Lab Session #1

Computational Neurophysiology [E010620A]

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

Jorne Laton, Lloyd Plumart, Talis Vertriest, Jeroen Van Schependom, Sarah Verhulst

Student names and IDs: Robbe De Beck [01902805] & Robbe De Muynck [01908861]
Academic Year: 2022-2023

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.

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: 2 weeks after lecture. 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 (https://neuronaldynamics.epfl.ch/online/Ch2.S2.html)](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` : 20 μA between 1 and 2 ms.

- Fill in answers here

Q1.3. Plot V(t)

Plot the first 20 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 20 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 20 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 20 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/ (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/ (https://briansimulator.org/)](https://briansimulator.org/).

- [Import these modules]()

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 [1]:
```
# 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 [2]:
```python
# 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

##################################
##   A1.1a solutions 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):
    beta_n = -0.002*(u-25)/(1-np.exp((u-25)/9))
    alpha_n = 0.02*(u-25)/(1-np.exp(-(u-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)
```

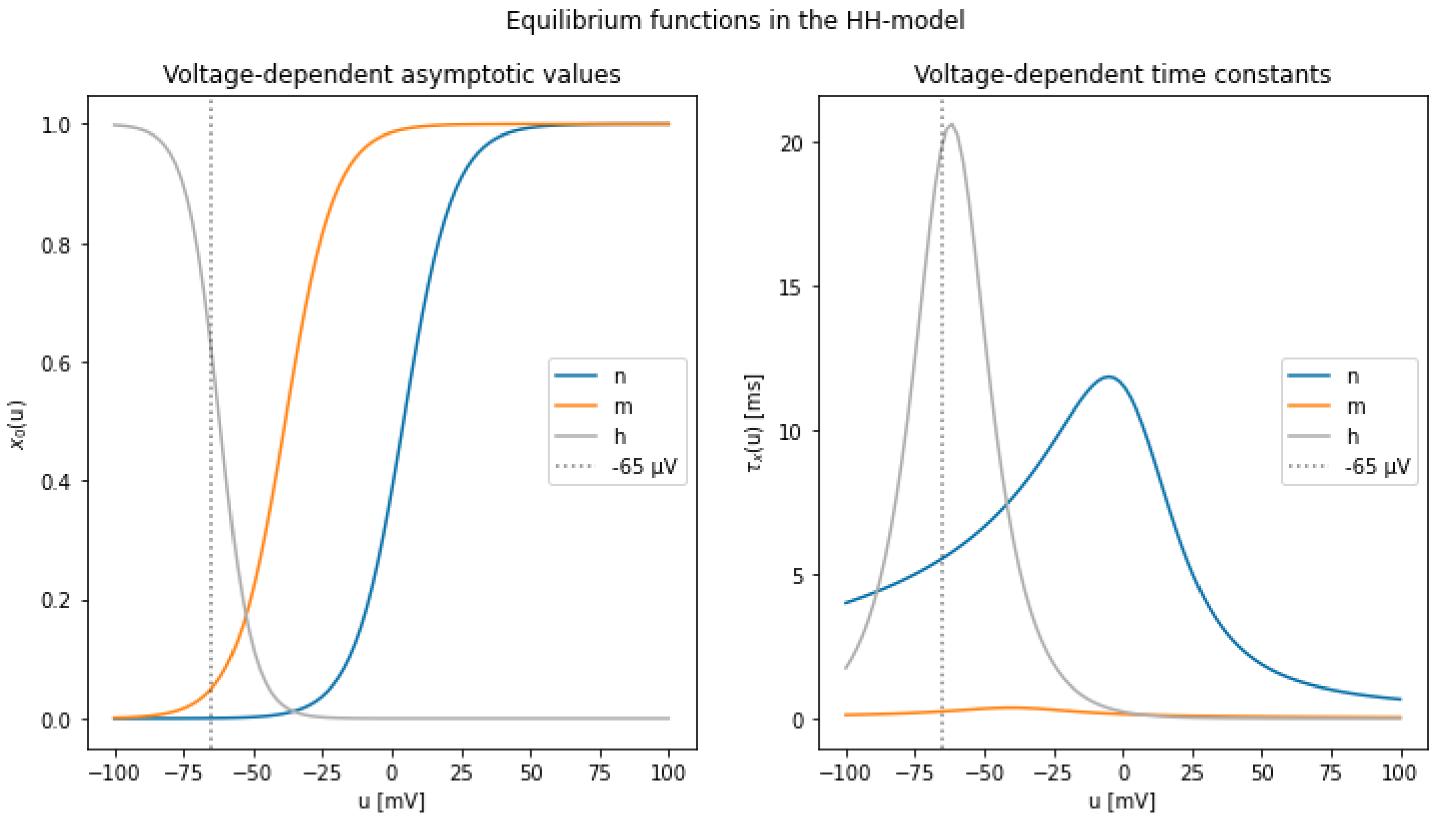
```python
    alpha_h= 0.25*np.exp(-(u+90)/12)
    return alpha_h, beta_h

def x_0(gating_variable):
    alpha_x, beta_x = gating_variable
    return alpha_x/(alpha_x + beta_x)

def tau_x(gating_variable):
    alpha_x, beta_x = gating_variable
    return 1/(alpha_x + beta_x)

def HH_model(T, I_input, dt):
    ### set conductances & reversal potentials
    g_Na_0 = 40 # mS/cm²
    E_Na = 55   # mV
    g_K_0 = 35
    E_K = -77
    g_L_0 = 0.3
    E_L = -65
    C_m = 1     # µF/cm²

    ### construct simulation time array
    T_n = int(np.ceil(T/dt))
    t = np.arange(0, T_n)*dt # ms

    ### initialize output arrays
    V = np.zeros(T_n)
    I_Na = np.zeros(T_n)
    I_K = np.zeros(T_n)
    I_L = np.zeros(T_n)

    alpha_m, beta_m = np.zeros(T_n), np.zeros(T_n)
    alpha_h, beta_h = np.zeros(T_n), np.zeros(T_n)
    alpha_n, beta_n = np.zeros(T_n), np.zeros(T_n)

    m = np.zeros(T_n)
    h = np.zeros(T_n)
    n = np.zeros(T_n)

    # neuronal resting potential
    V[0]= -65   # mV

```

```
    # asymptotic target values for gating variables
    alpha_m[0], beta_m[0] = gating_variable_m(V[0])
    alpha_h[0], beta_h[0] = gating_variable_h(V[0])
    alpha_n[0], beta_n[0] = gating_variable_n(V[0])
    m[0]= x_0((alpha_m[0], beta_m[0]))
    h[0]= x_0((alpha_h[0], beta_h[0]))
    n[0]= x_0((alpha_n[0], beta_n[0]))
    print(f"Initial gating parameters:\nm_0: {m[0]:.4f}\nh_0: {h[0]:.4f}\nn_0: {n[0]:.4f}")

    # ionic currents
    I_Na[0] = g_Na_0 * m[0]**3 * h[0] * (V[0]-E_Na)
    I_K[0] = g_K_0 * n[0]**4 * (V[0]-E_K)
    I_L[0] = g_L_0 * (V[0]-E_L)

    ### run update loop
    for i in range(0, T_n-1):
        # voltage & gating variables for NEXT timepoint
        # using CURRENT alpha, beta & currents
        V[i+1] = V[i] + dt/C_m*(-I_Na[i] - I_K[i] - I_L[i] + I_input[i])

        m[i+1] = m[i] + dt*(alpha_m[i]*(1-m[i]) - beta_m[i]*m[i])
        h[i+1] = h[i] + dt*(alpha_h[i]*(1-h[i]) - beta_h[i]*h[i])
        n[i+1] = n[i] + dt*(alpha_n[i]*(1-n[i]) - beta_n[i]*n[i])

        # calculate NEXT alpha, beta & currents
        alpha_m[i+1], beta_m[i+1] = gating_variable_m(V[i+1])
        alpha_h[i+1], beta_h[i+1] = gating_variable_h(V[i+1])
        alpha_n[i+1], beta_n[i+1] = gating_variable_n(V[i+1])

        I_Na[i+1] = g_Na_0 * m[i+1]**3 * h[i+1] * (V[i+1]-E_Na)
        I_K[i+1] = g_K_0 * n[i+1]**4 * (V[i+1]-E_K)
        I_L[i+1] = g_L_0 * (V[i+1]-E_L)

    ### cluster gating params
    gating_params = (alpha_m,  beta_m, alpha_h, beta_h, alpha_n, beta_n)

    return V, m, h, n, t, I_Na, I_K, I_L, gating_params
```

In [3]:

```python
# A1.1b Plot your graph below

# Hint: the first graph plots rate constants (n, m and h in function of membrane voltage)
# Hint: the second graph plots the voltage dependent time constants (tau_n, tau_m and tau_h in function of memb

# What is your conclusion?

##############################
##   A1.1b plots   ##
##############################

def nmh_plot(n, m, h):
    fig, axs = plt.subplots(1,2, figsize=(12,6))
    axs[0].plot(V, x_0(n),label='n')
    axs[0].plot(V, x_0(m),label='m')
    axs[0].plot(V, x_0(h),label='h')
    axs[0].axvline(x=-65, linestyle=':', color='k', alpha=0.5, label='-65 µV')
    axs[0].legend()
    axs[0].set_title('Voltage-dependent asymptotic values')
    axs[0].set_xlabel('u [mV]')
    axs[0].set_ylabel('$x_0$(u)')

    axs[1].plot(V, tau_x(n), label='n')
    axs[1].plot(V, tau_x(m), label='m')
    axs[1].plot(V, tau_x(h), label='h')
    axs[1].axvline(x=-65, linestyle=':', color='k', alpha=0.5, label='-65 µV')
    axs[1].legend(loc=5)
    axs[1].set_title('Voltage-dependent time constants')
    axs[1].set_xlabel('u [mV]')
    axs[1].set_ylabel(r'$\tau_x$(u) [ms]')
    plt.suptitle('Equilibrium functions in the HH-model')
    plt.show()

V = np.linspace(-100,100,100)
nmh_plot(gating_variable_n(V), gating_variable_m(V), gating_variable_h(V))

#########################
##   A1.1b conclusion   ##
#########################

```

