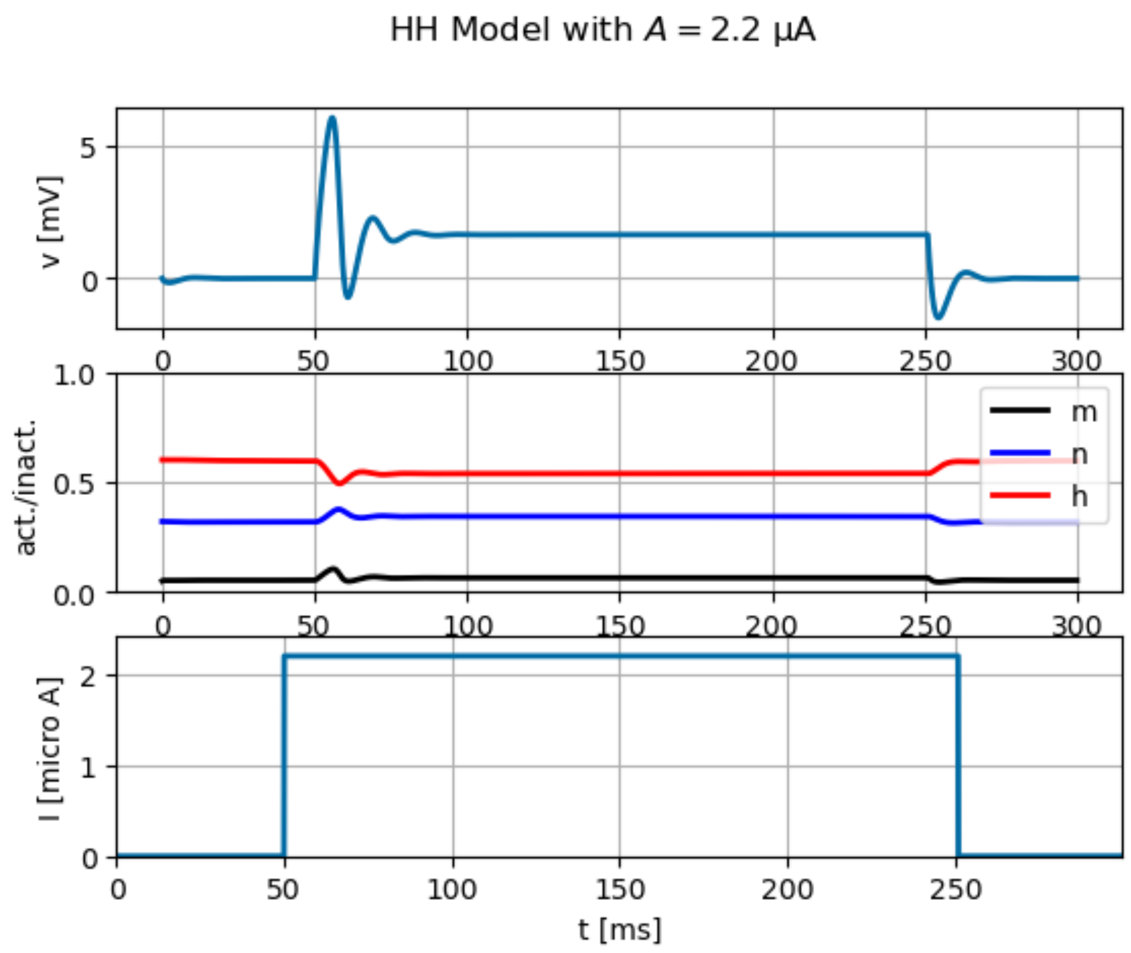
In [14…

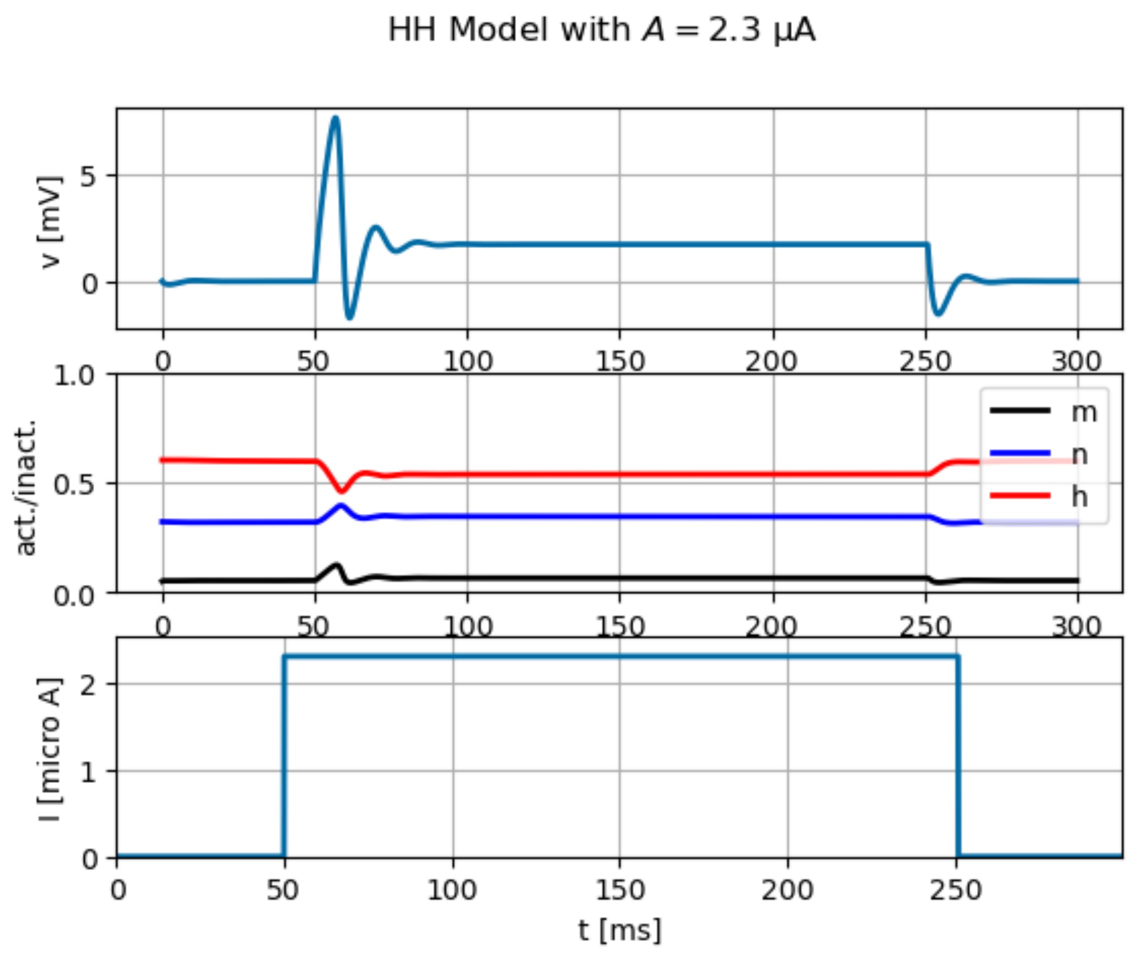
```
# A2.1a Enter your answer below

# Hint: The unit of the I_input current in the neurodynex3.hodgkin_
# 0. Simulation parameters
T_sim = 300*b2.ms
t_start, t_end = 50, 250
Amp = [0.1, 1.0, 2.0, 2.2, 2.3, 2.4, 3.0, 4.0, 5.0, 6.0, 6.1, 6.2,

# 1. Step function for the input current
step_currents = [input_factory.get_step_current(t_start, t_end, b2.

# 2. Run the HH model for 300 ms
state_monitors = [HH.simulate_HH_neuron(step_current, T_sim) for st

# 3. Plot the results of the simulation
for i, state_monitor in enumerate(state_monitors):
    HH.plot_data(state_monitor, title = 'HH Model with $A=' + str(A
WARNING    Cannot use Cython, a test compilation failed: Cython is
not available (ImportError) [brian2.codegen.runtime.cython_rt.cytho
n_rt.failed_compile_test]
INFO       Cannot use compiled code, falling back to the numpy code
generation target. Note that this will likely be slower than using
compiled code. Set the code generation to numpy manually to avoid t
his message:
prefs.codegen.target = "numpy" [brian2.devices.device.codegen_fallb
ack]
```

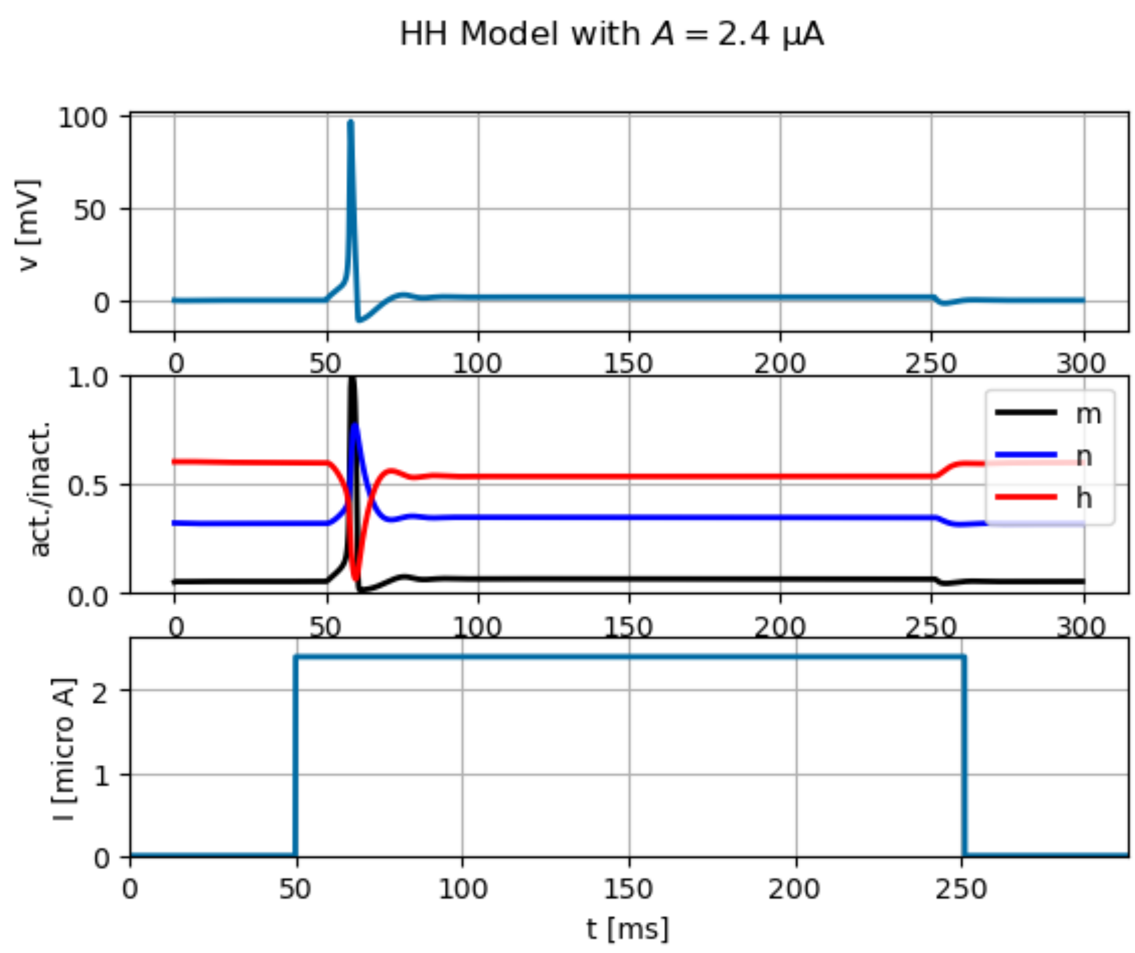


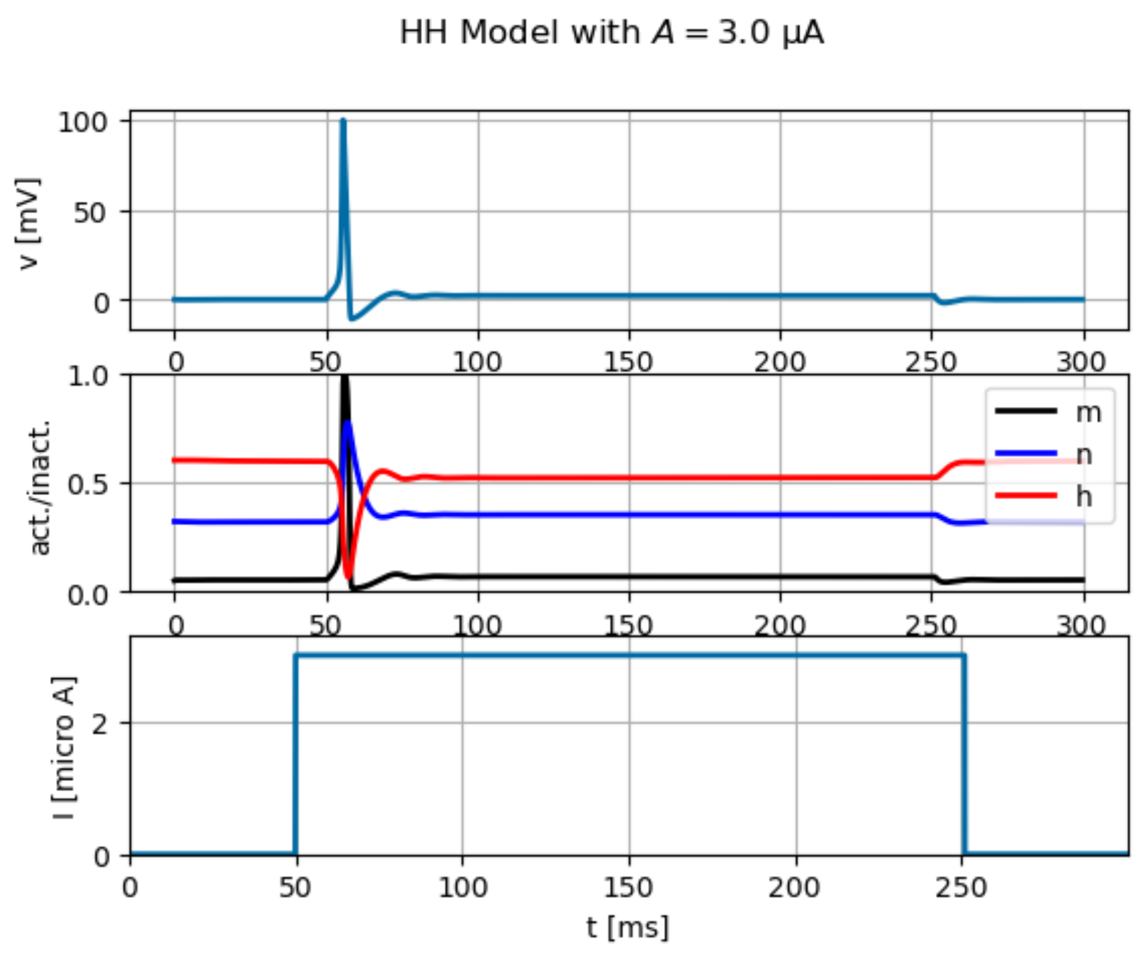

HH Model with $A = 1.0$ µA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

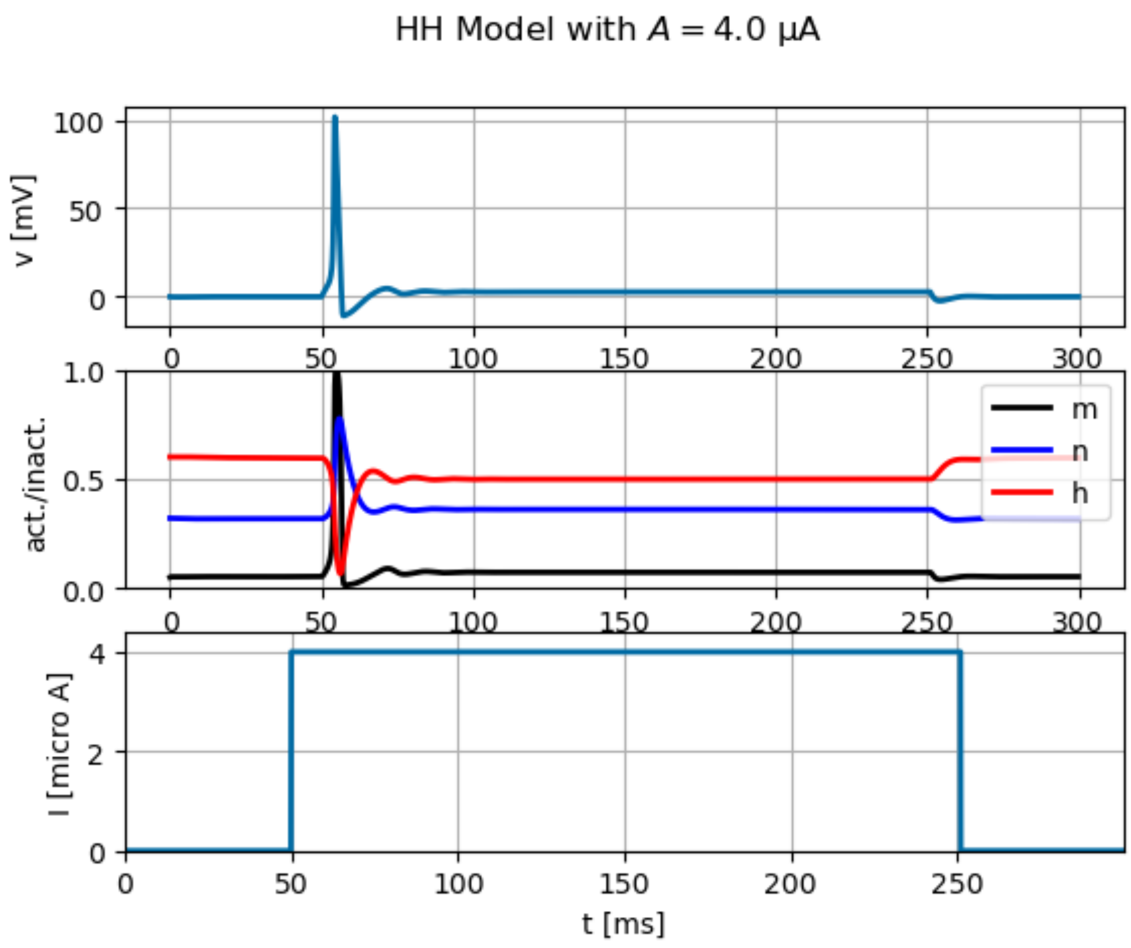

HH Model with $A = 2.0$ µA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

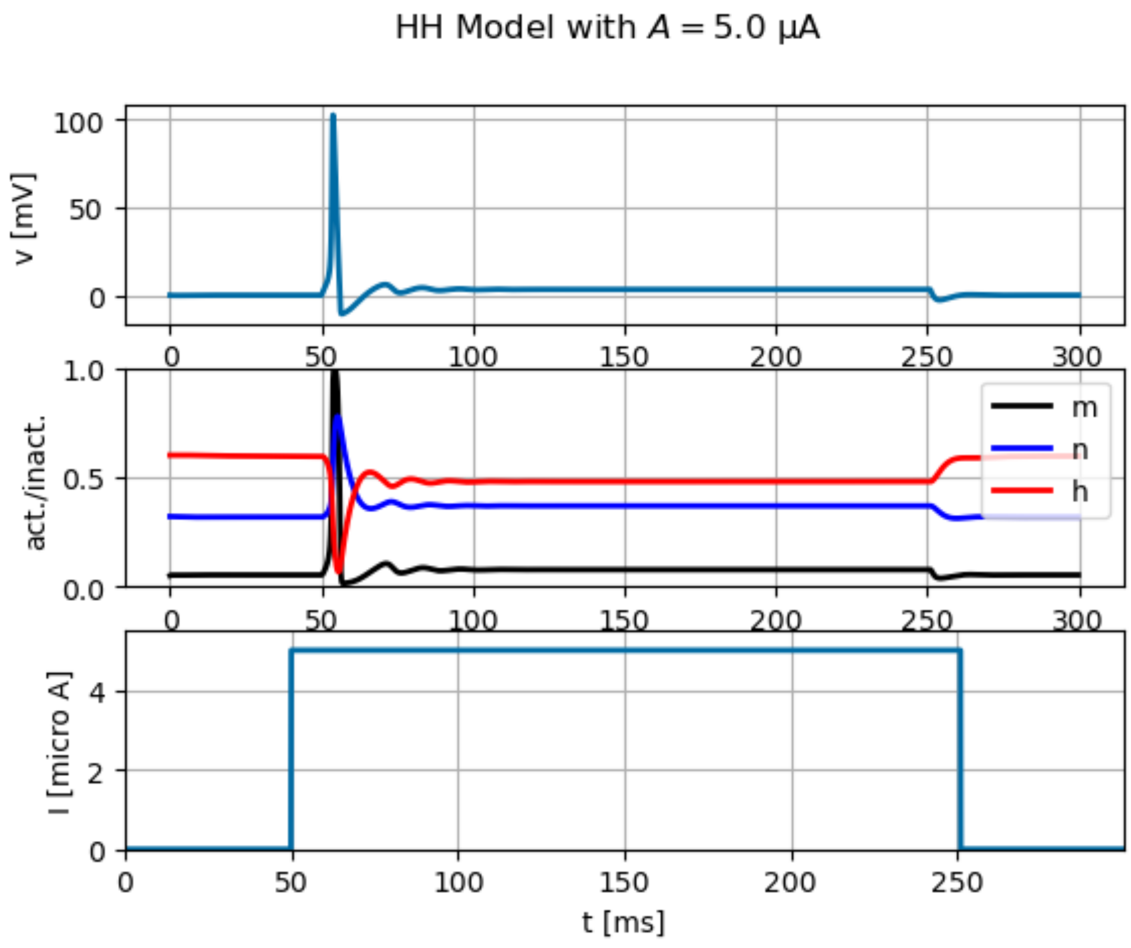

HH Model with $A = 2.2$ μA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

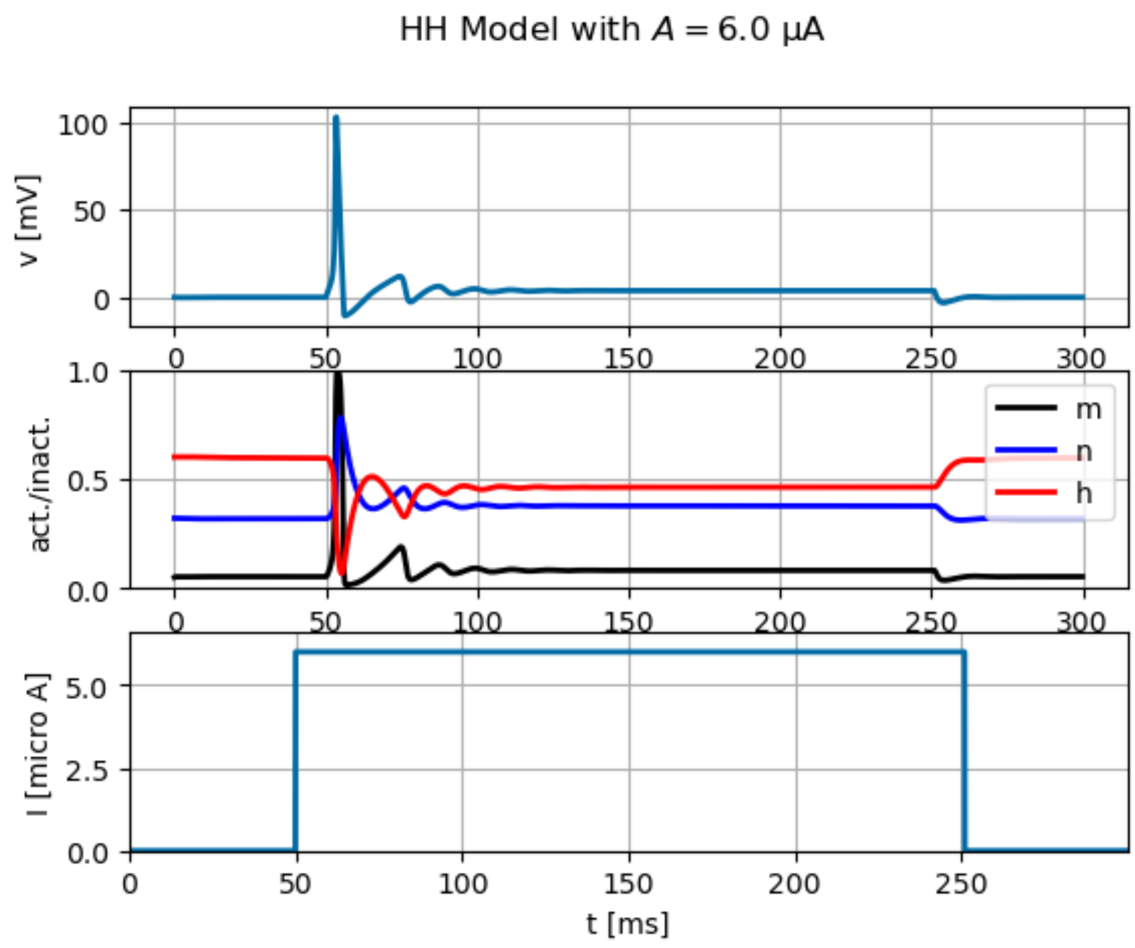

HH Model with $A = 2.3$ μA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

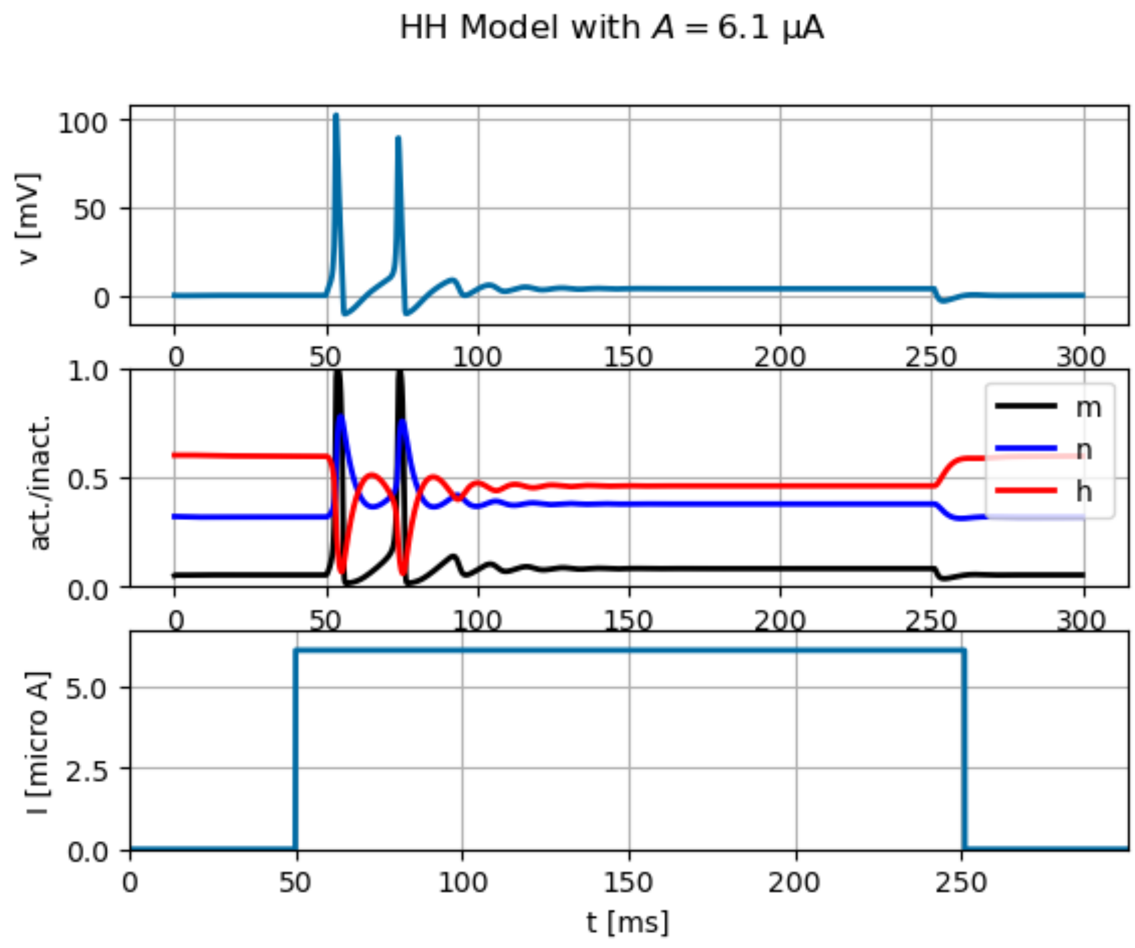

HH Model with $A = 2.4$ µA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

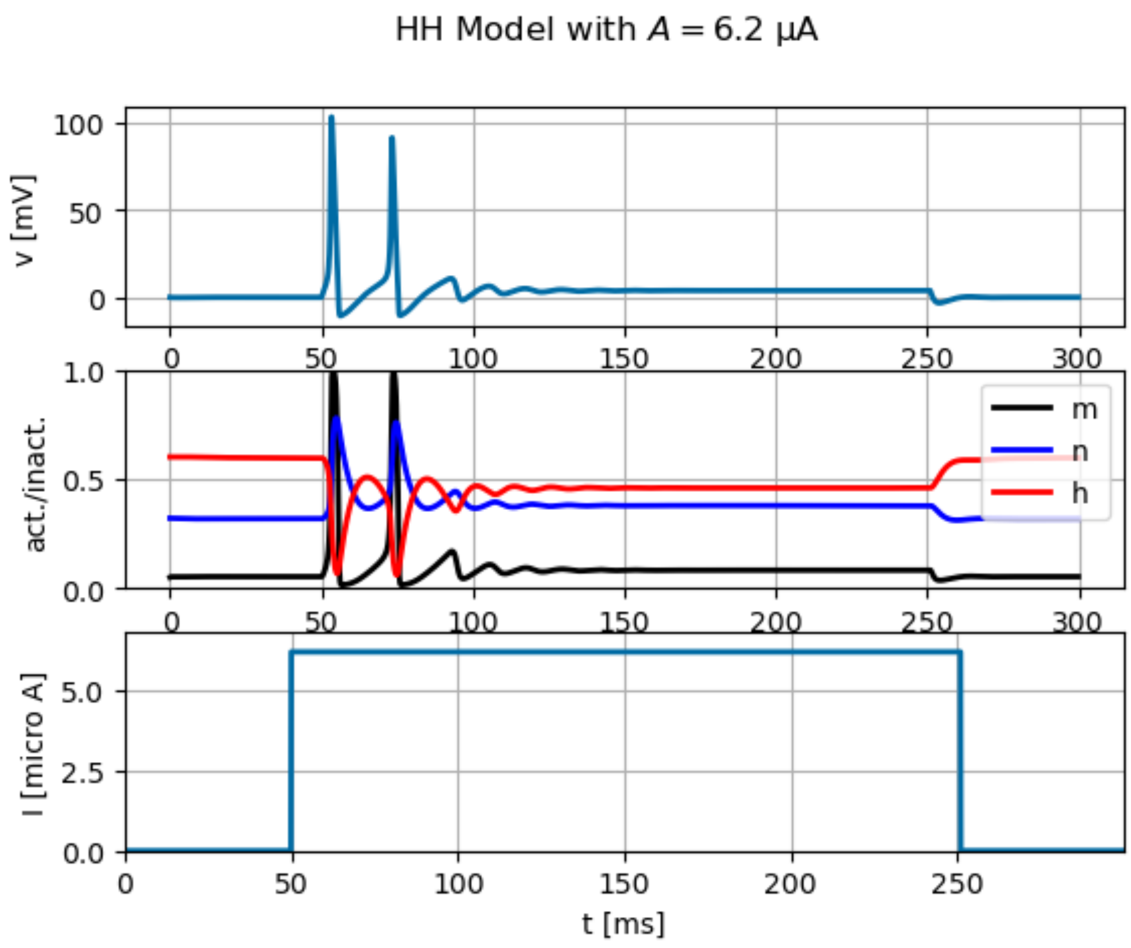

HH Model with $A = 3.0$ µA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

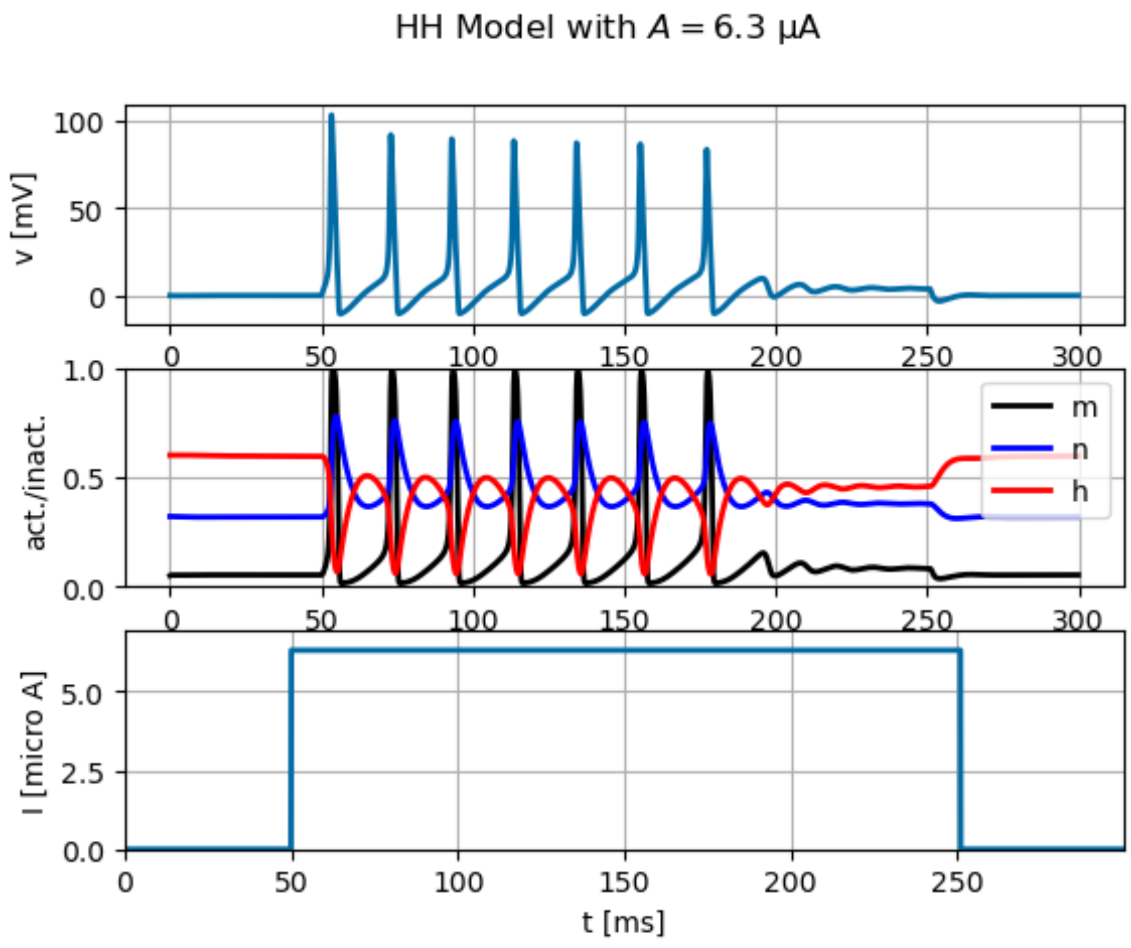

HH Model with $A = 4.0$ µA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

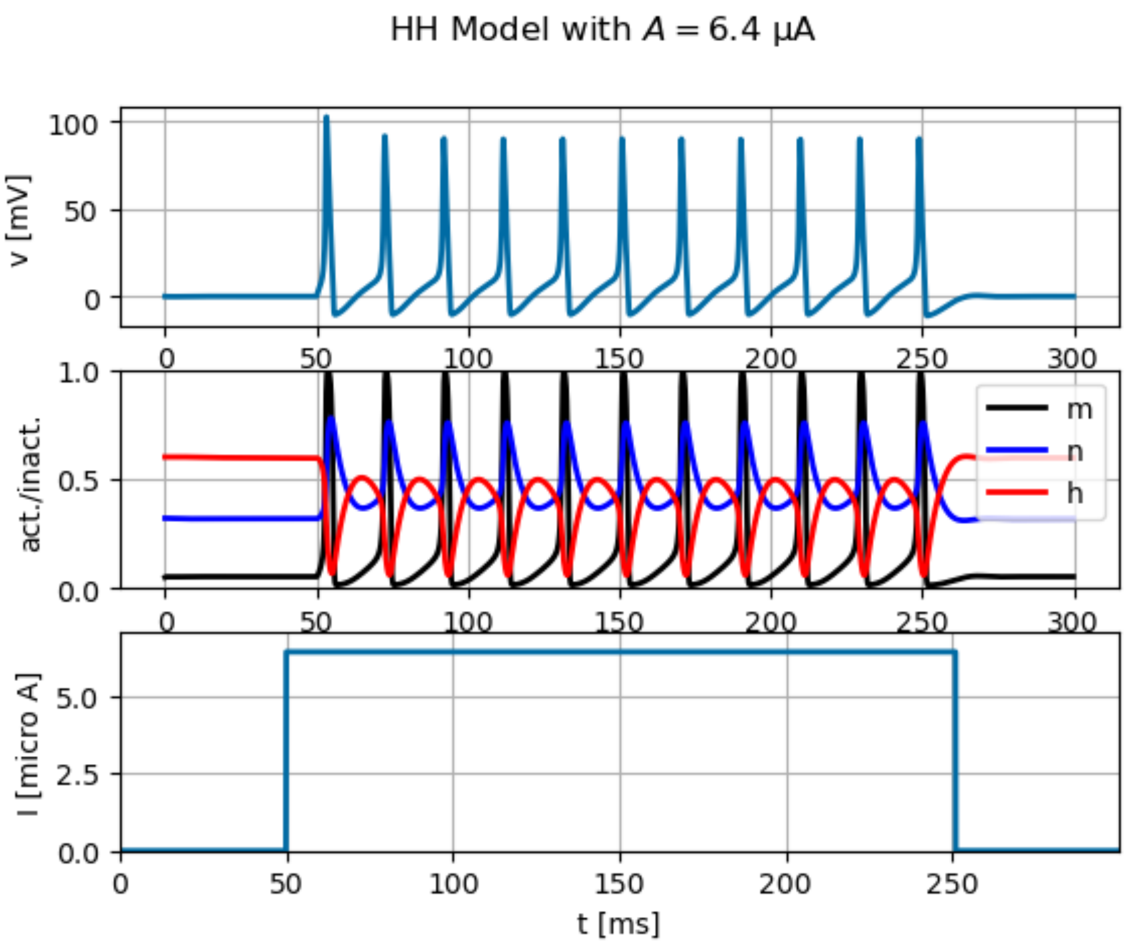

HH Model with $A = 5.0$ µA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

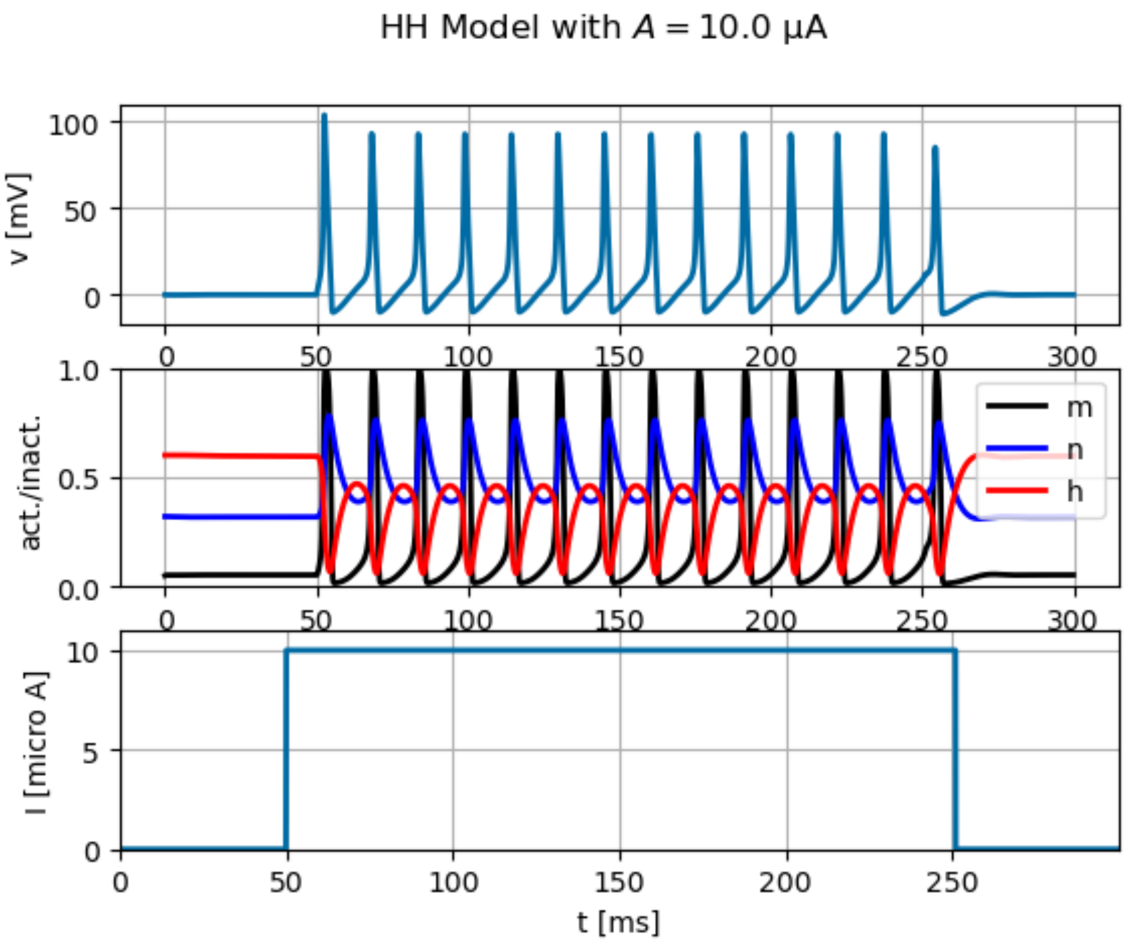

HH Model with $A = 6.0$ µA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

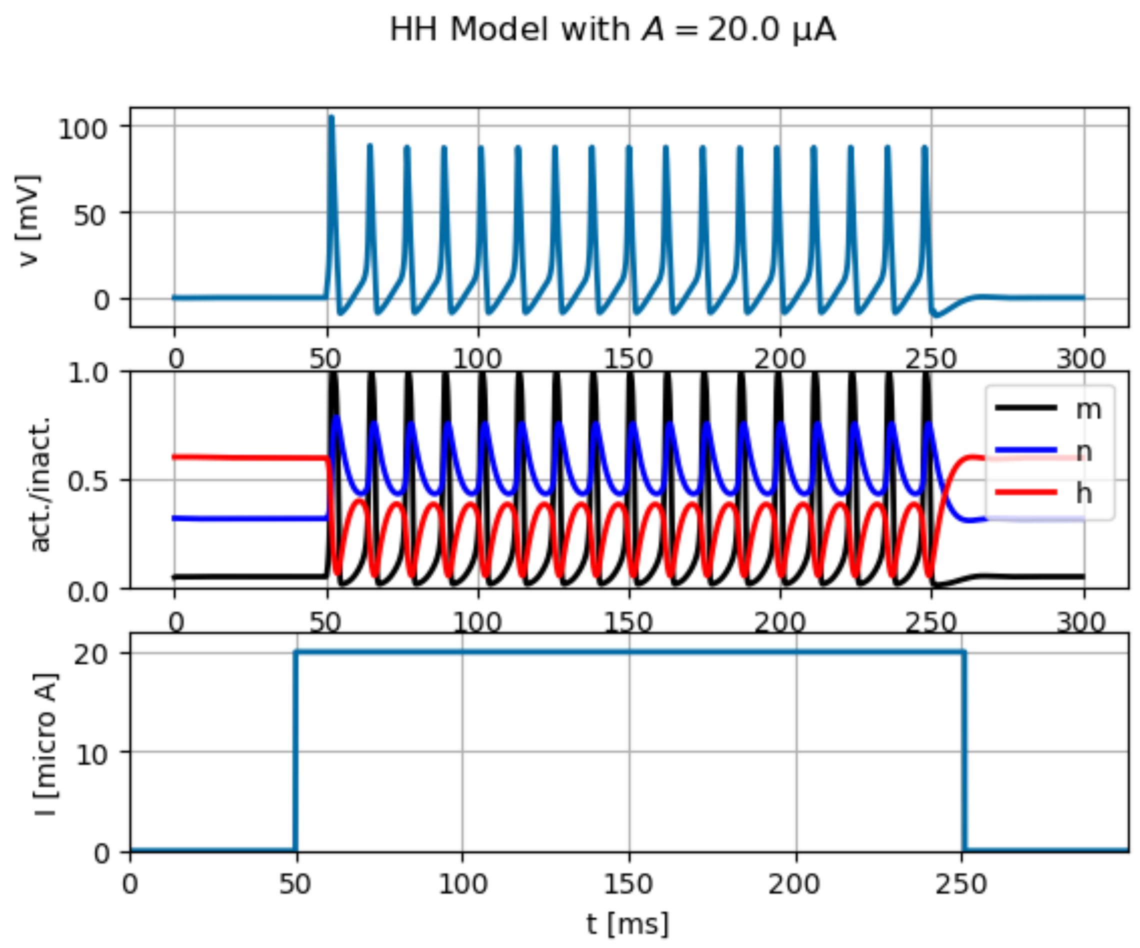