```
# Answer in green box below
```



A1.1b conclusion

In the above plots, the characteristics of the parameters of the HH-model are visualised: m, n and h (which represent the channel kinetics) are gating variables to model the probability that a given channel is open in time.

'h' is used to describe the inhibitory characteristics of the Na^+ channels, while 'm' is used to describe the excitatory characteristics of the Na^+ channels. 'n' is used to describe the 'openness' of the K^+ channels.

In the HH-model, the gating variable approaches an asymptotic target value x0 with a time constant τx: the asymptotic target value (left plot) and the time constant are voltage-dependent (right plot). This voltage-dependency is visualised for each gating variable in the plot above. As a visual aid, the resting membrane potential of -65 μV is plotted as a verticle line.

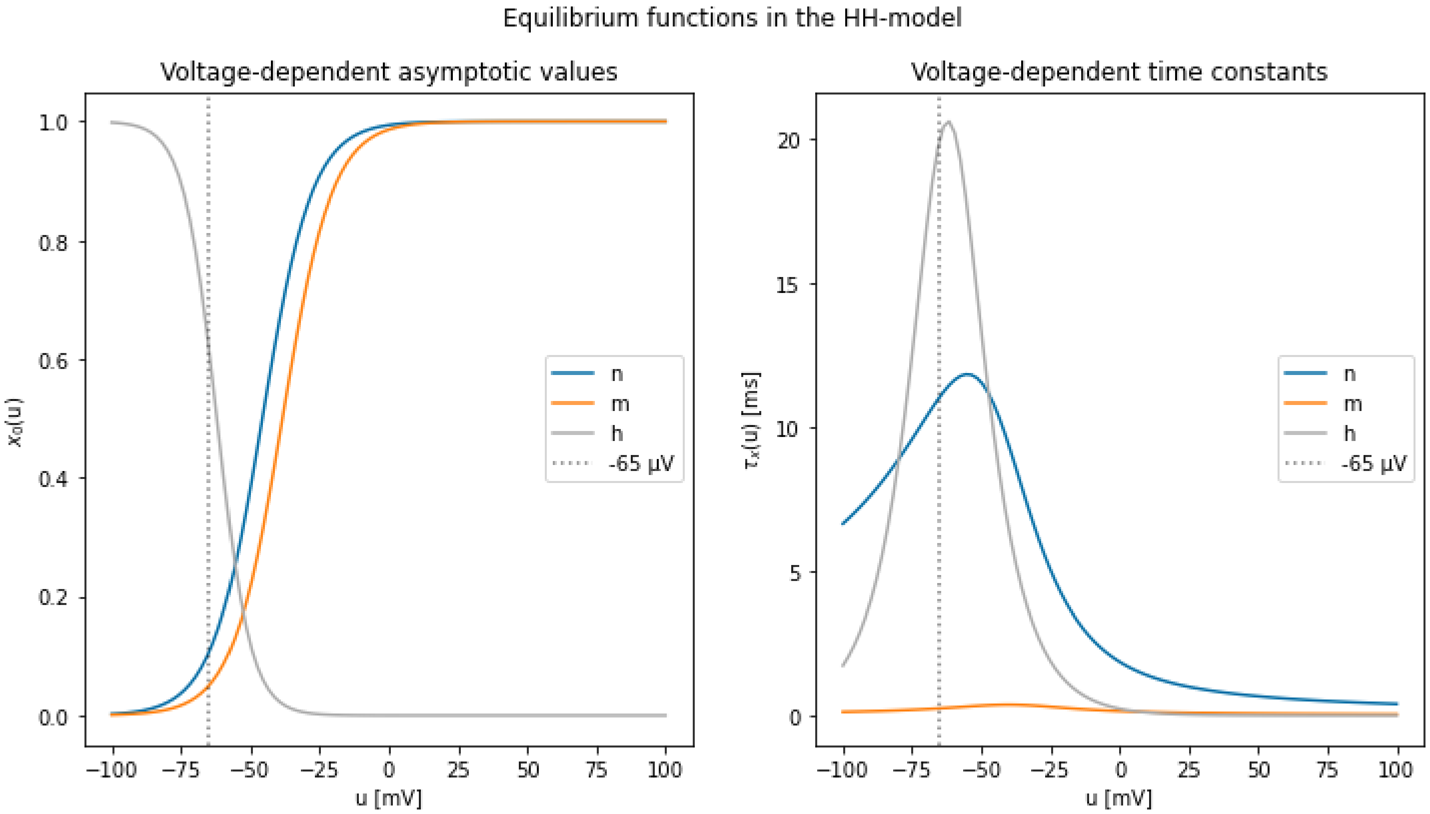
In [4]:

```python
# A1.1c Enter the updated HH model here

########################################
##   A1.1c updated HH model solution   ##
########################################

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

def HH_model_updated(T, I_input, dt):
    ### set conductances & reversal potentials
    g_Na_0 = 40 # mS/cm²
    E_Na = 55   # mV
    g_K_0 = 35
    E_K = -77
    g_L_0 = 0.3
    E_L = -65
    C_m = 1     # µF/cm²

    ### construct simulation time array
    T_n = int(np.ceil(T/dt))
    t = np.arange(0, T_n)*dt # [ms]

    ### initialize output arrays
    V = np.zeros(T_n)
    I_Na = np.zeros(T_n)
    I_K = np.zeros(T_n)
    I_L = np.zeros(T_n)

    alpha_m, beta_m = np.zeros(T_n), np.zeros(T_n)
    alpha_h, beta_h = np.zeros(T_n), np.zeros(T_n)
    alpha_n, beta_n = np.zeros(T_n), np.zeros(T_n)

    m = np.zeros(T_n)
    h = np.zeros(T_n)
    n = np.zeros(T_n)

    # neuronal resting potential
    V[0]= -65
```

```python

    # asymptotic target values for gating variables
    alpha_m[0], beta_m[0] = gating_variable_m(V[0])
    alpha_h[0], beta_h[0] = gating_variable_h(V[0])
    alpha_n[0], beta_n[0] = gating_variable_n_updated(V[0])
    m[0]= x_0((alpha_m[0], beta_m[0]))
    h[0]= x_0((alpha_h[0], beta_h[0]))
    n[0]= x_0((alpha_n[0], beta_n[0]))
    print(f"Initial gating parameters:\nm_0: {m[0]:.4f}\nh_0: {h[0]:.4f}\nn_0: {n[0]:.4f}")

    # ionic currents

    I_Na[0] = g_Na_0 * m[0]**3 * h[0] * (V[0]-E_Na)
    I_K[0] = g_K_0 * n[0]**4 * (V[0]-E_K)
    I_L[0] = g_L_0 * (V[0]-E_L)

    ### run update loop
    for i in range(0, T_n-1):
        # voltage & gating variables for NEXT timepoint
        # using CURRENT alpha, beta & currents
        V[i+1] = V[i] + dt/C_m*(-I_Na[i] - I_K[i] - I_L[i] + I_input[i])

        m[i+1] = m[i] + dt*(alpha_m[i]*(1-m[i]) - beta_m[i]*m[i])
        h[i+1] = h[i] + dt*(alpha_h[i]*(1-h[i]) - beta_h[i]*h[i])
        n[i+1] = n[i] + dt*(alpha_n[i]*(1-n[i]) - beta_n[i]*n[i])

        # calculate NEXT alpha, beta & currents
        alpha_m[i+1], beta_m[i+1] = gating_variable_m(V[i+1])
        alpha_h[i+1], beta_h[i+1] = gating_variable_h(V[i+1])
        alpha_n[i+1], beta_n[i+1] = gating_variable_n_updated(V[i+1])

        I_Na[i+1] = g_Na_0 * m[i+1]**3 * h[i+1] * (V[i+1]-E_Na)
        I_K[i+1] = g_K_0 * n[i+1]**4 * (V[i+1]-E_K)
        I_L[i+1] = g_L_0 * (V[i+1]-E_L)

    ### cluster gating params
    gating_params = (alpha_m,  beta_m, alpha_h, beta_h, alpha_n, beta_n)

    return V, m, h, n, t, I_Na, I_K, I_L, gating_params
```

In [5]:

```
# Plot with updated gating variable n
V = np.linspace(-100,100,100)
nmh_plot(gating_variable_n_updated(V), gating_variable_m(V), gating_variable_h(V))
```

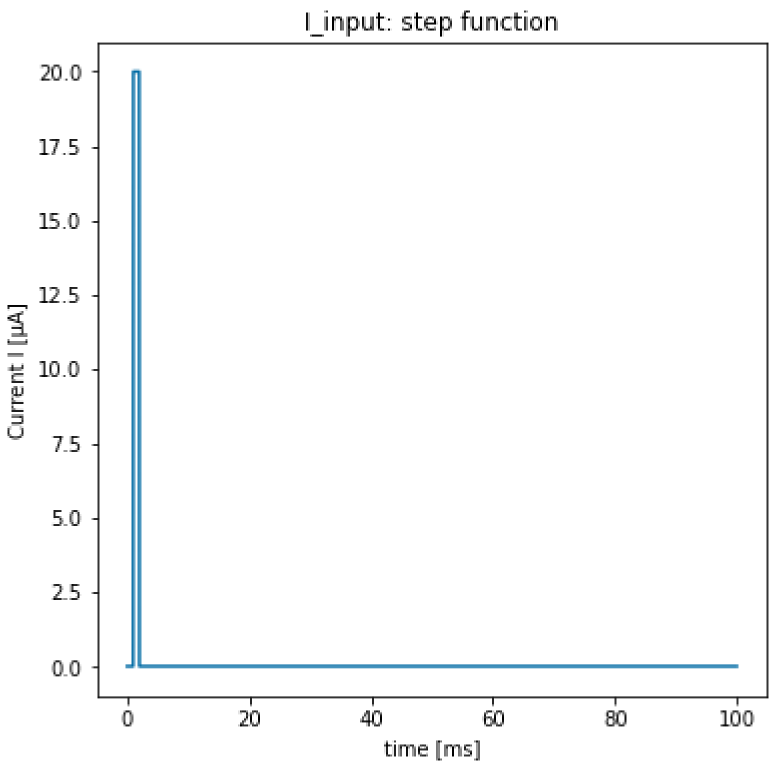


A1.2: Response simulation

- Go back to Q1.2

In [6]:

```
# A1.2a Set up and plot your input current

######################################
##   A1.2a solutions input current   ##
######################################

def I_step(T, dt, I, start_imp=1, stop_imp=2):
    """Returns a step current function [μA], based on design parameters"""
    # T = simulation time [ms]
    # dt = update time [ms]

    t = np.arange(0, np.ceil(T/dt))*dt # time array [ms]
    I_step = np.where((t >= start_imp)*(t < stop_imp), I, 0) # step current array [μA]

    return t, I_step

# Plot of impulse current
T, dt, I = 100, 0.01, 20
t, I_input = I_step(T, dt, I)

fig, ax = plt.subplots(1,1, figsize=(6,6))
ax.plot(t, I_input)
ax.set_title('I_input: step function')
ax.set_xlabel('time [ms]')
ax.set_ylabel('Current I [μA]')
plt.show()
```



In [7]:

```
# A1.2b Insert the input current into your HH_model function

#######################################
##   A1.2b insert input in HH_model   ##
#######################################
T, dt, I = 100, 0.01, 20
t, I_input = I_step(T, dt, I)

V, m, h, n, t, I_Na, I_K, I_L, gating_params = HH_model_updated(T, I_input, dt)
```

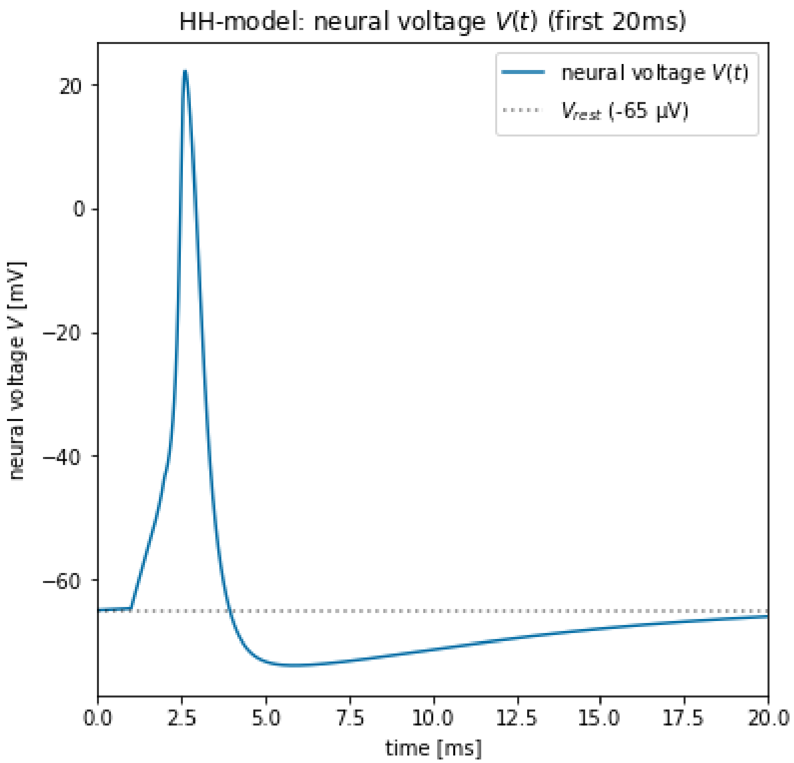
```
Initial gating parameters:
m_0: 0.0498
h_0: 0.6225
n_0: 0.1051
```

A1.3: Plot V(t)

- Go back to Q1.3

In [8]:

```
# A1.3a Enter your answer below

###########################
##   A1.3a solution plot  ##
###########################

fig, ax = plt.subplots(1,1, figsize=(6,6))

ax.plot(t, V, label='neural voltage $V(t)$')
ax.axhline(y=-65, linestyle=':', color='k', alpha=0.5, label='$V_{rest}$ (-65 µV)')

ax.set_title('HH-model: neural voltage $V(t)$ (first 20ms)')
ax.set_xlabel('time [ms]')
ax.set_ylabel('neural voltage $V$ [mV]')
ax.legend()
ax.set_xlim([0, 20])
plt.show()
```



In [9]:

```python
# A1.3b Enter your interpretation and conclusion below

#########################
##   A1.3b conclusion   ##
#########################

# Answer in green box below
```

A1.3b conclusion

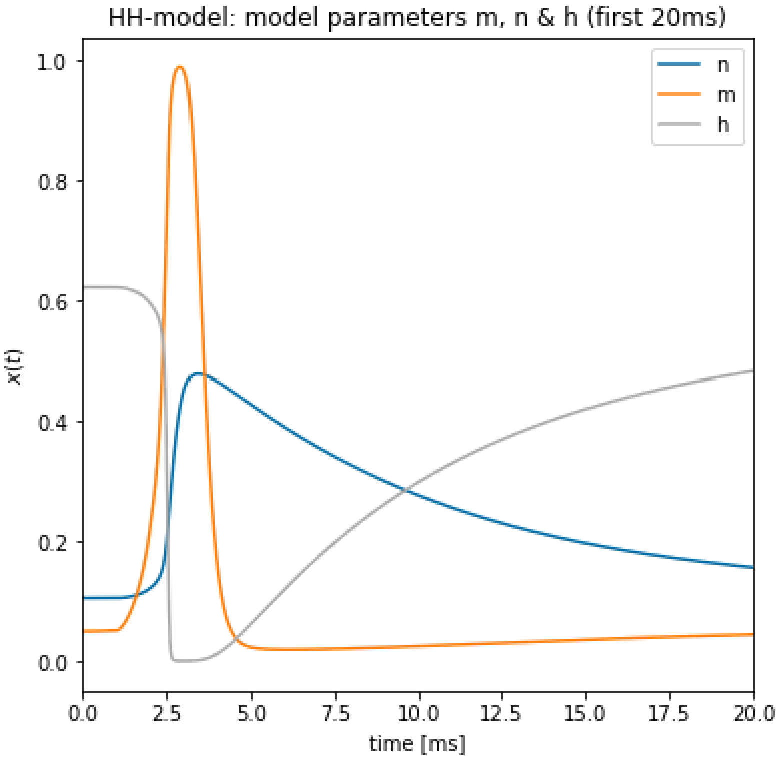
The HH-model is stimulated by a short but strong input current during the time interval between 1ms and 2ms: at first, a moderate increase in the membrane potential is seen until a certain treshold is reached. Subsequently, the membrane potential rapidly increases (depolarized) and an action potential is generated. From there on, the membrane experiences hyperpolarization (around 5ms) before returning to its initial resting potential (towards 20ms).

A1.4: Plot the model parameters

- Go back to Q1.4

In [10]:

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

############################
##   A1.4a solution plot   ##
############################

fig, ax = plt.subplots(1,1, figsize=(6,6))
ax.plot(t, n, label='n')
ax.plot(t, m, label='m')
ax.plot(t, h, label='h')

ax.set_title('HH-model: model parameters m, n & h (first 20ms)')
ax.set_xlabel('time [ms]')
ax.set_ylabel('$x(t)$')
ax.legend()
ax.set_xlim([0, 20])
plt.show()
```



In [11]:

```
# A1.4b Enter your interpretation and conclusion answer below

#########################
##   A1.4b conclusion   ##
#########################

# Answer in green box below
```

A1.4b conclusion

The above figure shows the dynamics of the gating variables and relate to how the action potential is mediated by Na^+ and K^+ channels. 'm' has a very fast response: Na^+ flows rapidly into the neuron, which causes the depolarization of the neuron.

Initially, 'h' inhibits the Na^+ transfer. If a certain membrane potential threshold is reached, 'h' drops, results in the loss of inhibitory function and an action potential is generated. After the peak is reached, 'h' slowly rises again (due to a very long time constant) and starts to inhibit the Na^+ inflow, which results in slowly repolarization of the neuron.

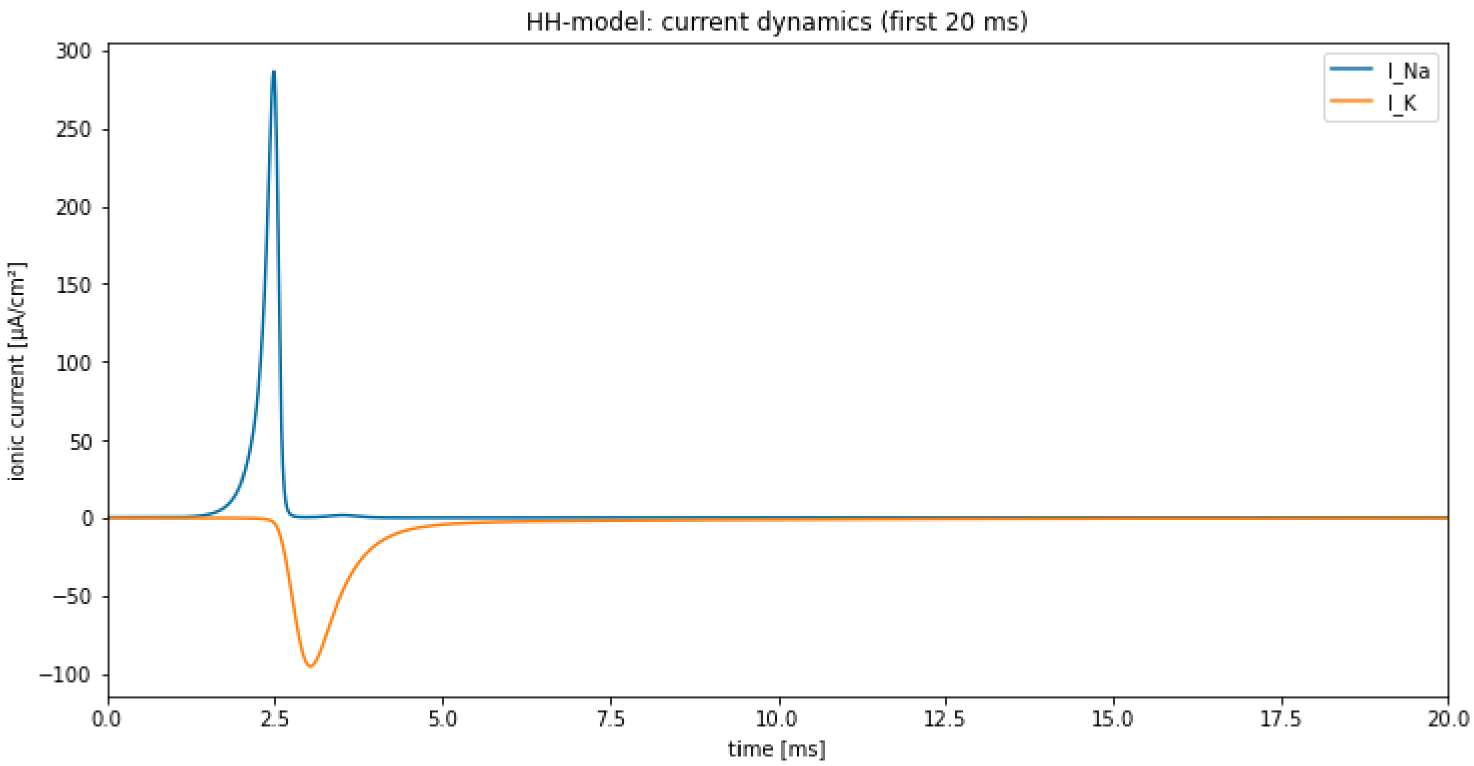
'n' has an intermediate time constant: right after the action potential, the K^+ channels open, which causes K^+ outflow and therefore hyperpolarization of the neuronal membrane potential.

A1.5: Plot I_Na and I_K

- Go back to Q1.5

In [12]:

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

############################
##   A1.5a plot   ##
############################

fig, ax = plt.subplots(1,1, figsize=(12,6))

ax.plot(t, -I_Na, label='I_Na')
ax.plot(t, -I_K, label='I_K')

ax.set_title('HH-model: current dynamics (first 20 ms)')
ax.set_xlabel('time [ms]')
ax.set_ylabel('ionic current [µA/cm²]')
ax.legend()
ax.set_xlim([0, 20])
plt.show()
```



In [13]:

```
# A1.5b Enter your interpretation and conclusion

########################
##   A1.5b conclusion   ##
########################

# Answer in green box below
```

A1.5b conclusion

The sodium current I_Na (Na^+ inflow) depends on the variables m and h: it has a sharp positive peak during the upswing of an action potential. This is due to the fast inflow of Na^+ ions into the neuronal cell.

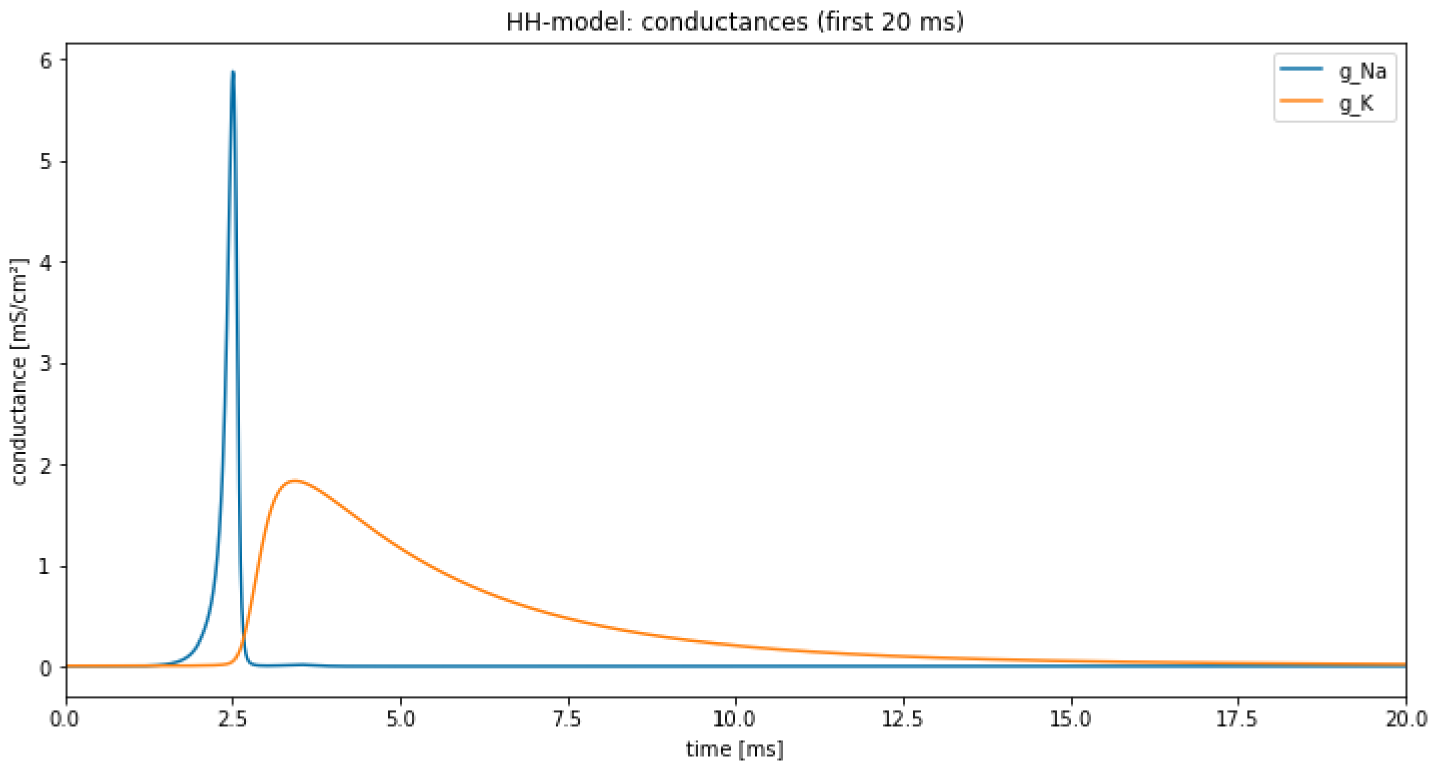
The potassium current I_K (K^+ outflow) is controlled by the variable n: it starts with a delay compared to I_Na current and has a negative peak due to the outflow of K^+ ions. The K^+ profile is more 'smeared out' with respect to the Na^+ peak, resulting in the slower K^+ outflow mentioned before. Towards the end of the simulation, both currents return to the 0 μA base value due to the closing of the ion channels.

A1.6: Plot the conductances g_Na and g_K

- Go back to Q1.6

In [14]:

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

###########################
##   A1.6a solution plot  ##
###########################

g_Na_0 = 40 # mS/cm²
g_K_0 = 35

g_Na = g_Na_0 * m**3 * h
g_K = g_K_0 * n**4

fig, ax = plt.subplots(1,1, figsize=(12,6))

ax.plot(t, g_Na, label='g_Na')
ax.plot(t, g_K, label='g_K')

ax.set_title('HH-model: conductances (first 20 ms)')
ax.set_xlabel('time [ms]')
ax.set_ylabel('conductance [mS/cm²]')
ax.legend()
ax.set_xlim([0, 20])
plt.show()
```



In [15]:

```
# A1.6b Enter your interpretation and conclusion

########################
##   A1.6b conclusion   ##
########################

# Answer in green box below
```

A1.6b conclusion

All ion channels can be characterized by their conductance. This conductance is voltage- and time-dependent (due to the voltage-dependency of m, h and n, which are used to model the ion channel conductances). If the ion channels are open (Na^+ or K^+), they transmit current with a maximum conductance, as can be seen in the plot above. During the spike onset, a high Na^+ conductance is found (depolarization) and afterwards, the K^+ conductance has a peak (hyper- and re-polarization).

Part 2: Package BRIAN

Import

In [16]:

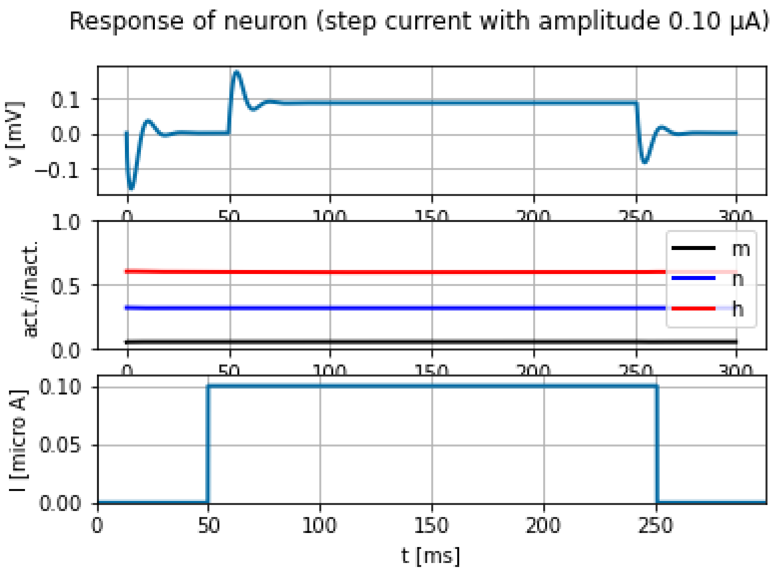
```
# The new libraries we need to add

%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

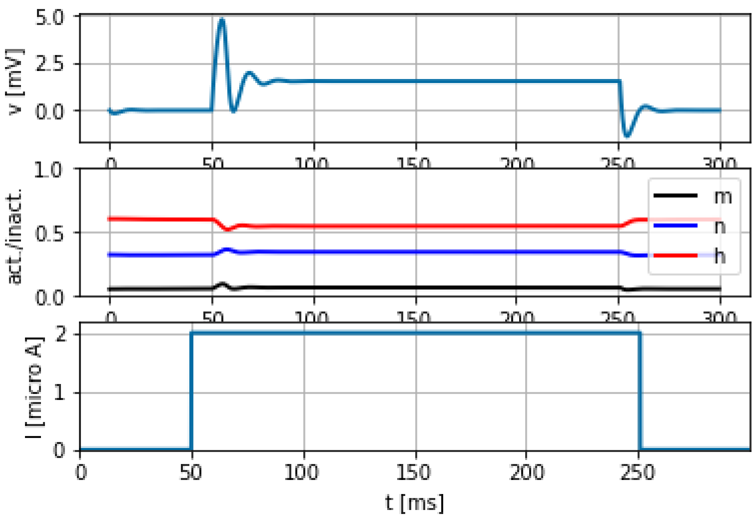
- Go back to Q2.1

In [27]:

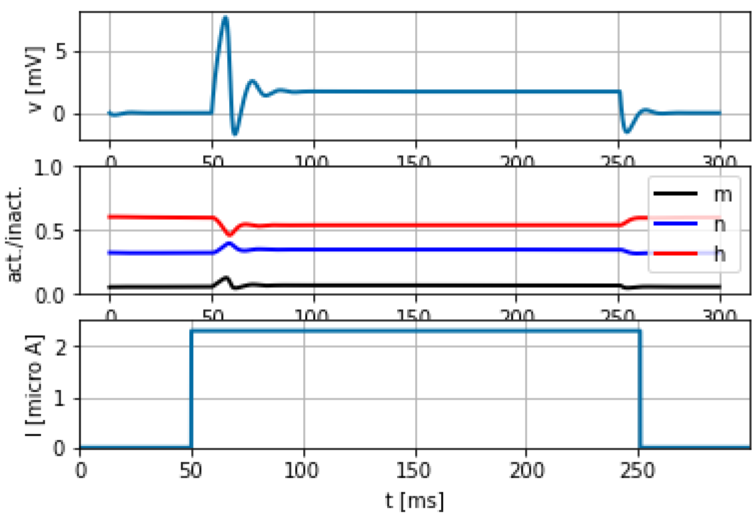
```python
# A2.1a Enter your answer below

# Hint: The unit of the I_input current in the neurodynex3.hodgkin_huxley module is μA (coded b2.uA)

########################
##   Q2.1. solution   ##
########################
# N = 27
# current_amplitudes = np.linspace(0.1, 50, N)
current_amplitudes = [0.1, 2.0, 2.3, 2.4, 5.0, 6.0, 6.1, 6.2, 6.3, 6.4, 10.0, 20.0, 30.0, 40.0, 50.0] #μA

for i in current_amplitudes:
#     print('input current: ', i)
    I_input = input_factory.get_step_current(50,250,b2.ms, i*b2.uA) # μA
    State_monitor = HH.simulate_HH_neuron(I_input, 300*b2.ms)
    HH.plot_data(State_monitor, title='Response of neuron (step current with amplitude {:.2f} μA)'.format(i))
    plt.show()
```



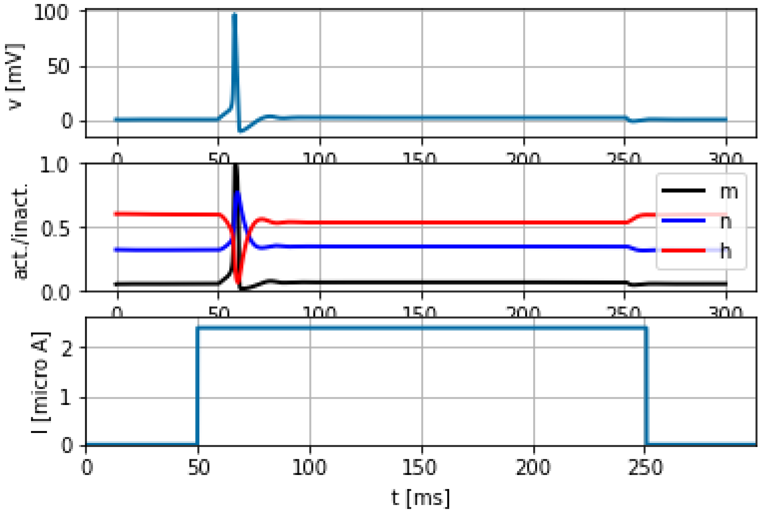
Response of neuron (step current with amplitude 2.00 µA)



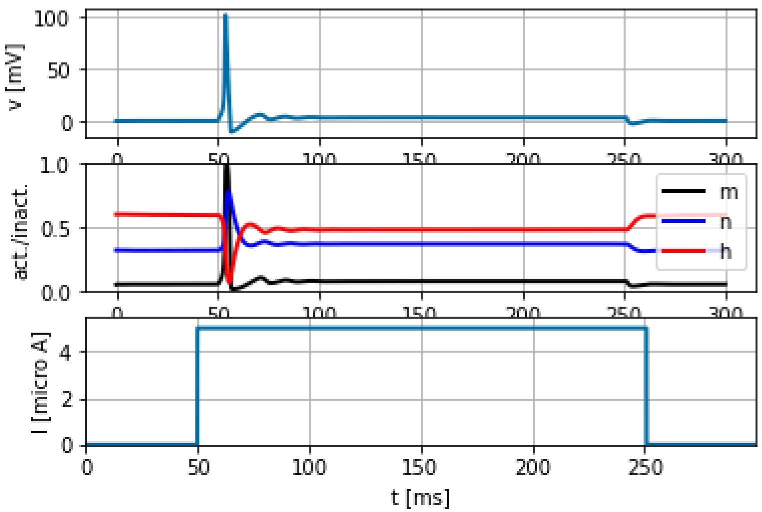
Response of neuron (step current with amplitude 2.30 µA)



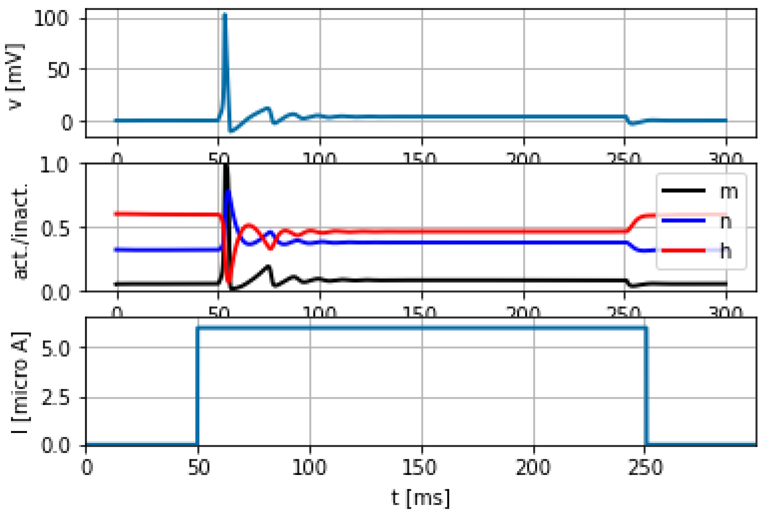
Response of neuron (step current with amplitude 2.40 µA)



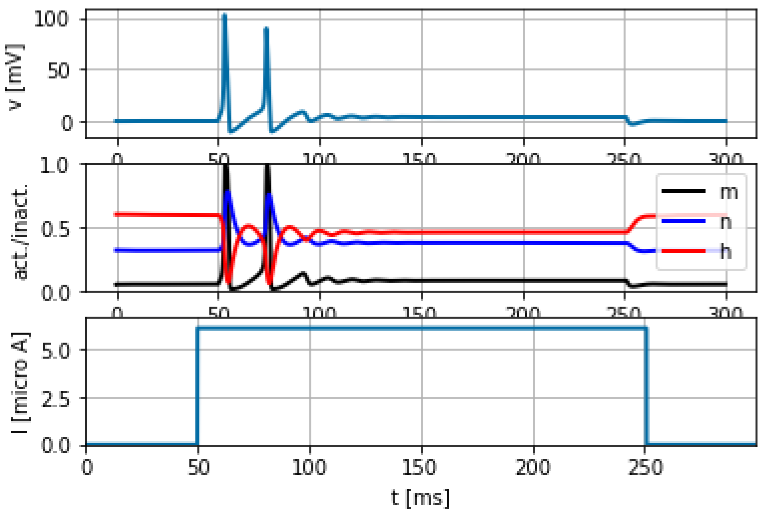
Response of neuron (step current with amplitude 5.00 µA)

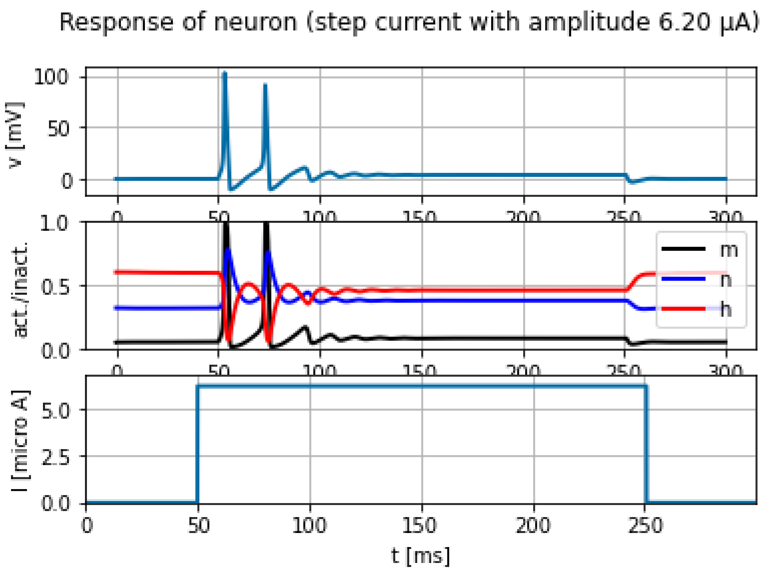


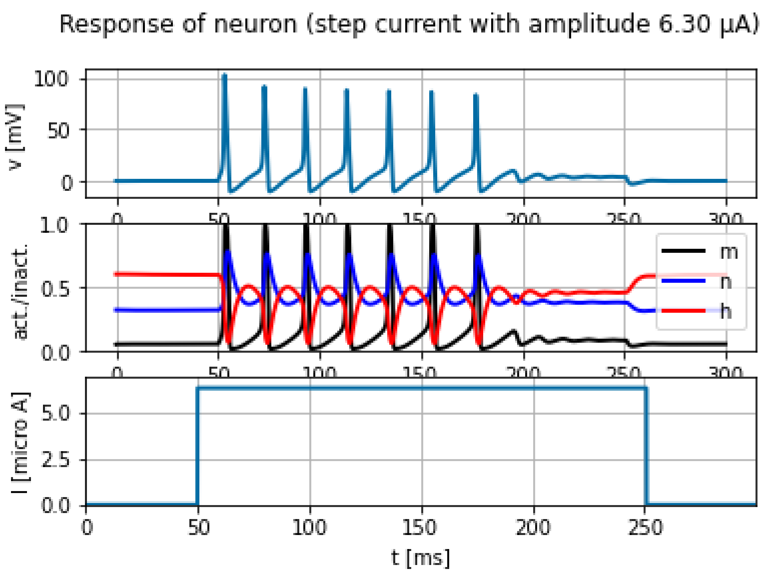
Response of neuron (step current with amplitude 6.00 μA)



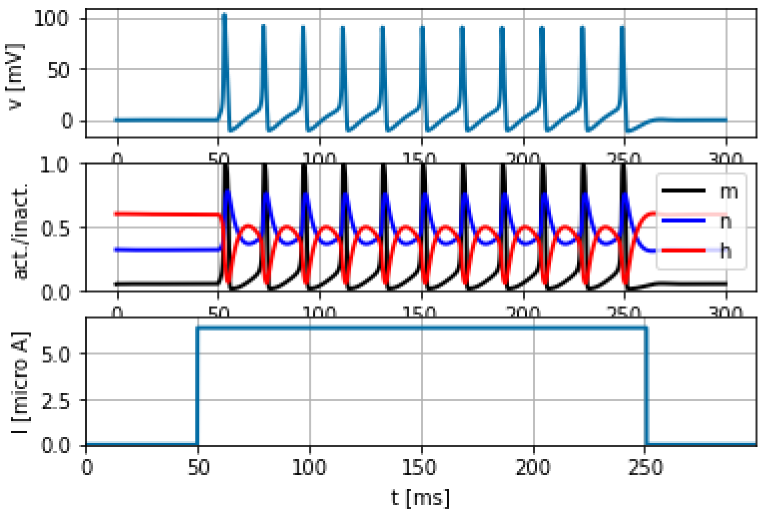
Response of neuron (step current with amplitude 6.10 μA)



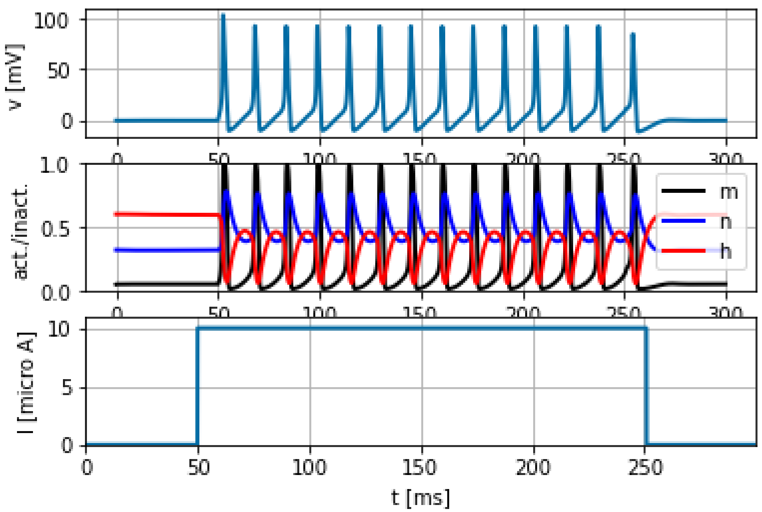

Response of neuron (step current with amplitude 6.20 µA)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

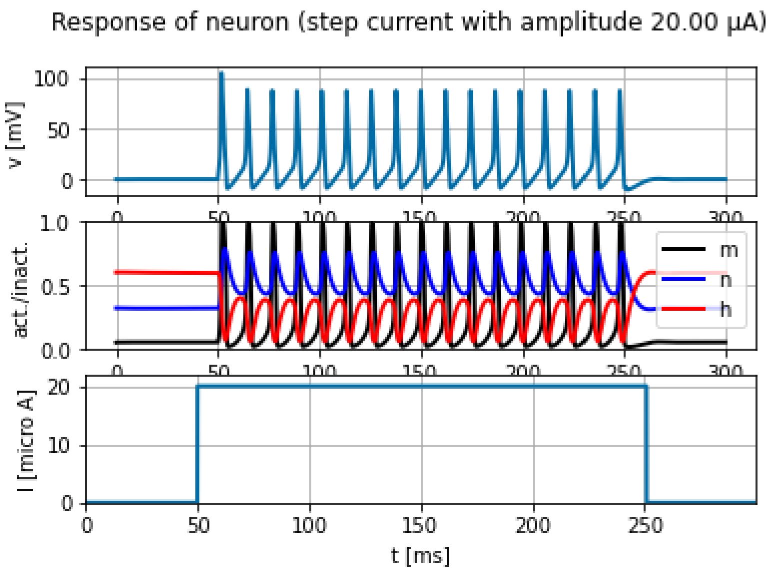

Response of neuron (step current with amplitude 6.30 µA)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

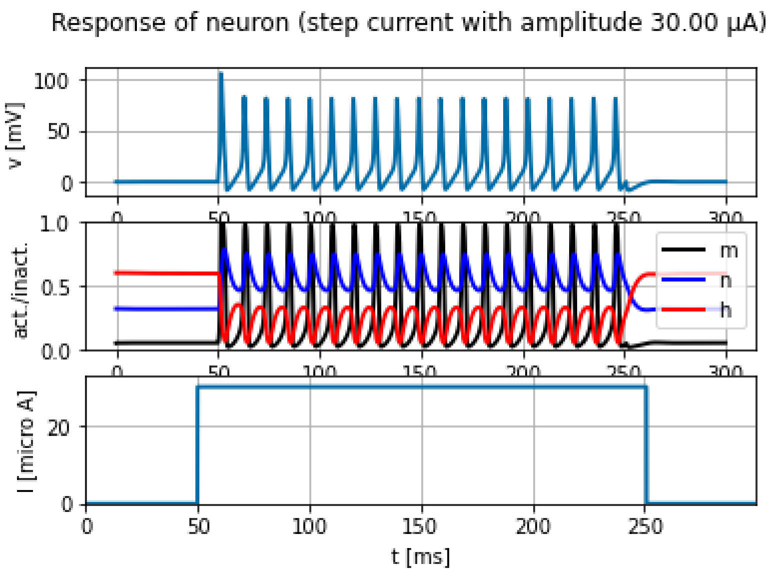
Response of neuron (step current with amplitude 6.40 µA)



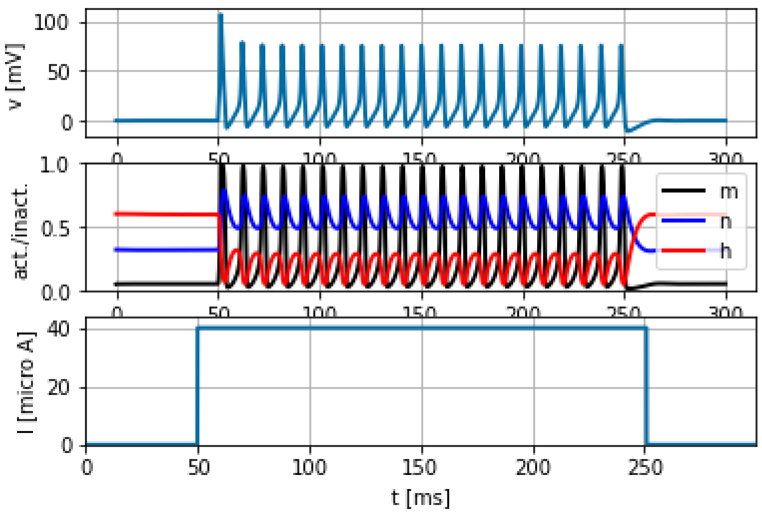
Response of neuron (step current with amplitude 10.00 µA)



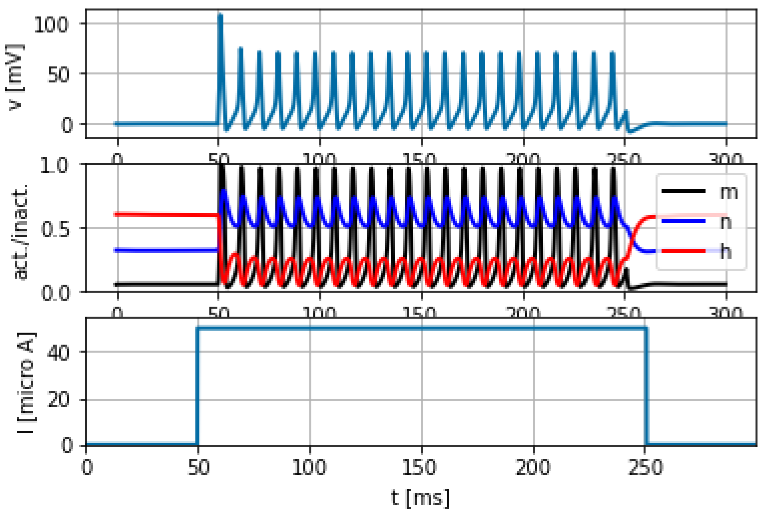

Response of neuron (step current with amplitude 20.00 μA)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]


Response of neuron (step current with amplitude 30.00 μA)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]

Response of neuron (step current with amplitude 40.00 μA)



Response of neuron (step current with amplitude 50.00 μA)



In [18]:

```
# Enter your conclusion and interpretation

##########################
##   Q2.1. conclusion   ##
##########################


# Answer in green box below
```

A2.1 conclusion

The lowest step current amplitude to generate a spike corresponds to an input current between 2.3 μA and 2.4 μA. Somewhere between these values the neuronal threshold potential is reached.

From input currents starting from 6.1 μA a second spike is generated. Somewhere between an input current of 6.2 μA and 6.3 μA repetitive firing occurs.

Further increase of the input current results in repetitive firing with a higer frequency. Due to the hyperpolarisation of the neuron, a higher input current amplitude is needed to reach the threshold. If the input current is increased, this threshold is reached faster, which results in a lower refractory period and thus a higher firing frequency.

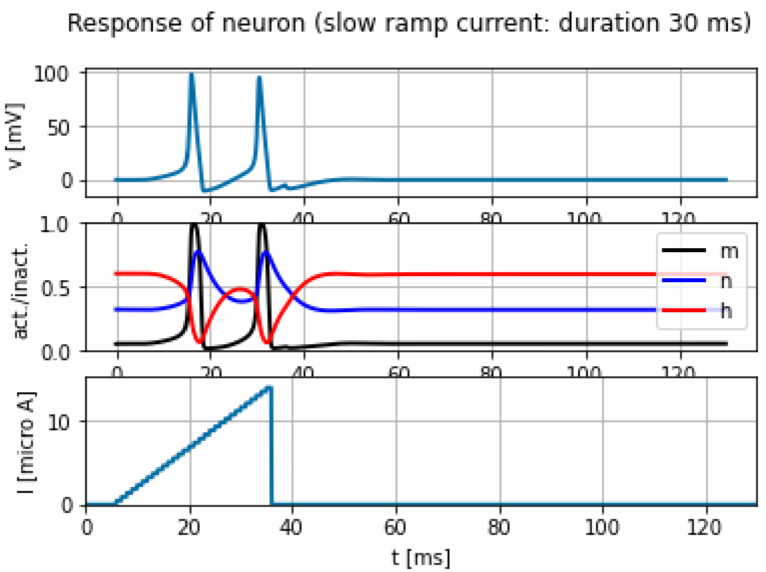
A2.2.1 Slow ramp current

- Go back to Q2.2.1

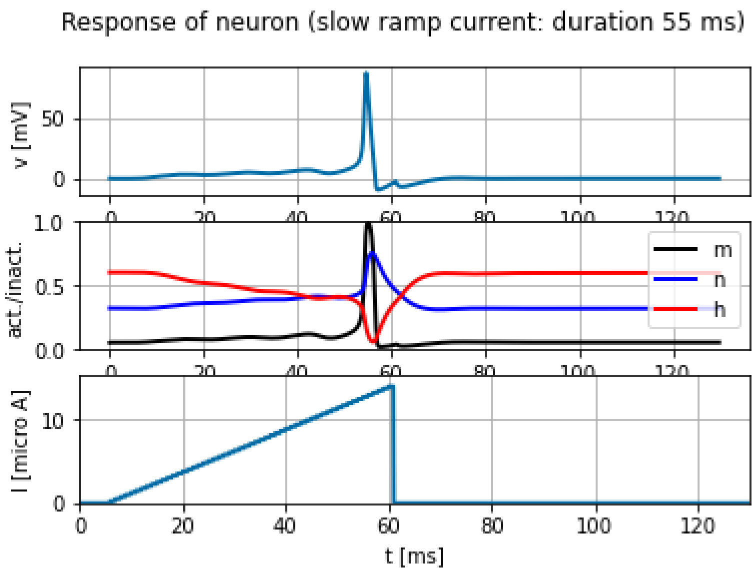
In [19]:

```
# Enter your code below

#########################
##   Q2.2.1a solution   ##
#########################
N = 25
# current_durations = np.linspace(35, 105, N)
current_durations = [35, 60, 75, 80, 90, 105]
duration_total = 130
for t_end in current_durations:
    I_input = input_factory.get_ramp_current(5, int(t_end), b2.ms, 0 * b2.uA, 14*b2.uA) # μA
    State_monitor = HH.simulate_HH_neuron(I_input, duration_total*b2.ms)
    sampling_time = duration_total/len(State_monitor.vm[0])
    print('Membrane voltage at time when the current injection stops: {:.2f} mV'.format(State_monitor.vm[0][in
    HH.plot_data(State_monitor, title='Response of neuron (slow ramp current: duration {} ms)'.format(t_end - !
    plt.show()
```

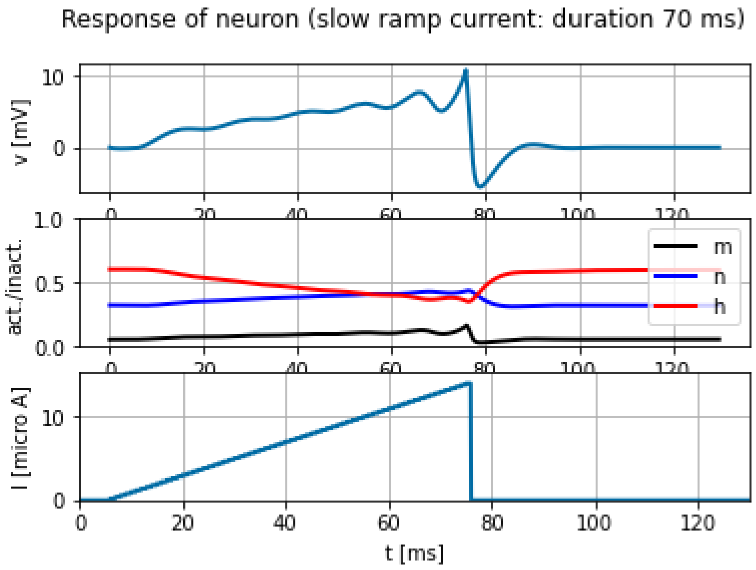
```
Membrane voltage at time when the current injection stops: -7.01 mV
```



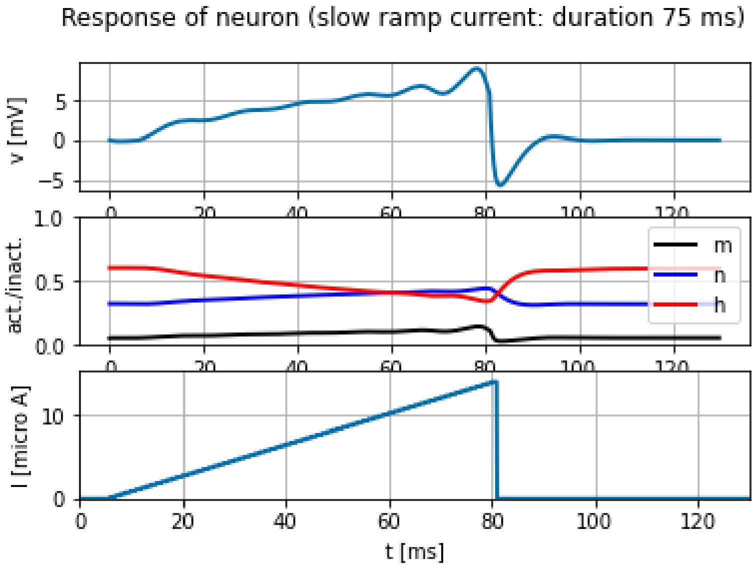

```
Membrane voltage at time when the current injection stops: -4.81 mV
```



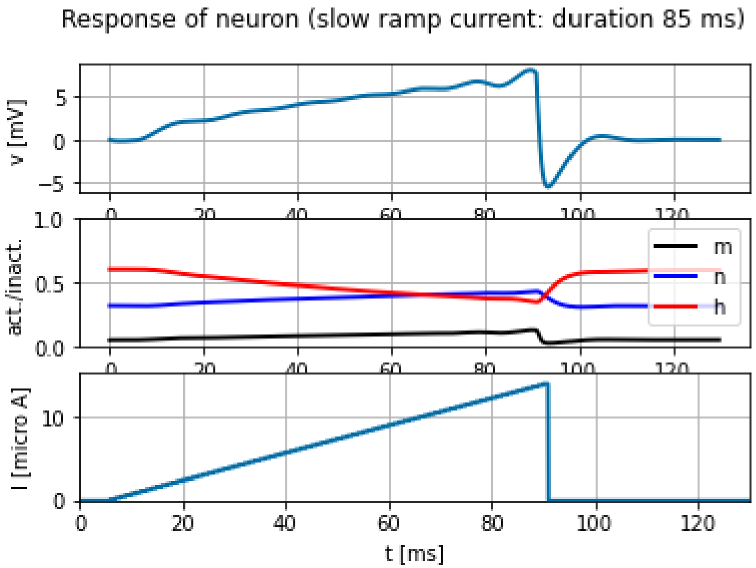
Membrane voltage at time when the current injection stops: 8.87 mV



Membrane voltage at time when the current injection stops: 7.67 mV

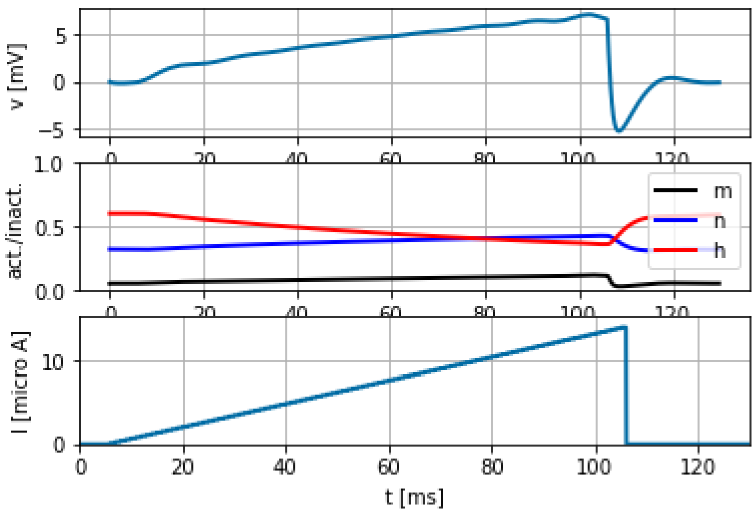


Membrane voltage at time when the current injection stops: 8.07 mV



Membrane voltage at time when the current injection stops: 6.79 mV

Response of neuron (slow ramp current: duration 100 ms)



In [20]:

```
##########################
##   Q2.2.1a conclusion   ##
##########################


# Answer in green box below
```

A2.2.1a conclusion

The minimal duration of a slow ramp current that doesn't induce spike behaviour lays between a duration of 55 ms and 70 ms, i.e. with a slope between 0.255 μA/ms and 0.2 μA/ms.

In [21]:

```
# Enter your code and answer below

###########################
##   Q2.2.1b solution    ##
###########################
    # see code in Q2.2.1a
```

In [22]:

```
# Enter your answer below

###########################
##   Q2.2.1b answer      ##
###########################



# Answer in green box below
```

A2.2.1b conclusion

ramp current duration	membrane voltage at t_end
30 ms	-7.01 mV
55 ms	-4.81 mV
70 ms	8.87 mV
75 ms	7.67 mV
85 ms	8.07 mV
100 ms	6.79 mV

From the table we observe that the first two simulations (where spiking occurs) have a negative membrane voltage when the current injection stops, because an AP was generated during current injection. The other simulations (without spiking) have a positive membrane voltage when the current injection stops. This indicates that no spike (with a hyperpolarisation phase) has occured. Note that the membrane potential remains below the threshold value needed for a spike.

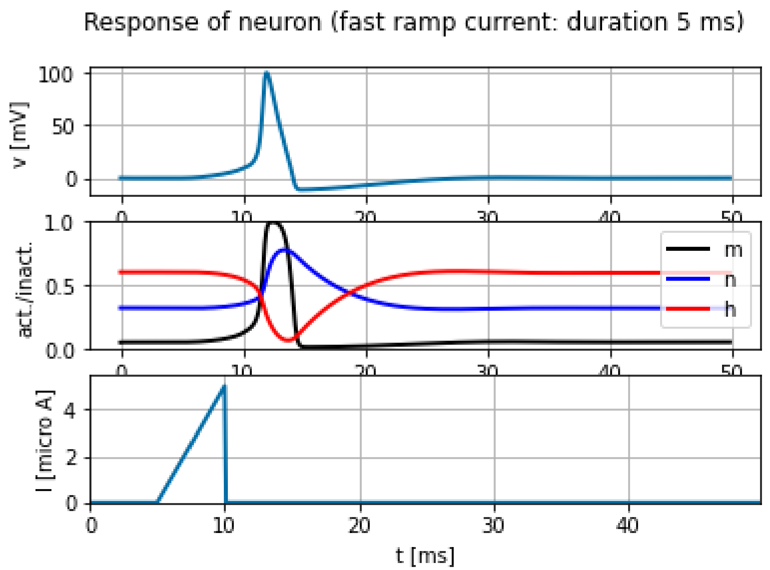
A2.2.2 Fast ramp current

- [Go back to Q2.2.2]

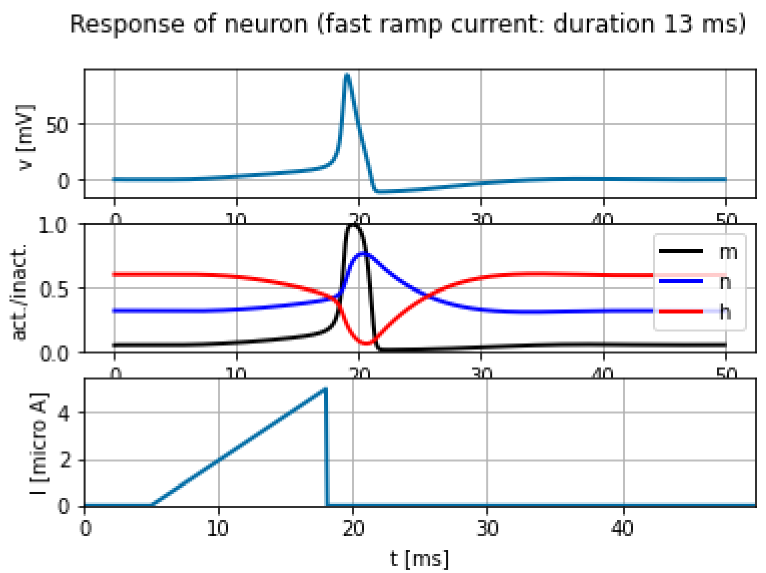
In [23]:

```python
# Enter your code below

# Hint: change the unit time to 0.1ms to get a smooth ramp current.

#########################
##   Q2.2.2a solution   ##
#########################

N = 25
# current_durations = np.linspace(10, 40, N)
current_durations = [10, 18, 19, 20, 25, 30]
duration_total = 50
for t_end in current_durations:
    I_input = input_factory.get_ramp_current(50, int(t_end)*10, 0.1*b2.ms, 0 * b2.uA, 5*b2.uA) # μA
    State_monitor = HH.simulate_HH_neuron(I_input, duration_total*b2.ms)
    sampling_time = duration_total/len(State_monitor.vm[0])
    print('Membrane voltage at time when the current injection stops: {:.2f} mV'.format(State_monitor.vm[0][in
    HH.plot_data(State_monitor, title='Response of neuron (fast ramp current: duration {} ms)'.format(t_end - 
    plt.show()


###########################
##   Q2.2.2a conclusion   ##
###########################


# Answer in green box below
```

```
Membrane voltage at time when the current injection stops: 9.32 mV
```

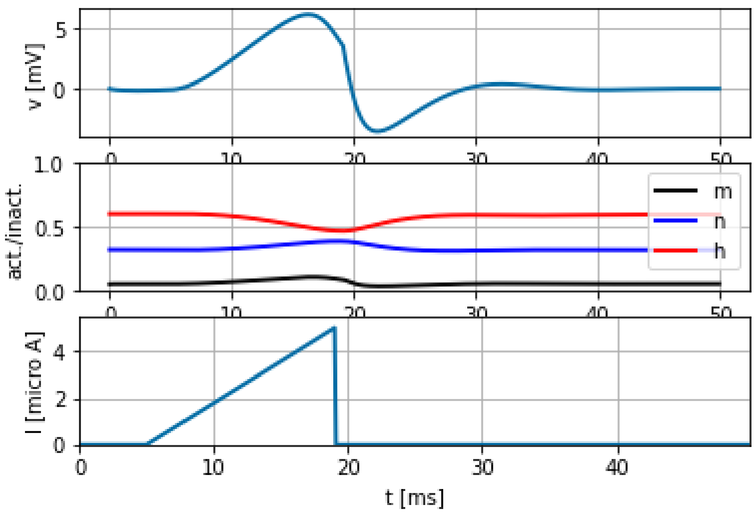


Membrane voltage at time when the current injection stops: 18.80 mV



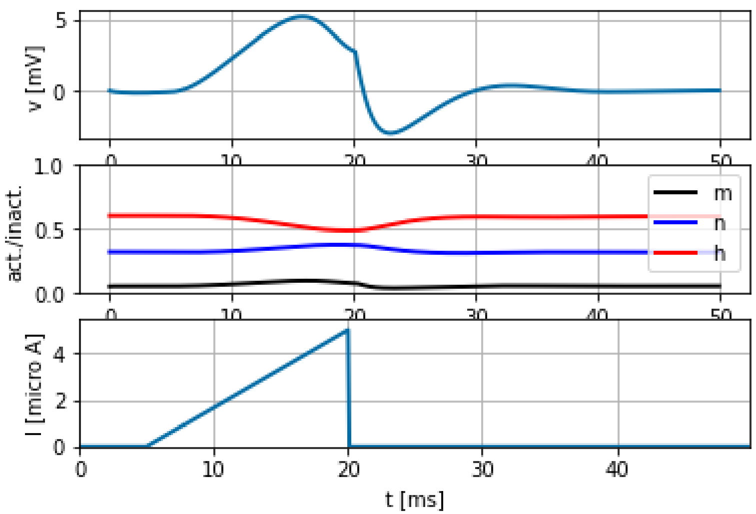
Membrane voltage at time when the current injection stops: 3.71 mV

Response of neuron (fast ramp current: duration 14 ms)

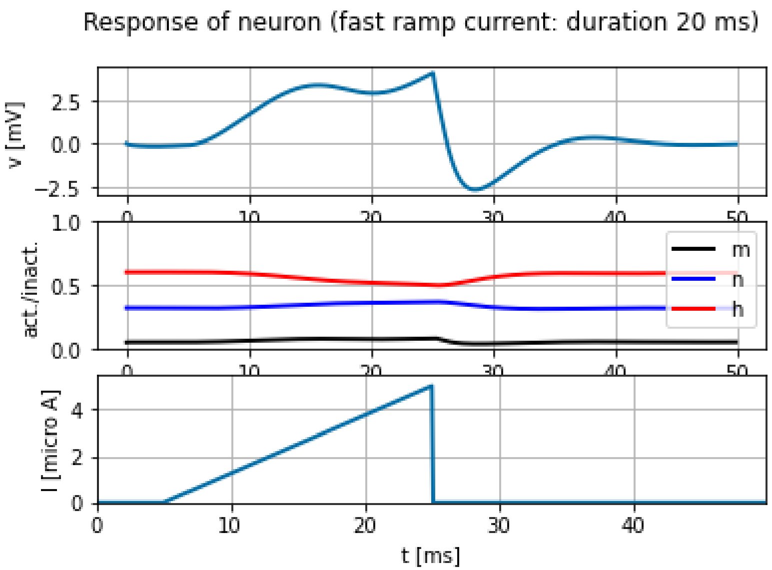