HH Model with $A = 6.1$ µA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

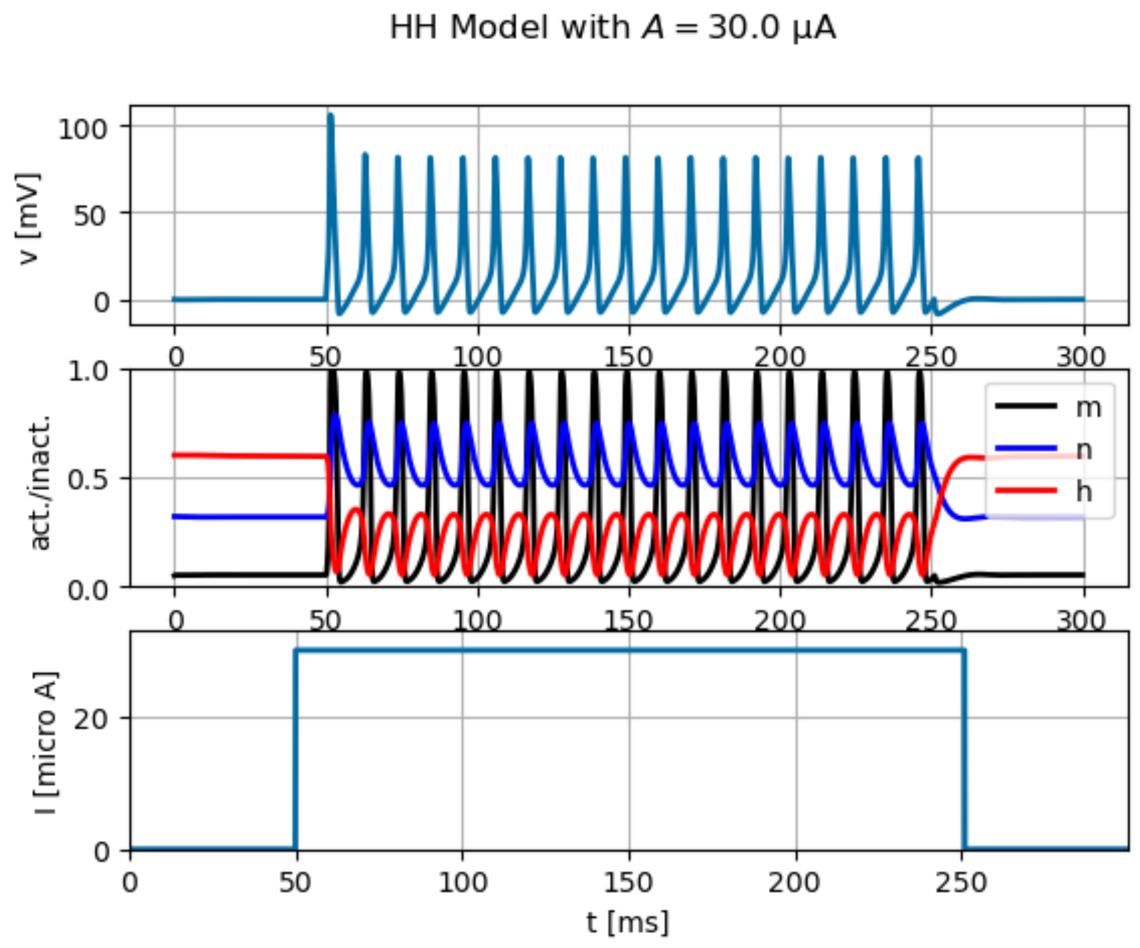

HH Model with $A = 6.2$ μA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

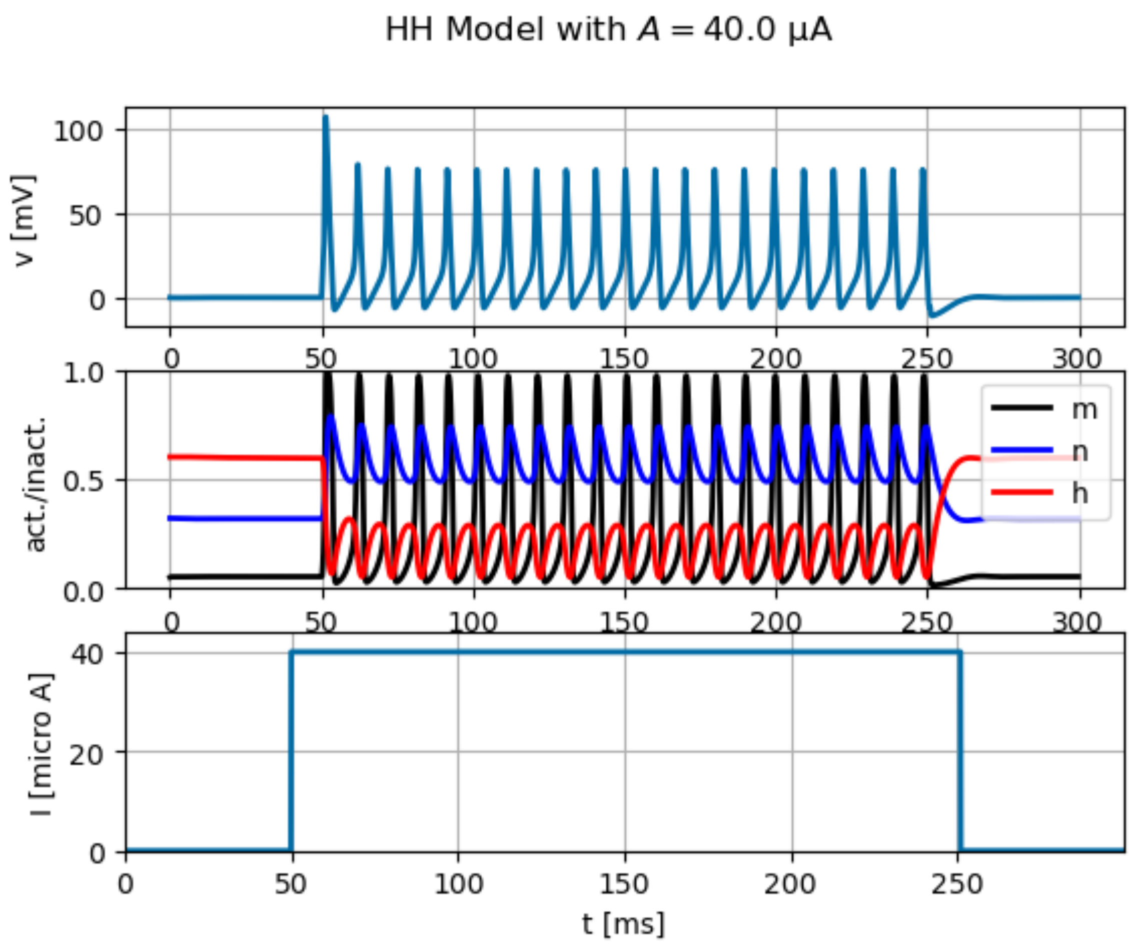

HH Model with $A = 6.3$ μA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

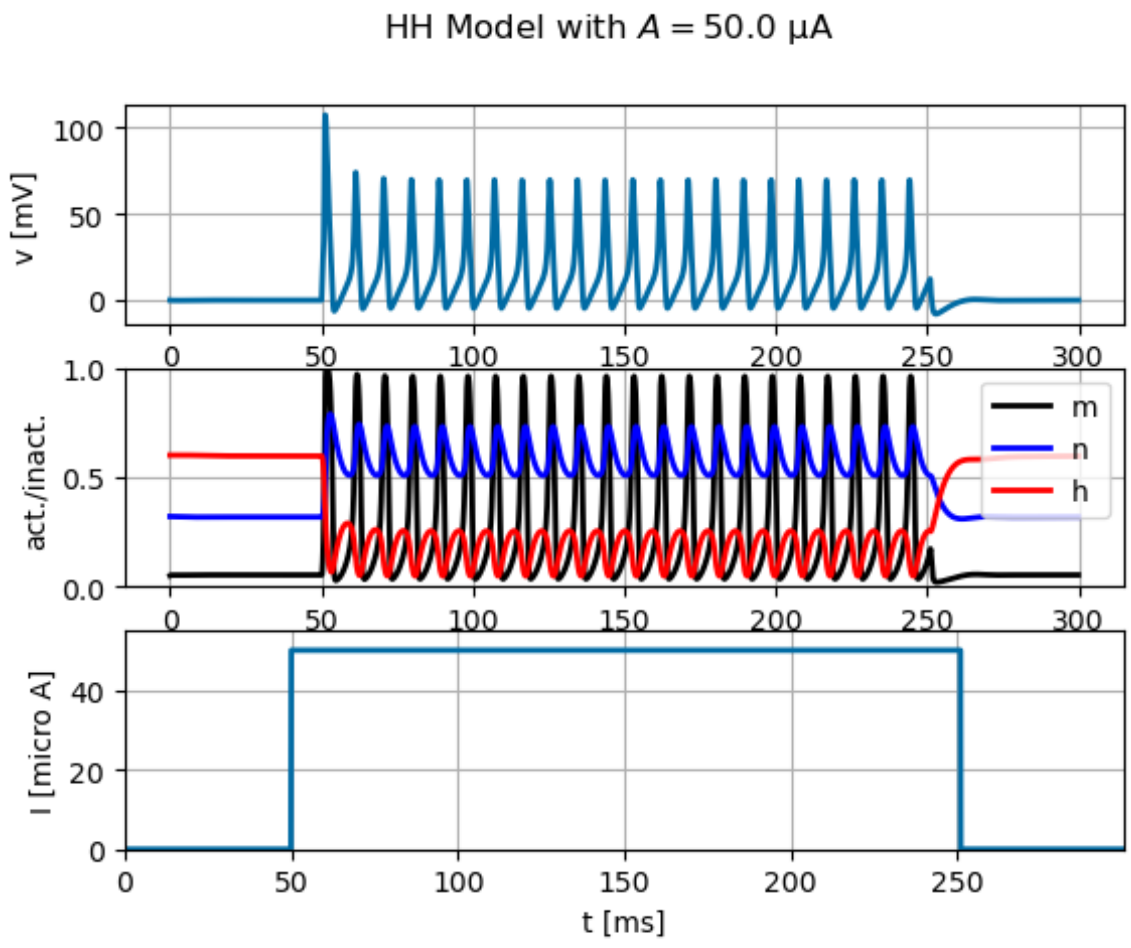

HH Model with $A = 6.4$ µA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]


HH Model with $A = 10.0$ µA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]


HH Model with $A = 20.0$ µA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]


HH Model with $A = 30.0$ µA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]


HH Model with $A = 40.0$ μA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]


HH Model with $A = 50.0$ μA
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

A2.1 conclusion

1. **Non-spiking**
   For neurons, we have a threshold-type behavior: if we apply a current with an amplitude below a certain threshold, the membrane potential will return to its resting potential after the current pulse, without the generation of a spike. In the plots above (0.1 µA, 1.0 µA, 2.0 µA, 2.2 µA, 2.3 µA) you can observe a dampend oscillation at transcient times (start/end of the pulse) and in between a stationary ($u = \mathrm{cst}$) membrane potential. During the application of the current, more Na$^+$ ions are able to enter than K$^+$ ions and cause a small depolarization of the membrane (the reason for this can be found in section A1.4b).

2. **Single spiking**
   Spikes will be generated from input currents with an amplitude of at least 2.3 - 2.4 µA. The dynamics of the spiking neuron is extensively described in part 1 of the task.

3. **Repetitive spiking**
   When the current amplitude is increased to approximately 6.0 - 6.1 µA, repetitive spiking ensues. Further increments in amplitude correspond to higher spiking frequencies.
   Following each spike, a refractory period occurs. To evoke a subsequent potential, additional stimulation is required to counteract (1) the hyperpolarization and (2) the reduced membrane resistance. Certain channels remain open (lower resistance), allowing current leakage from the neuron. Higher input current amplitudes elicit increased stimulation, hastening the attainment of threshold potential

A2.2.1 Slow ramp current

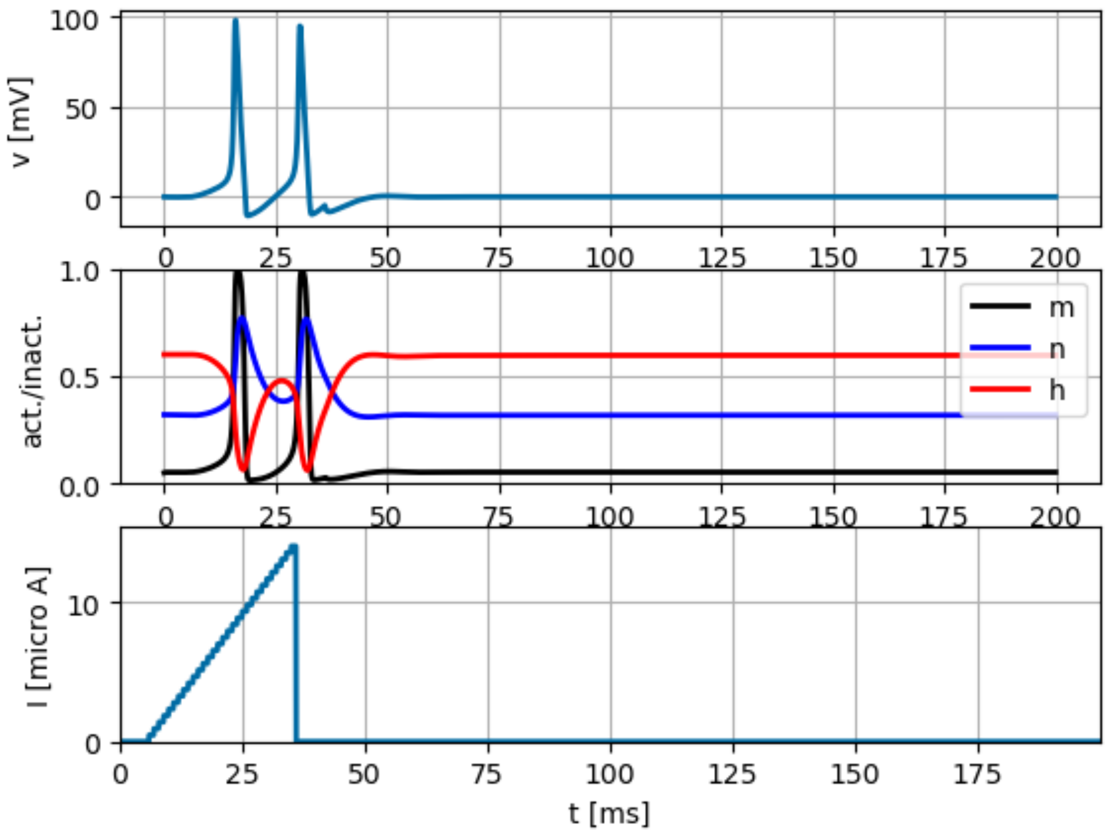
- Go back to Q2.2.1

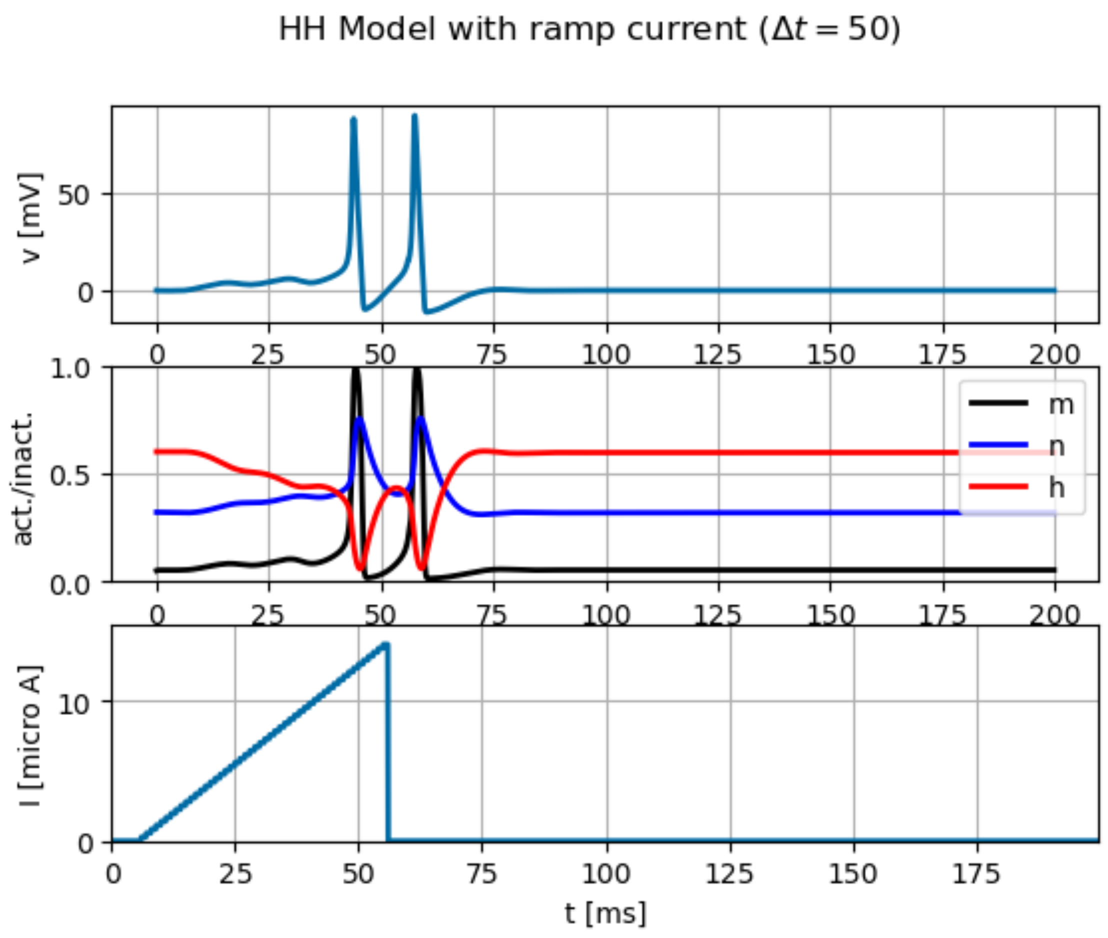
In [16…

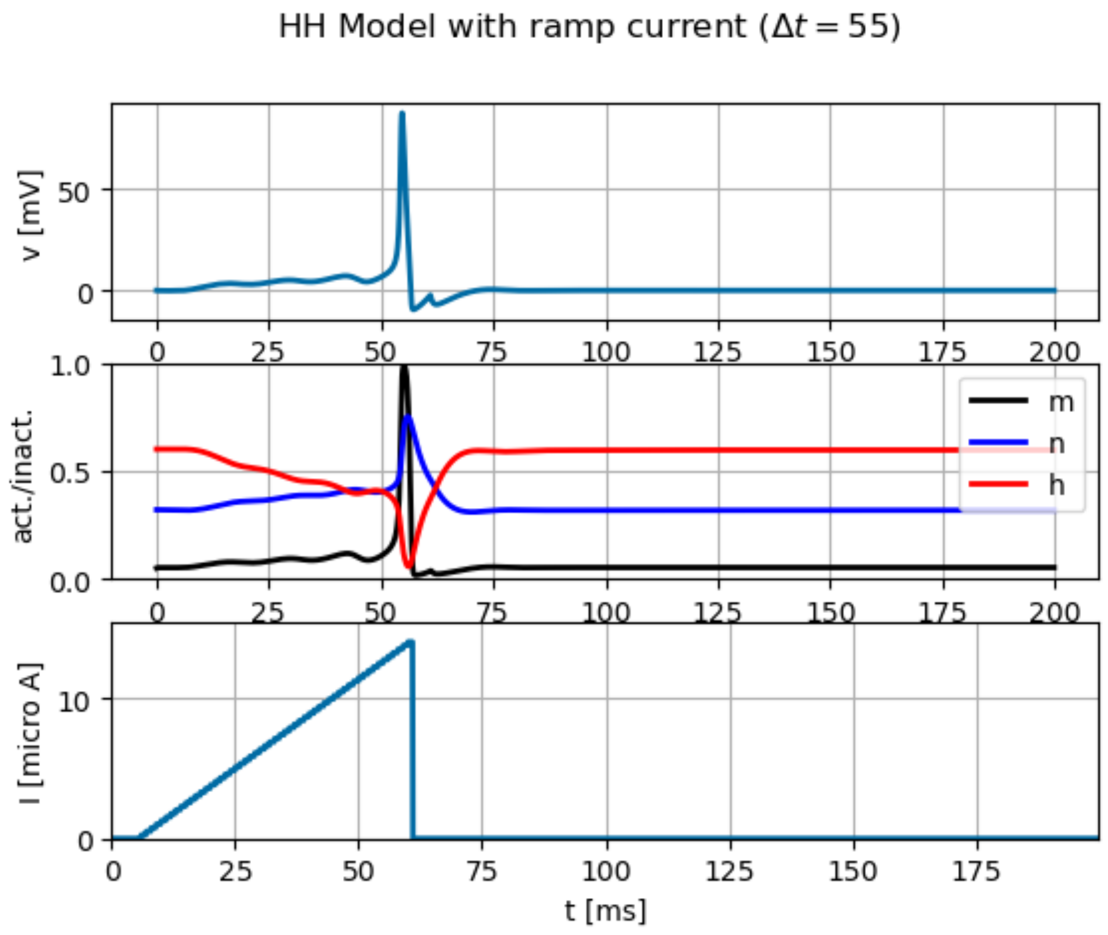
```
# 0. Simulation parameters
T_sim = 200
t_start, t_ends = 5, [35, 55, 60, 65, 70, 75, 95, 105]
A = 14
vms = []

# 1. Step function for the input current
ramp_currents = [input_factory.get_ramp_current(t_start, t_end, b

# 2. Run the HH model for 300 ms
```

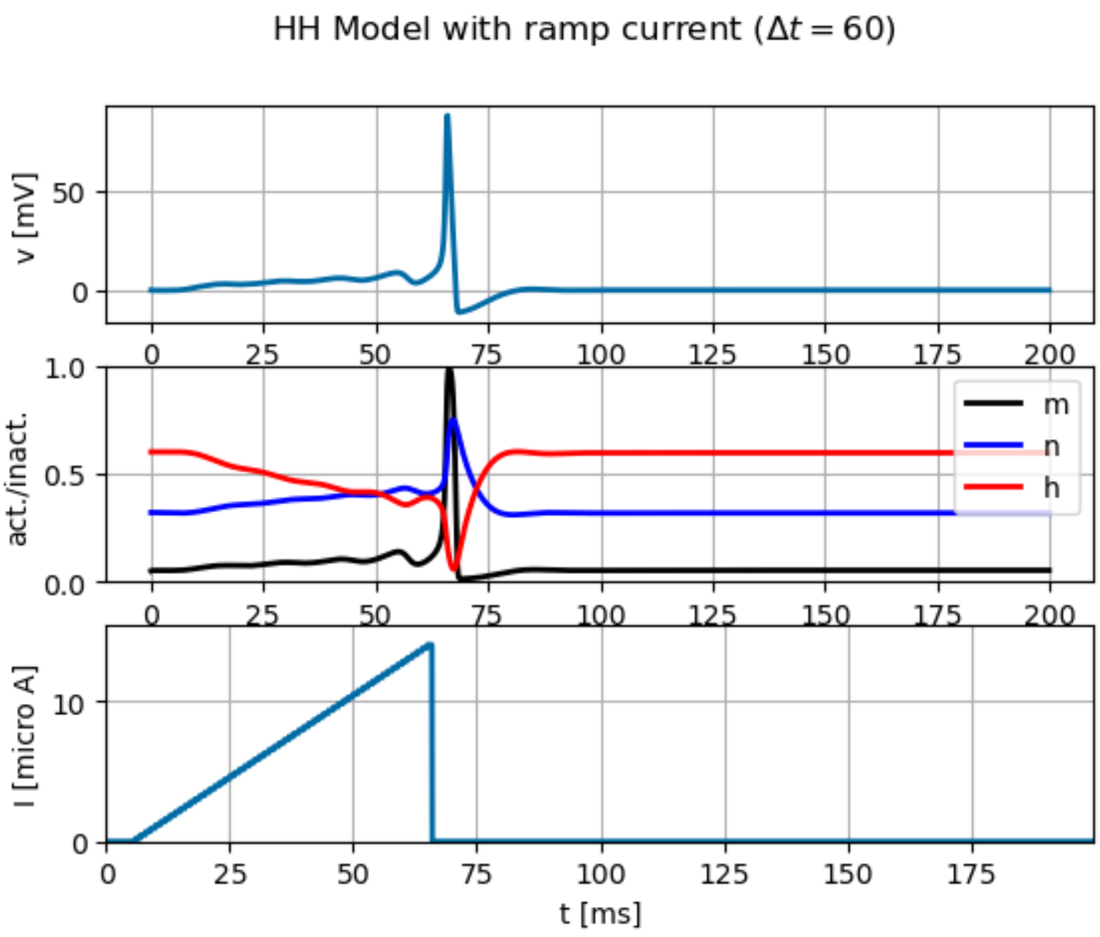
```
state_monitors_ramp_currents = [HH.simulate_HH_neuron(ramp_curren

# 3. Plot the results of the simulation
for i, state_monitor_ramp_current in enumerate(state_monitors_ram
    # duration
    duration = t_ends[i] - t_start

    # plot the data
    HH.plot_data(state_monitor_ramp_current, title = 'HH Model wi

    # get the membrane voltages
    sampling_time = T_sim/len(state_monitor_ramp_current.vm[0])
    vm = state_monitor_ramp_current.vm[0][int(t_ends[i] / samplin
    vms.append((duration, vm))
```

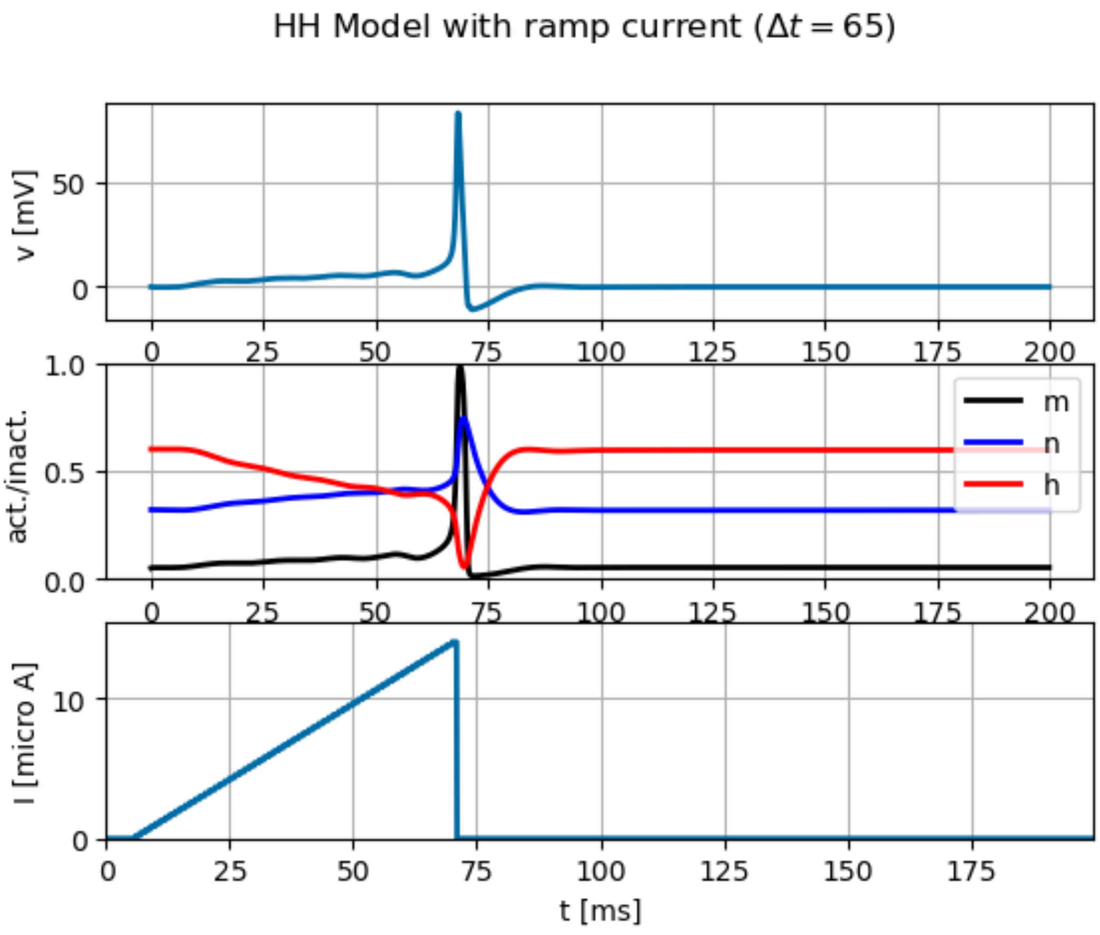
HH Model with ramp current ($\Delta t = 30$)