```
Membrane voltage at time when the current injection stops: 2.77 mV
```

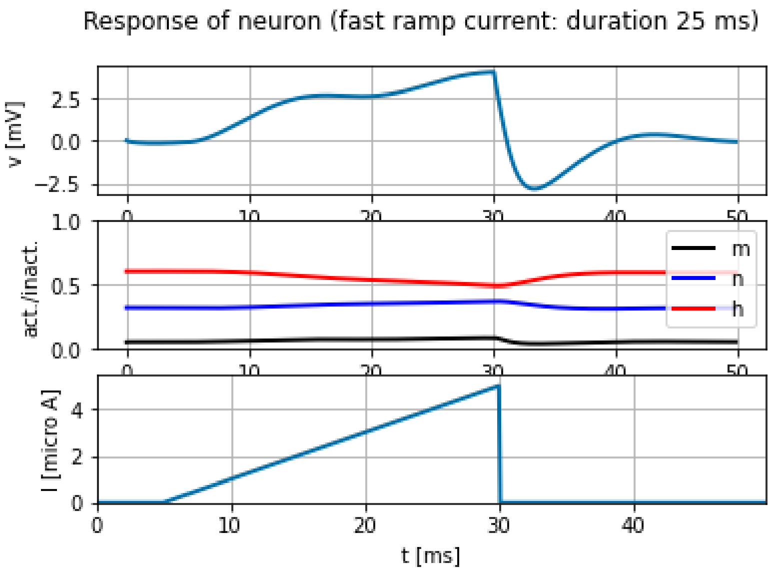
Response of neuron (fast ramp current: duration 15 ms)




```
Membrane voltage at time when the current injection stops: 4.05 mV
```



Membrane voltage at time when the current injection stops: 3.98 mV



A2.2.2a conclusion

\begin{tcolorbox}[colback=green!5]

The minimal duration of a fast ramp current that doesn't induce spike behaviour lays between a duration of 13 ms and 14 ms, i.e. with a slope between 0.385 μA/ms and 0.357 μA/ms.

\end{tcolorbox}

In [24]:

```
#insert your code here

#########################
##   Q2.2.2b solution  ##
#########################

# see code in Q2.2.2a

###########################
##   Q2.2.2b conclusion   ##
###########################


# Answer in green box below

```

A2.2.2b conclusion

ramp current duration	membrane voltage at t_end
5 ms	9.32 mV
13 ms	18.80 mV
14 ms	3.71 mV
15 ms	2.77 mV
20 ms	4.05 mV
25 ms	3.98 mV

From the plots above we can deduce that for the first two durations, the threshold membrane potential (for inducing a spike) was reached. For the other simulations, this value was not reached during the current injection and therefore, no action potential was generated.

\end{tcolorbox}

A2.2.3 Differences

- Go back to Q2.2.3

In [25]:

```
# Enter your answer below

########################
##   Q2.2.3 solution   ##
########################

# Answer in green box below
```

A2.2.3 conclusion

The difference between the fast and slow ramp focusses on the speed of change of activation of the ion channels. A slow ramp gives the model parameters more time to react to the input current and therefore, a smaller difference is observed between the slow (h & n) and fast (m) processes.

A faster ramp limits the action of parameters with a slow time constant (h & n) during possible spike generation. This means that K^+ outflow (n) is initially limited and Na^+ inflow (m) has a bigger (depolarizing) effect. Therefore, a spike is already generated for a lower membrane potential threshold.

The above simulations follow our theoretical reasoning: we conclude that the threshold for a spike generation is around 9 mV and around 4 mV for the slow and fast ramp input current respectively.

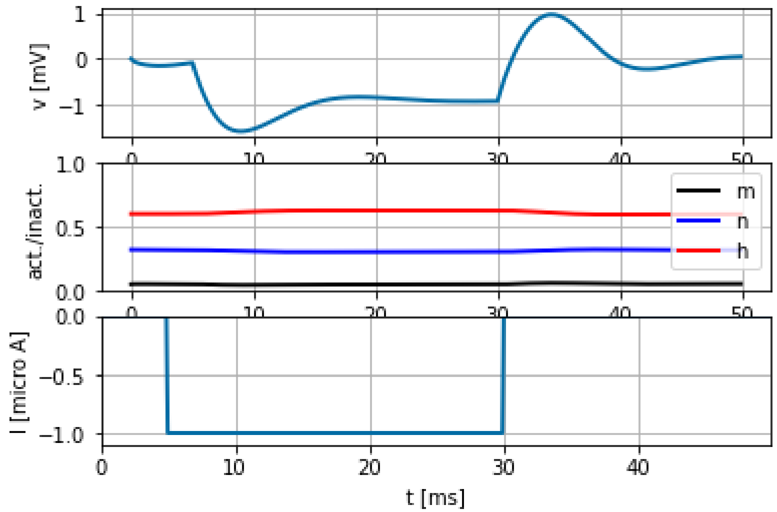
A2.3 Rebound spike

- Go back to Q2.3

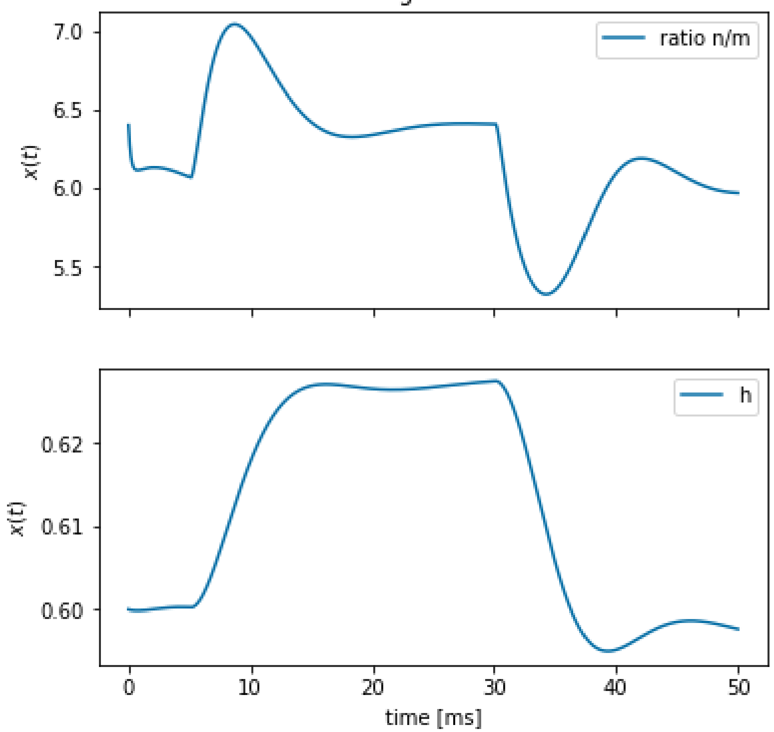
In [26]:

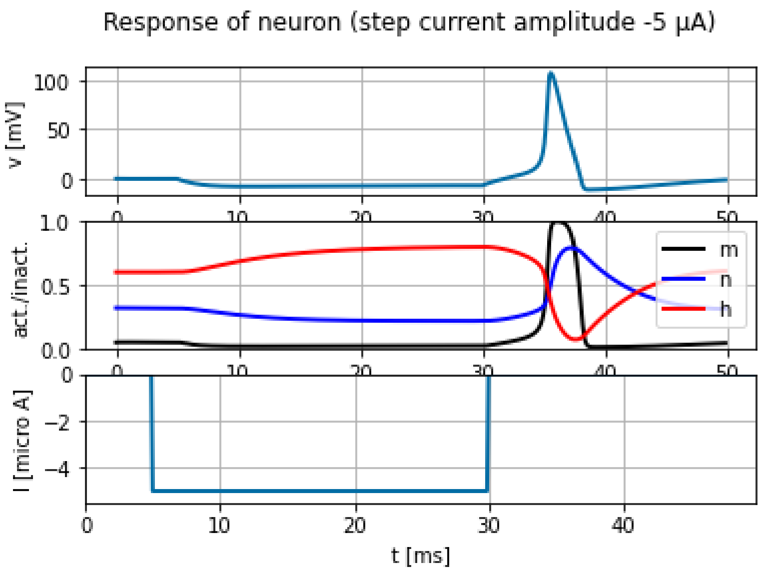
```python
# Enter your answer below

######################
##   Q2.3 solution   ##
######################

current_amplitudes = [-1, -5]

for i in current_amplitudes:
    I_input = input_factory.get_step_current(5,29,b2.ms, i*b2.uA) # μA
    State_monitor = HH.simulate_HH_neuron(I_input, 50*b2.ms)
    m, n, h = State_monitor.m[0], State_monitor.n[0], State_monitor.h[0]
    HH.plot_data(State_monitor, title='Response of neuron (step current amplitude {} μA)'.format(i))
    plt.show()

    time = np.linspace(0, 50, len(m))
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(6, 6), sharex=True)

    ax1.plot(time, n/m, label='ratio n/m')
    ax2.plot(time, h, label='h')

    ax1.set_title('Gating variables')
    ax2.set_xlabel('time [ms]')
    ax1.set_ylabel('$x(t)$')
    ax2.set_ylabel('$x(t)$')
    ax1.legend()
    ax2.legend()
    plt.show()

```

Response of neuron (step current amplitude -1 µA)

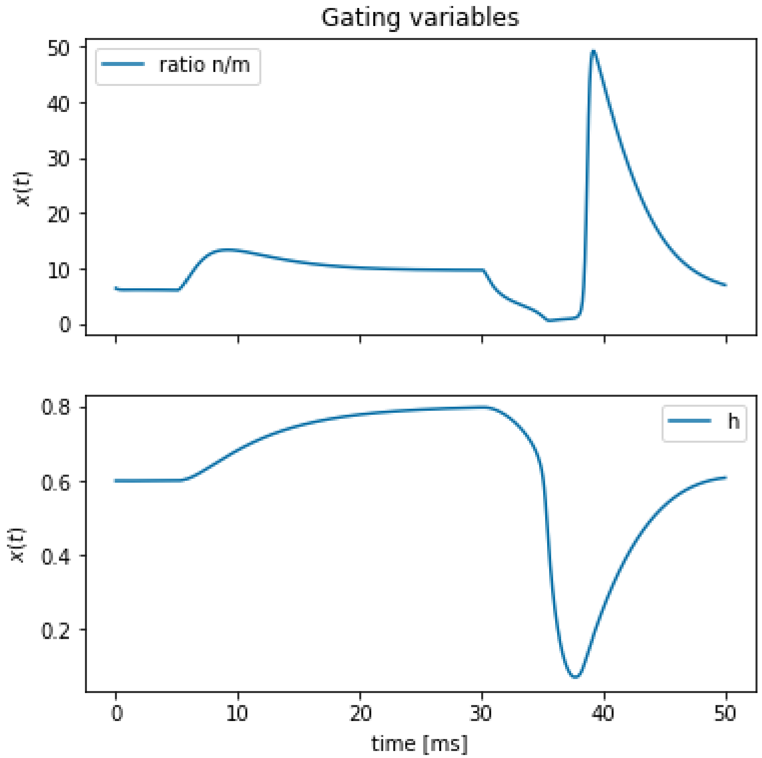


Gating variables





Response of neuron (step current amplitude -5 µA)
v [mV]
act./inact.
m
n
h
I [micro A]
t [ms]



Gating variables
ratio n/m
h
$x(t)$
time [ms]

A2.3 conclusion

Using a negative input current has an inhibiting function on ion channels: when the negative current is switched on (at 5 ms), the effect of n is larger than the effect of m, resulting in a net outflow of positive ions. This causes the threshold to be pulled down and the resting potential is lowered.

If the current is switched off (at 30 ms), there is a steep increase in the current input profile (from negative to zero) and if the membrane threshold is reached, an action potential is generated in the neuron. The AP is generated for the -5 μA step current, but not for the -1 μA case: the amplitude of the step current has to be large enough to generate the rebound spike.

There is less significant change for the h parameter between the -1 μA and -5 μA case, due to the slow time constant of this parameter. At 30 ms, the effect of m (Na^+ inflow) is starts to get larger than the effect of n (K^+ outflow) and hence, the neuron experiences depolarization: an action potential can be generated!

In [ ]:

```
1
```