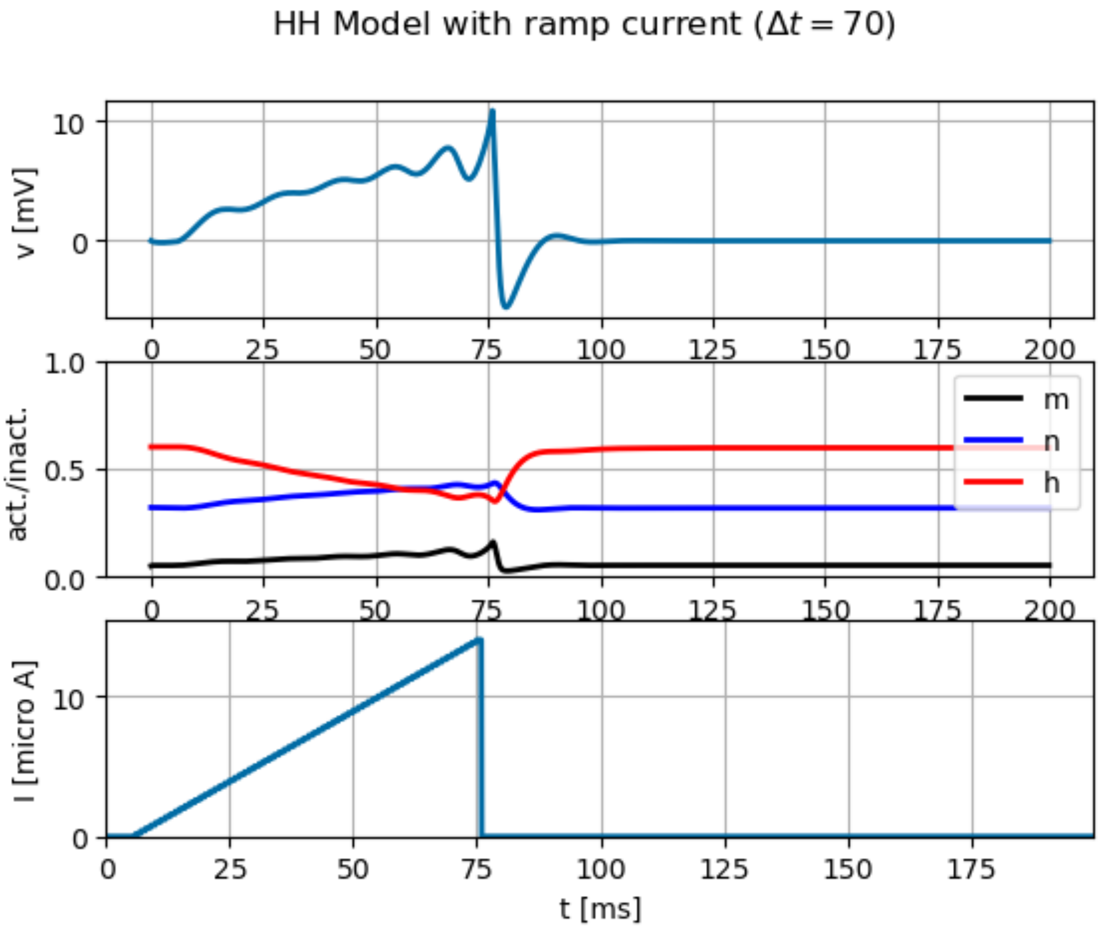


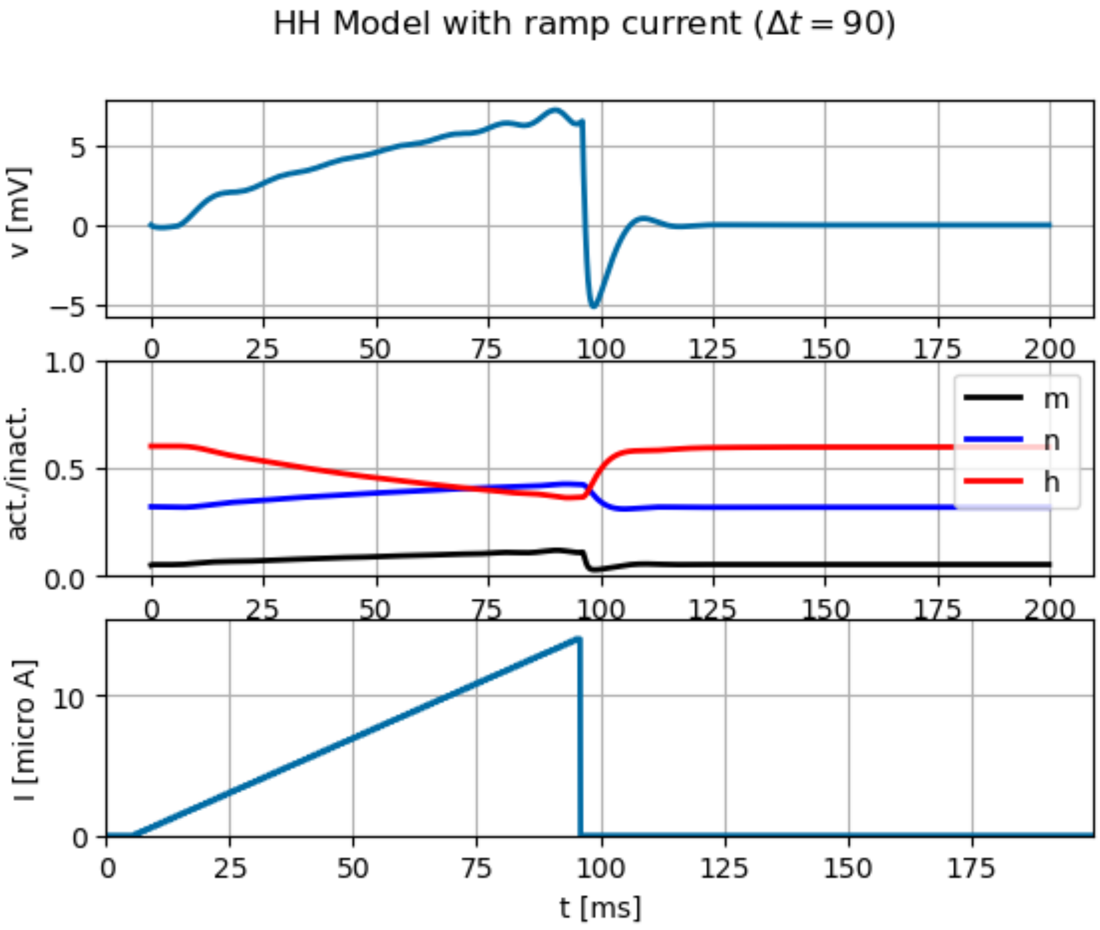

HH Model with ramp current ($\Delta t = 50$)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

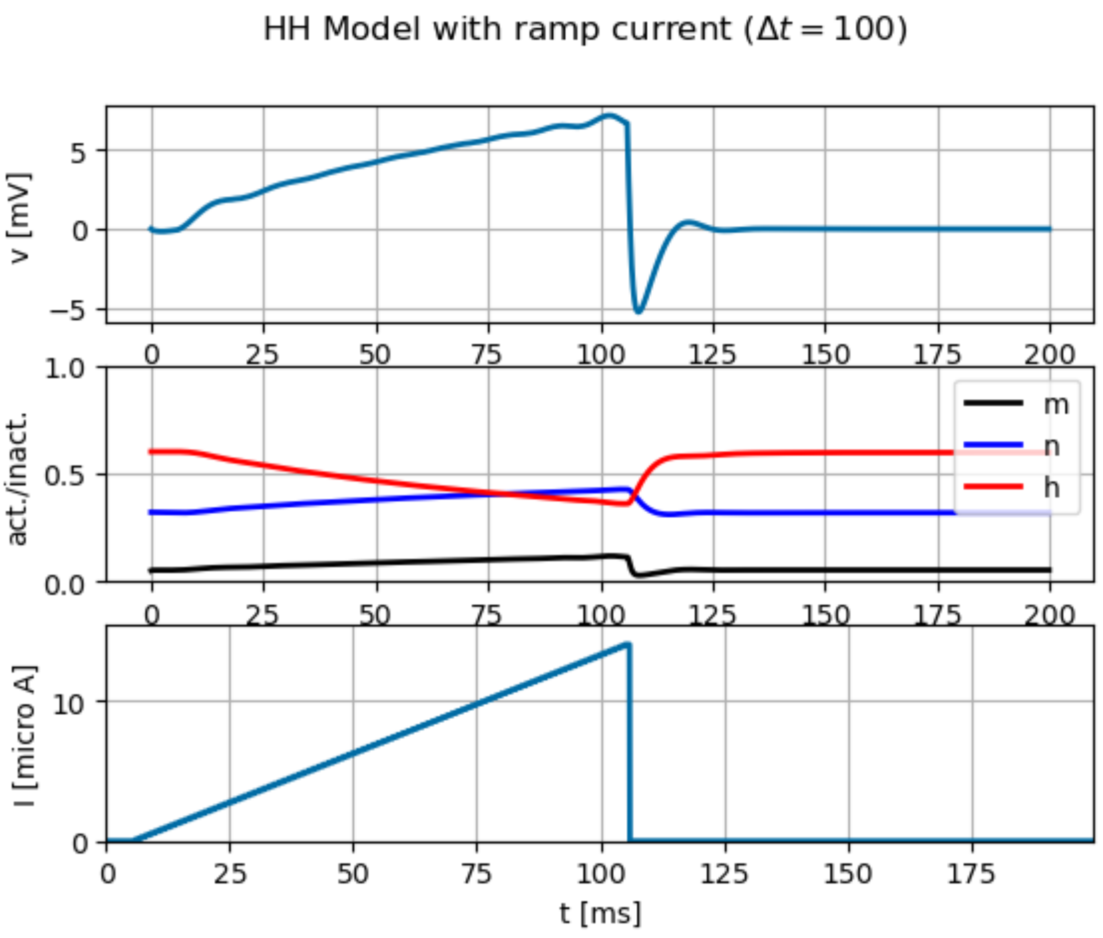

HH Model with ramp current ($\Delta t = 55$)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]


HH Model with ramp current ($\Delta t = 60$)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]


HH Model with ramp current ($\Delta t = 65$)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]


HH Model with ramp current ($\Delta t = 70$)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]


HH Model with ramp current ($\Delta t = 90$)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]



A2.2.1a conclusion

There occurs no spiking for a slow ramp current (max amplitude 14 µA) with a duration higher than $\approx 65 - 70$ ms (slope: 0.22 - 0.20 mA/s).

In [17…

```python
for duration, vm in vms:
    print('Membrane potential at t = t_end:', vm, ', duration:',
```

```
Membrane potential at t = t_end: -7.00579634 mV , duration: 30 ms
Membrane potential at t = t_end: 9.09742189 mV , duration: 50 ms
Membrane potential at t = t_end: -4.8078393 mV , duration: 55 ms
Membrane potential at t = t_end: 21.81022114 mV , duration: 60 ms
Membrane potential at t = t_end: 9.54408713 mV , duration: 65 ms
Membrane potential at t = t_end: 8.86590723 mV , duration: 70 ms
Membrane potential at t = t_end: 6.33381213 mV , duration: 90 ms
Membrane potential at t = t_end: 6.78747867 mV , duration: 100 ms
```

A2.2.1b conclusion

Δt **(ms)**	V_m (t_{end}) **(mV)**
30	-7.01

50	9.10
55	-4.81
60	21.8
65	9.54
70	8.87
90	6.33
100	6.79

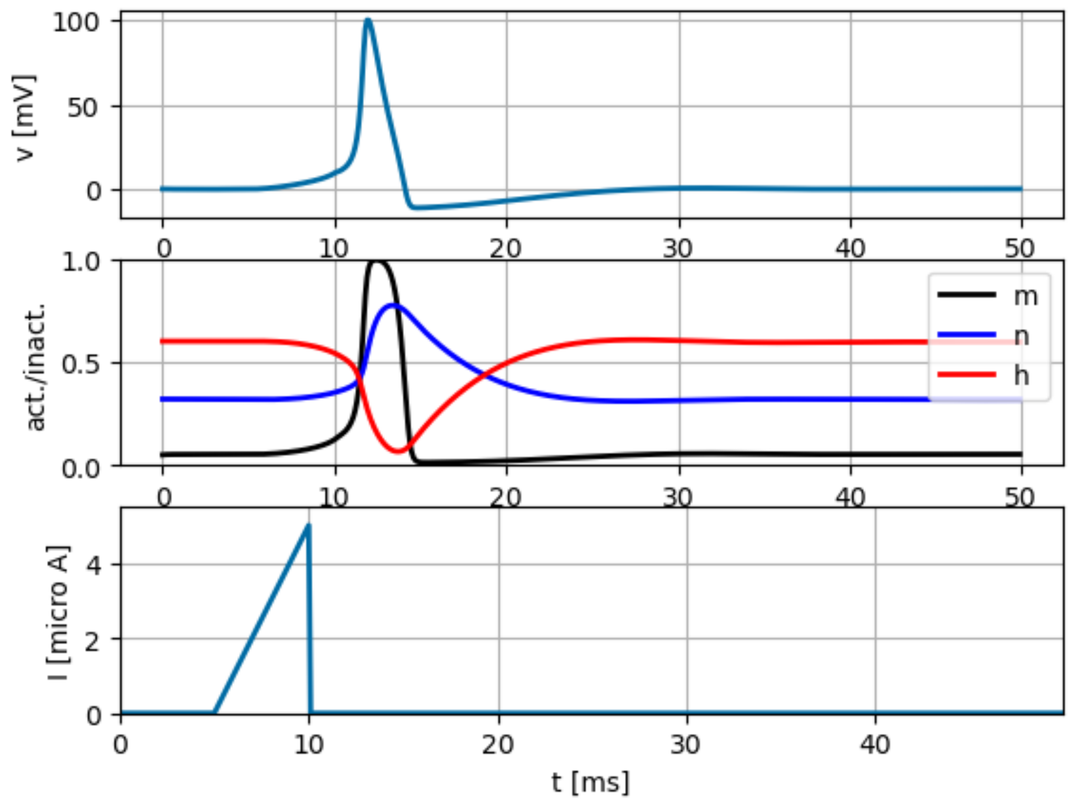
A2.2.2 Fast ramp current

- Go back to Q2.2.2

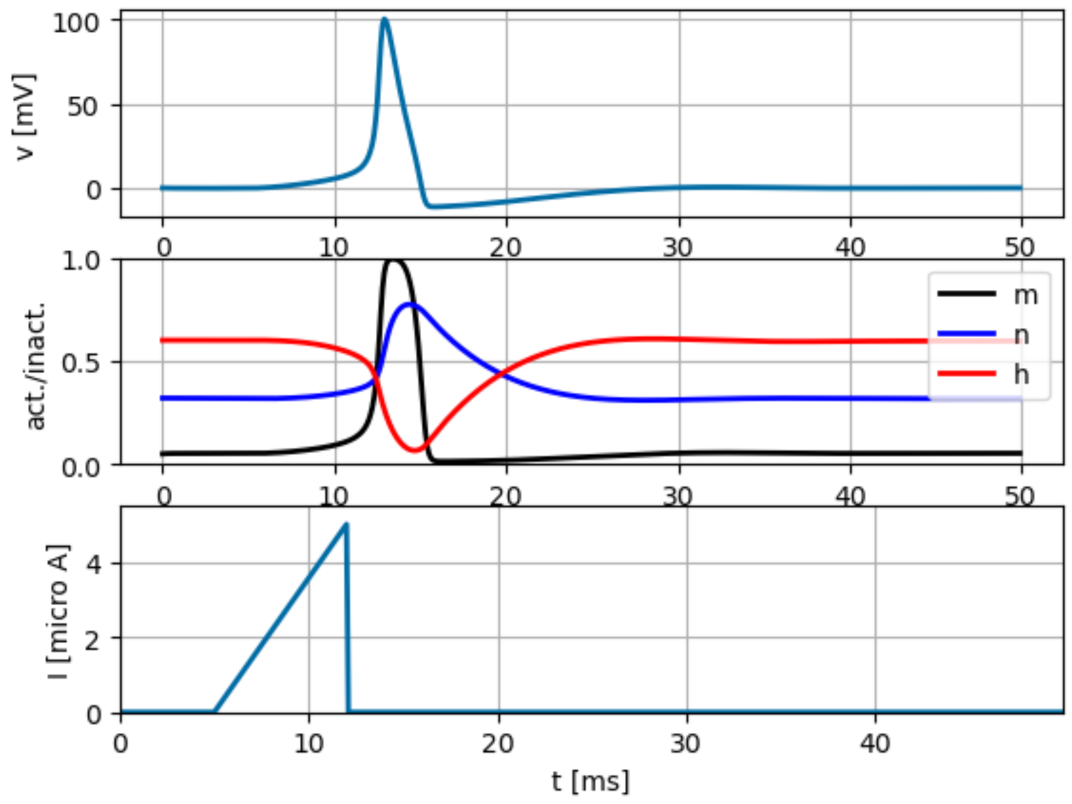
In [18…

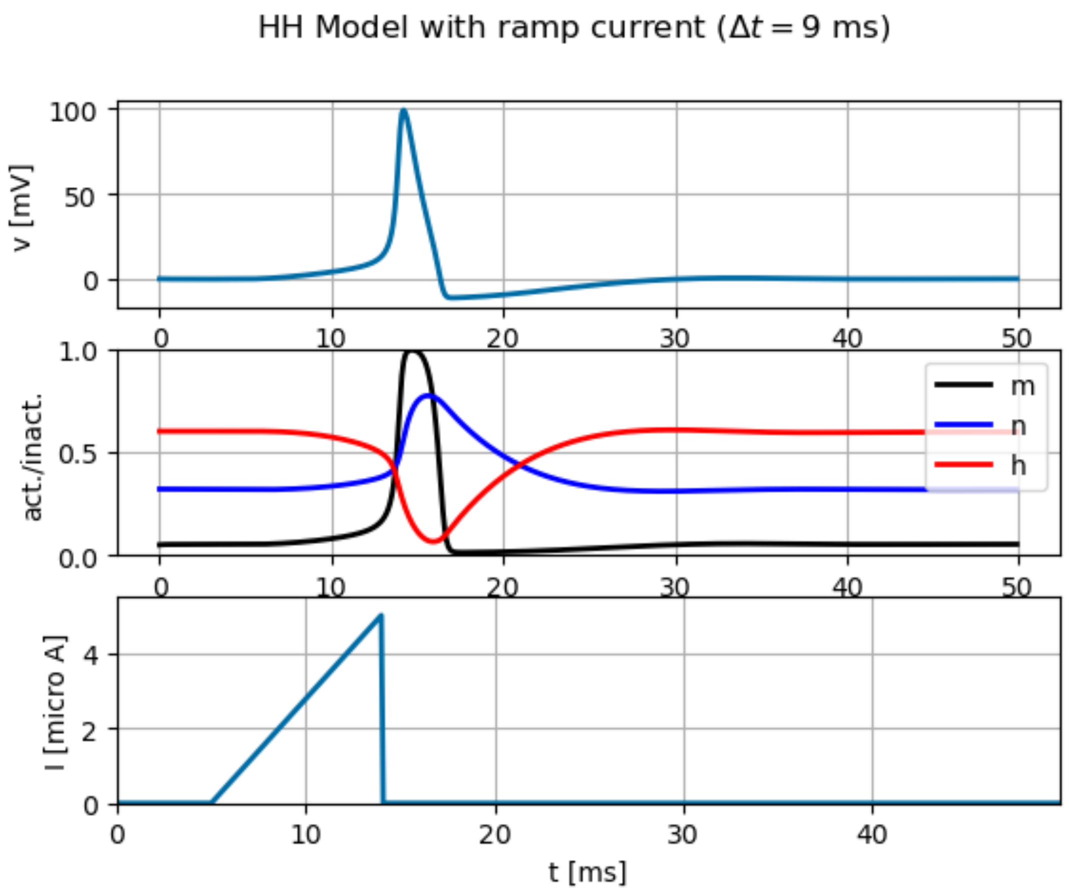
```
# Hint: change the unit time to 0.1ms to get a smooth ramp curr
# 0. Simulation parameters
T_sim = 50
t_start, t_ends = 5, [10, 12, 14, 16, 17, 18, 19, 20]
A = 5.0
vms = []

# 1. Step function for the input current
ramp_currents = [input_factory.get_ramp_current(t_start * 10, i

# 2. Run the HH model for 300 ms
state_monitors_ramp_currents = [HH.simulate_HH_neuron(ramp_curr

# 3. Plot the results of the simulation
for i, state_monitor_ramp_current in enumerate(state_monitors_r
    duration = t_ends[i] - t_start
    HH.plot_data(state_monitor_ramp_current, title = 'HH Model

    # get the membrane voltages
    sampling_time = T_sim / len(state_monitor_ramp_current.vm[0
    vm = state_monitor_ramp_current.vm[0][int(t_ends[i] / sampl
    vms.append((duration, vm))
```

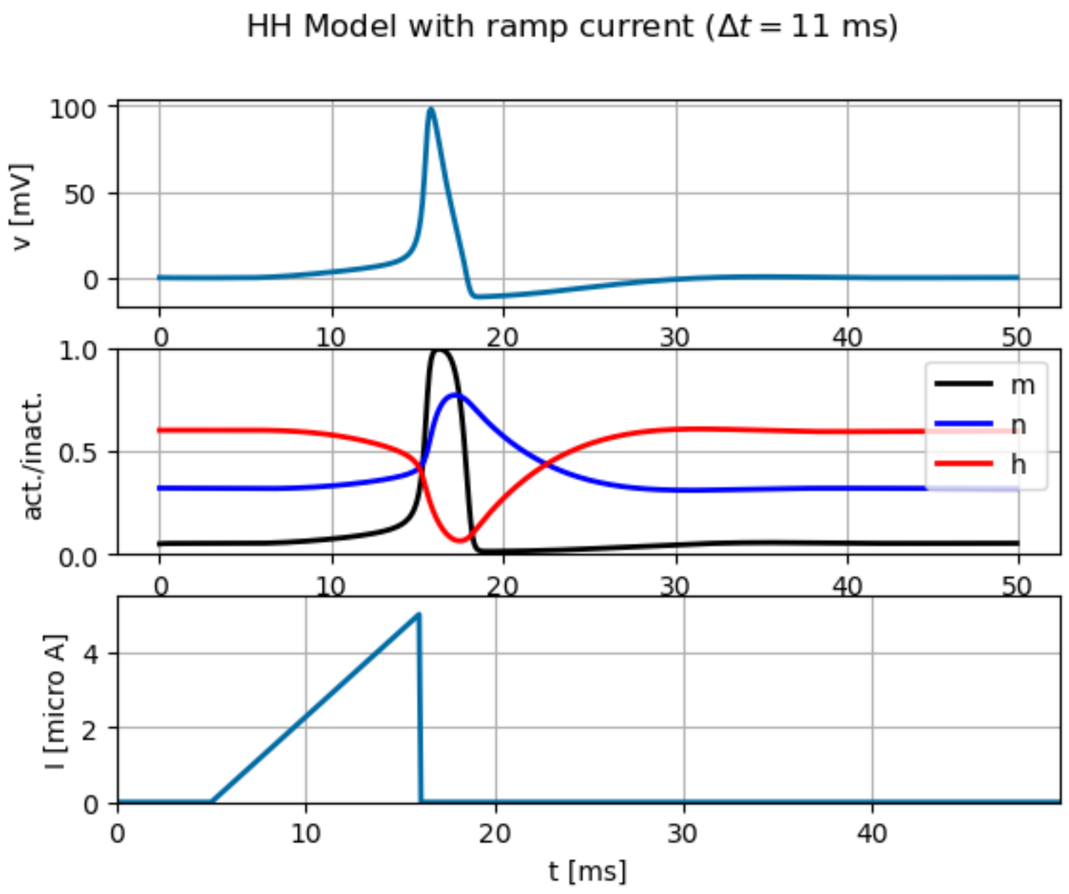
HH Model with ramp current ($\Delta t = 5$ ms)

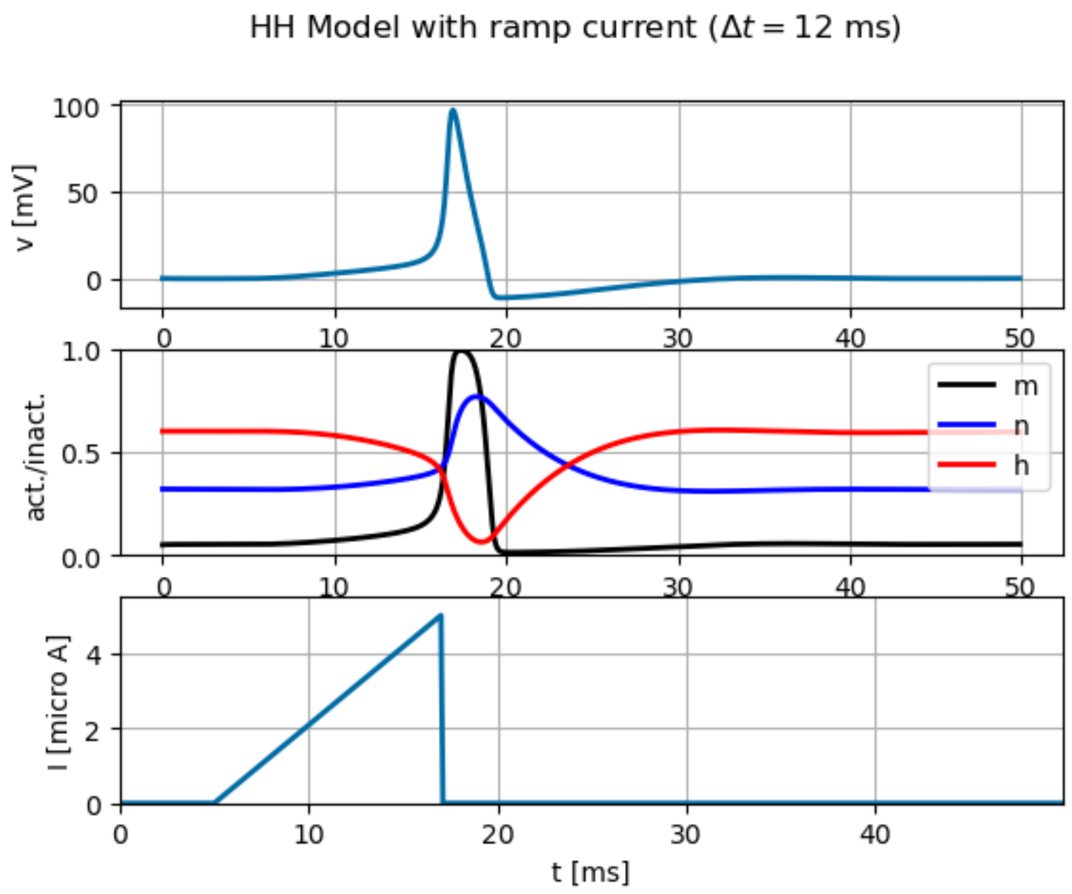


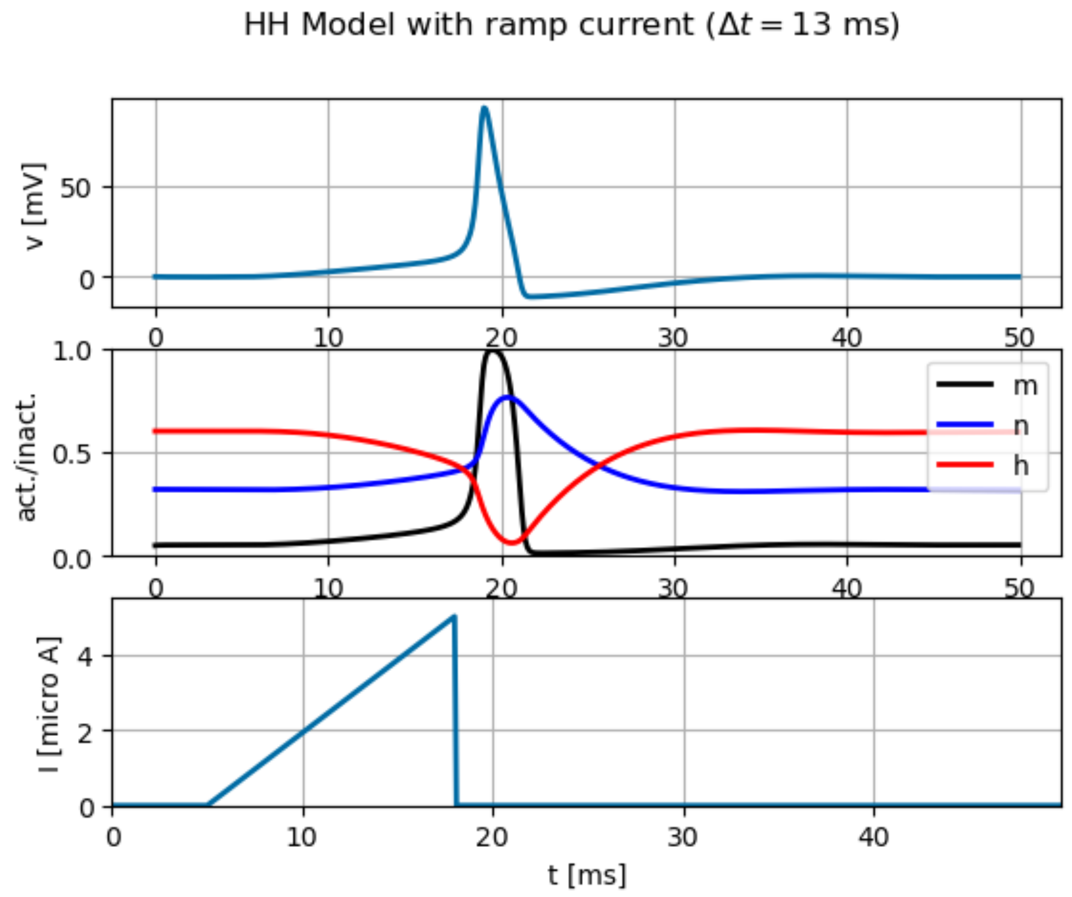
HH Model with ramp current ($\Delta t = 7$ ms)

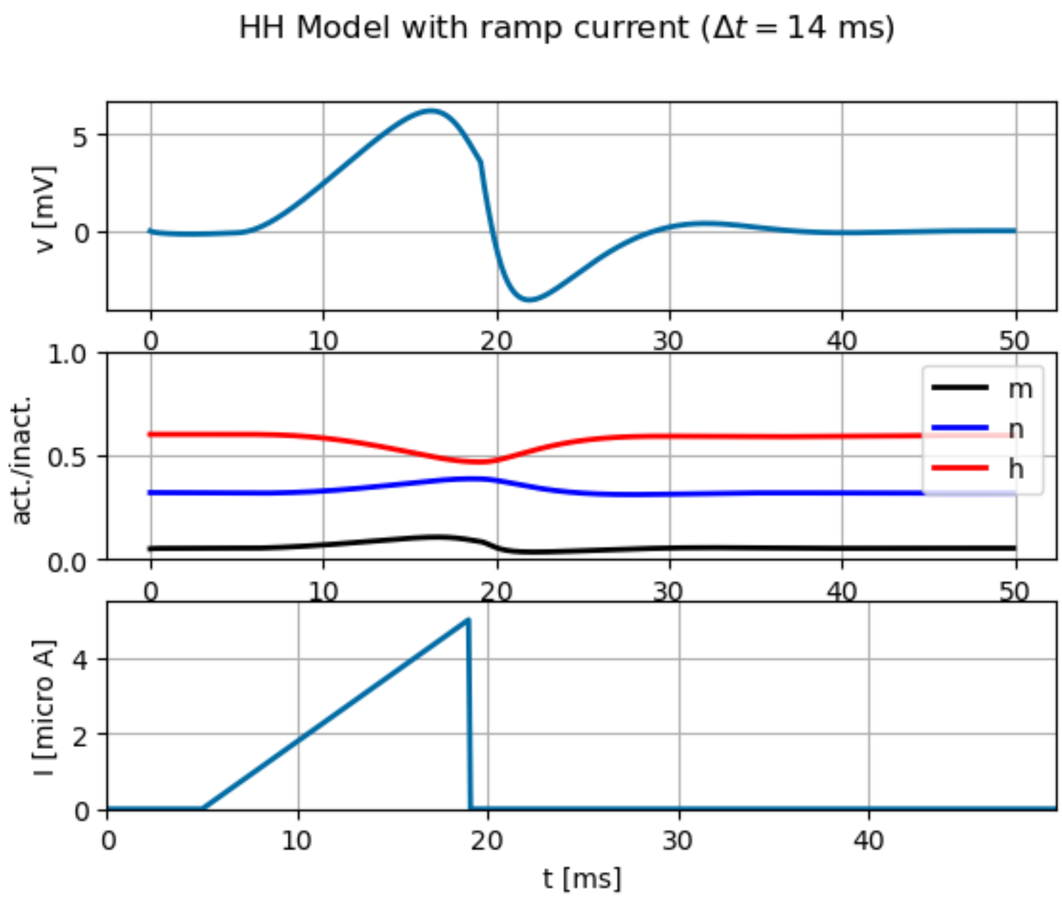


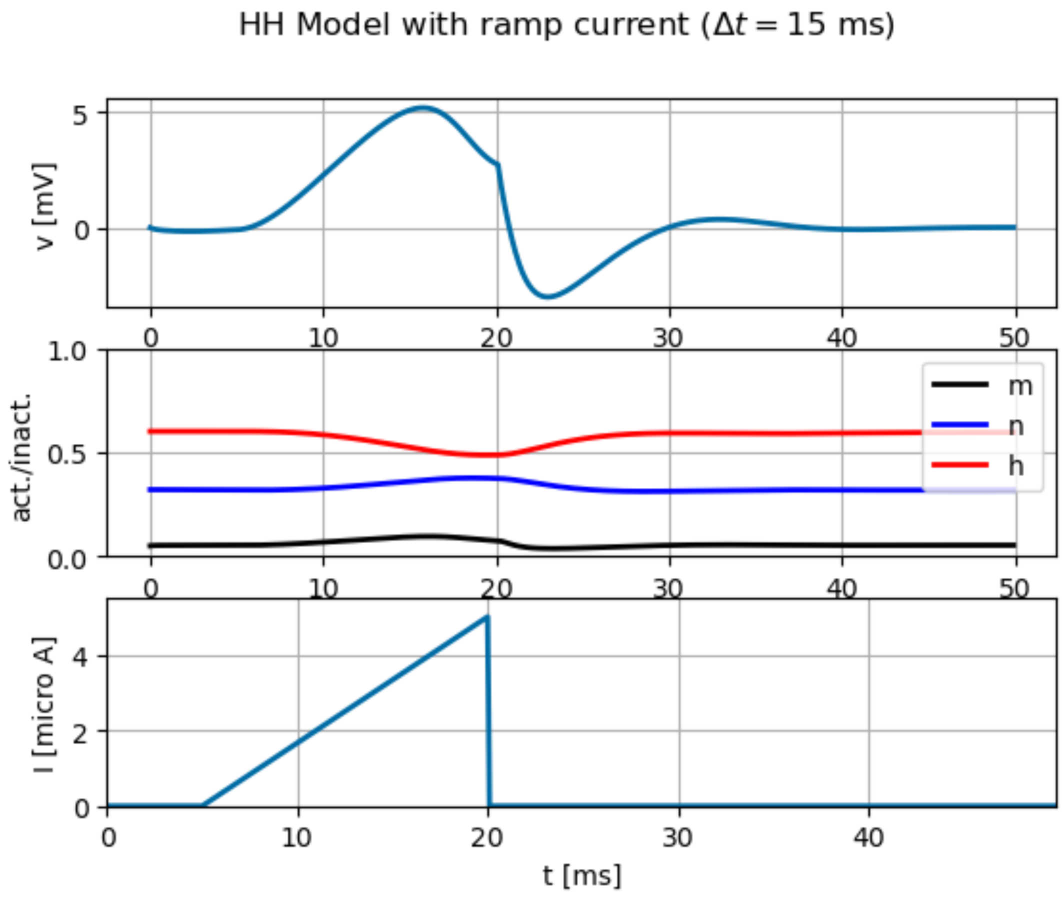

HH Model with ramp current ($\Delta t = 9$ ms)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]


HH Model with ramp current ($\Delta t = 11$ ms)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]


HH Model with ramp current ($\Delta t = 12$ ms)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]


HH Model with ramp current ($\Delta t = 13$ ms)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]





A2.2.2a conclusion

There is no spiking for a short ramp pulse (max amplitude of 5 μA) with a duration higher than 14 - 15 ms (slope: 0.36 - 0.33 mA/s)

In [17…

```
for duration, vm in vms:
    print('Membrane potential at t = t_end:', vm, ', duration
```

```
Membrane potential at t = t_end: -7.00579634 mV , duration: 30
ms
Membrane potential at t = t_end: 9.09742189 mV , duration: 50
ms
Membrane potential at t = t_end: -4.8078393 mV , duration: 55
ms
Membrane potential at t = t_end: 21.81022114 mV , duration: 60
ms
Membrane potential at t = t_end: 9.54408713 mV , duration: 65
ms
Membrane potential at t = t_end: 8.86590723 mV , duration: 70
ms
Membrane potential at t = t_end: 6.33381213 mV , duration: 90
ms
Membrane potential at t = t_end: 6.78747867 mV , duration: 100
ms
```

A2.2.2b conclusion

Δt **(ms)**	$V_m (t_{\text{end}})$ **(mV)**
5	9.32
7	19.1
9	87.5
11	92.2
12	95.1
13	18.8
14	3.71
15	2.77

A2.2.3 Differences

- Go back to Q2.2.3

A2.2.3 conclusion

In the slow ramp current, there is a slow injection of charge which gives the voltage gates the time to respond: i.e. the gating variabels h, n, m can respond to the input. Especially h and n since they have a slower time response (see 1.1). h is the inactivating parameter and will (partially) block (decreased with increasing voltage) the Na$^+$ influx, therby increasing the membrane potential threshold.

In this case we expect the membrane voltage threshold to be higher.

In the fast ramp current, there is a rapid injection of charge into the neuron. This gives the gating variables h, n not enough time to respond. The m parameter reacts very fast and will cause rapid Na$^+$ influx which will further depolarize the cell and provoke an action potential. We expect the membrane voltage to be lower.

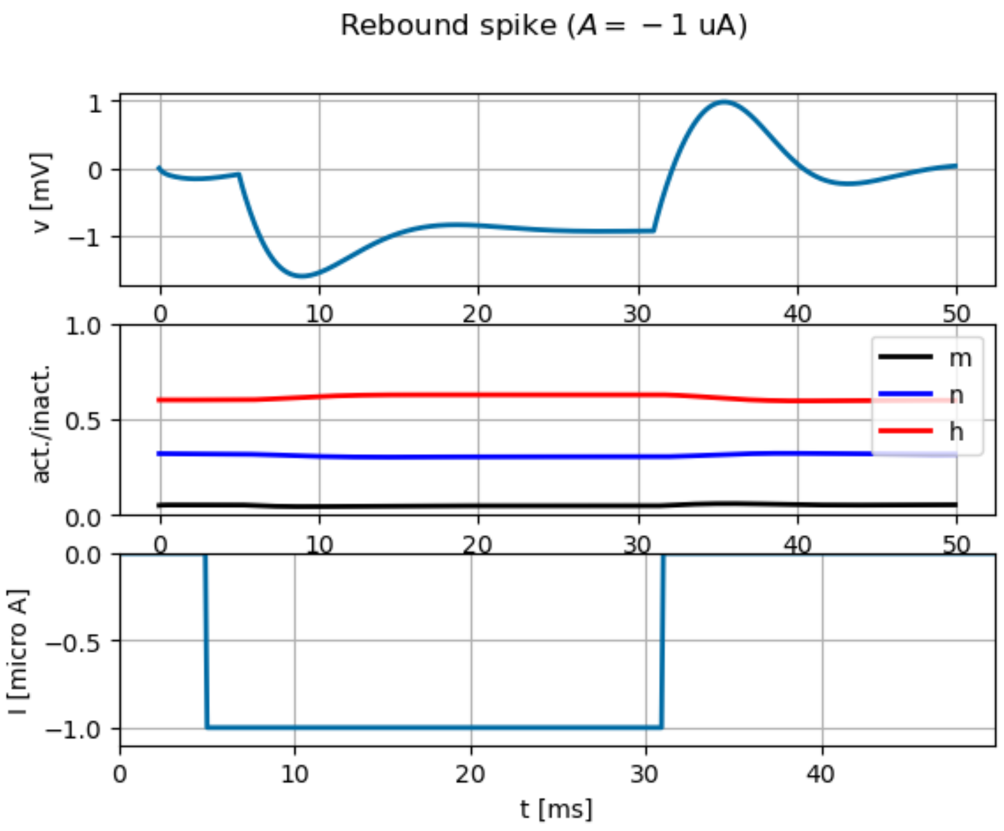
Our theoretical discussion matches the simulation findings: the threshold are 4 mV and 9 mV for the slow and fast ramp current respectively.

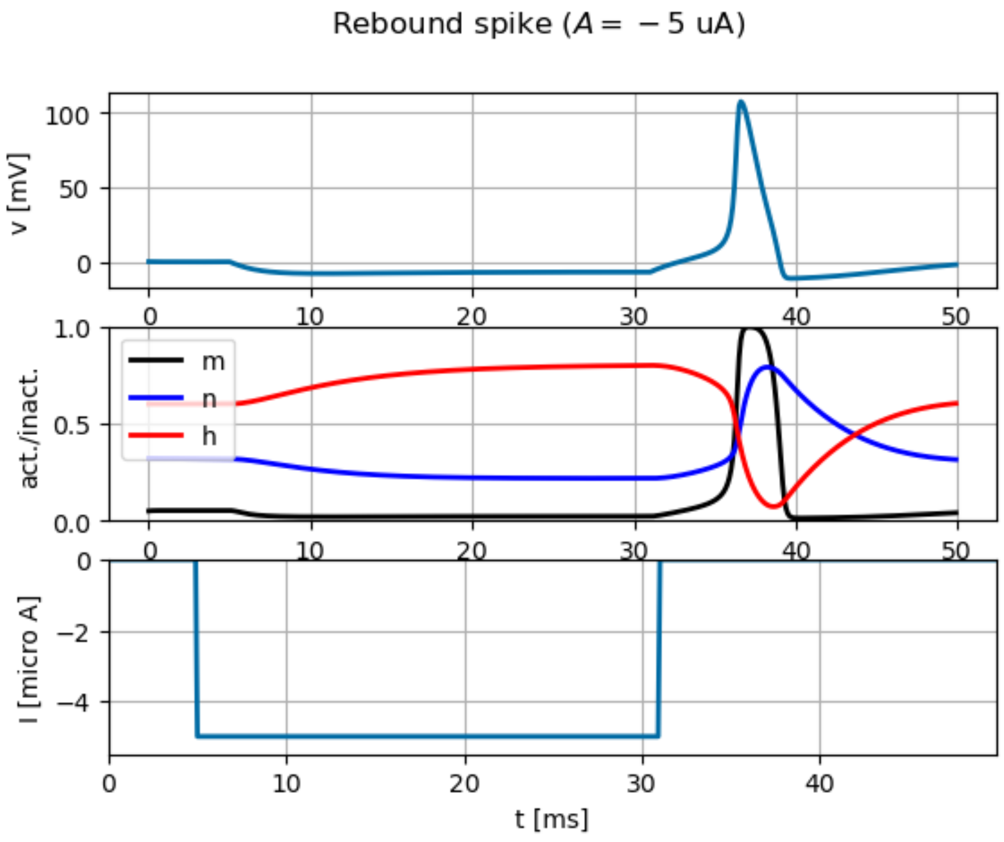
A2.3 Rebound spike

- Go back to Q2.3

In [18…
```
# hyperpolarizing step currents
Amp = [-1, -5]
t_start = 5
dt = 25
T_sim = 50
nm_ratio = []
h_par = []

step_currents = [input_factory.get_step_current(t_start, t_

# 2. Run the HH model for 300 ms
state_monitors_step_currents = [HH.simulate_HH_neuron(step_

# 3. Plot the results of the simulation
for i, state_monitor_step_current in enumerate(state_monito

    m, n, h = state_monitor_step_current.m[0], state_monito
    nm_ratio.append(n/m)
    h_par.append(h)
```

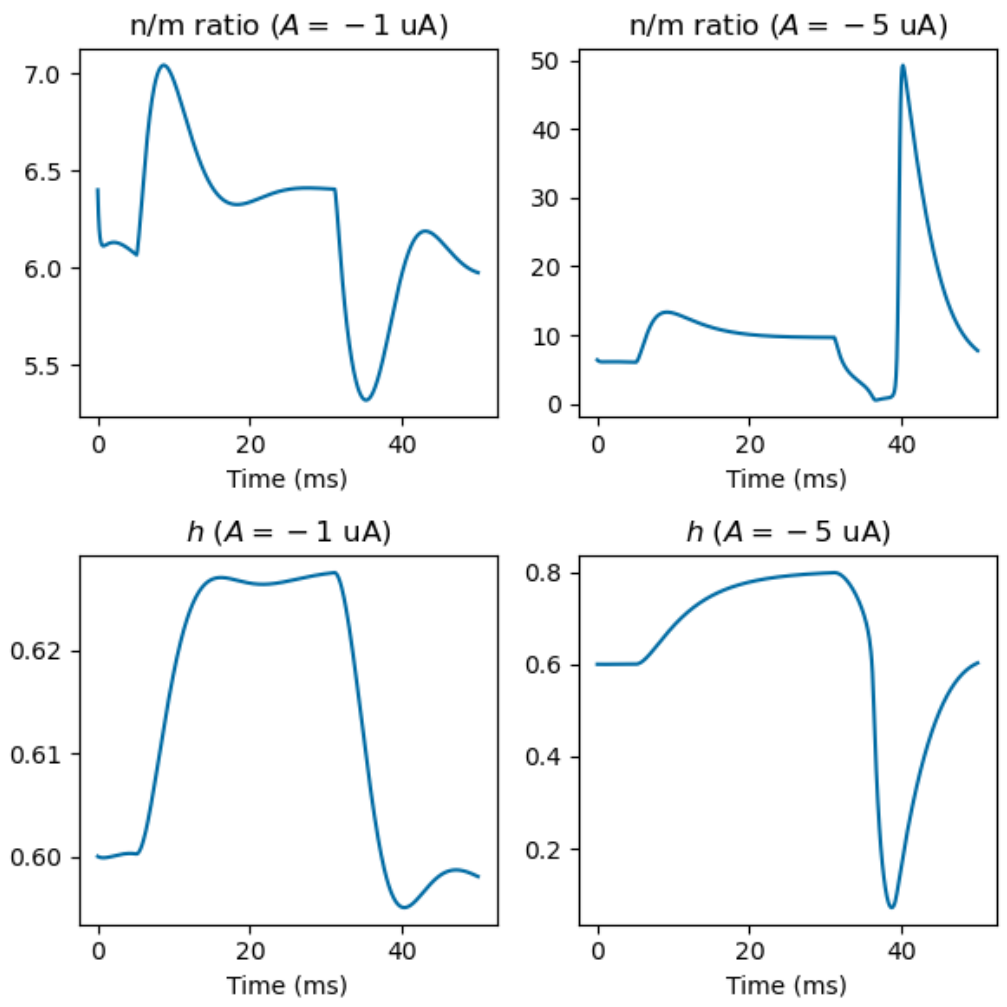
```
# plot the data
HH.plot_data(state_monitor_step_current, title = 'Rebou
```





In [19…
```
t = np.linspace(0, 50, len(h_par[0]))
```

```python
fig, (ax1, ax2) = plt.subplots(2, 2, figsize = (6, 6))


ax1[0].set_title('n/m ratio ($A='+str(Amp[0])+'$ uA)')
ax1[1].set_title('n/m ratio ($A='+str(Amp[1])+'$ uA)')
ax1[0].plot(t, nm_ratio[0])
ax1[1].plot(t, nm_ratio[1])

ax2[0].set_title('$h$ ($A='+str(Amp[0])+'$ uA)')
ax2[1].set_title('$h$ ($A='+str(Amp[1])+'$ uA)')
ax2[0].plot(t, h_par[0])
ax2[1].plot(t, h_par[1])

ax1[0].set_xlabel('Time (ms)')
ax1[1].set_xlabel('Time (ms)')
ax2[0].set_xlabel('Time (ms)')
ax2[1].set_xlabel('Time (ms)')

plt.tight_layout()
plt.show()
```



A2.3 conclusion

1. **No spiking**

When the negative input current is switched on, n is more dominant than m thus there will be more K$^+$ outflux than Na$^+$ influx, causing further polarization of the membrane. When the current is switched off, there is more influx than Na$^+$ than K$^+$ outflux, but not enough Na$^+$ will enter to provoke an action potential. The membrane potential will evolve towards the resting potential again.

2. **Spiking**
   When there is a large negative current switched on, the outflux of K$^+$ will be larger than the influx of Na$^+$: $n > m$. While the current is being applied, $h \to 0.8$ and therby forcing the Na$^+$ channels to open more (because m remains low). When the current is switched off again (around 35 ms) Na$^+$ rushes into the neuron ($m \ggg n$ because $\tau_m < \tau_n$), but around 40 ms the Na$^+$ channels quickly become inactivating ($h \to 0$), thereby allowing no further influx of Na$^+$. However, n becomes large which indicates that K$^+$ can easily flow out. An action potential has been generated!

From this you can see that the general generation of action potentials is comman: it is due to the (delay) opening and closing of Na$^+$ and K$^+$ channels. ^{1}note that the negtive current makes the Na$^+$ less inactivating, i.e. m increases.

```
In [ ]:
```