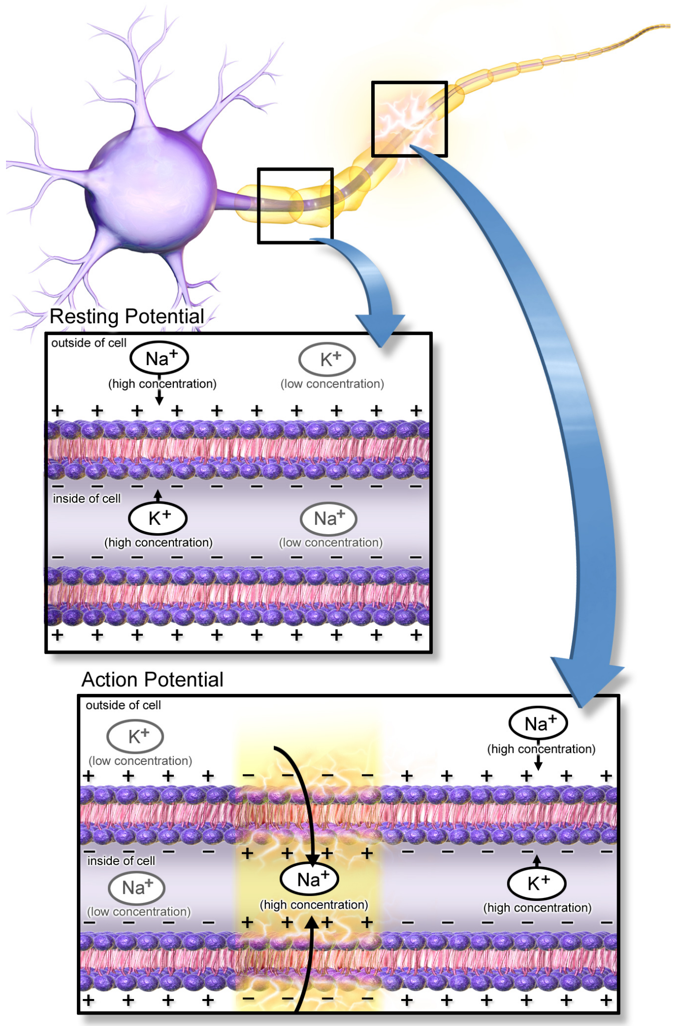
1 Background



An action potential is an event in which the electrical membrane potential of a neuron rapidly rises (as a response to a stimulus) and falls. It is the result of a controlled diffusion of ions across the membrane. A very useful model for the propagation of the action potential in cephalopods is given by

$$\frac{\mathrm{d}V}{\mathrm{d}t} = \left(I - g_{\mathrm{K}} n^4 (V - V_{\mathrm{K}}) - g_{\mathrm{Na}} m^3 h (V - V_{\mathrm{Na}}) - g_L (V - V_L)\right) / C_M \tag{1}$$

with n, m and h being solutions to the ODEs in the table.

Symbol	Meaning (units)	Formula/Value
C_M	Capacitance (μF/cm^2)	0.1 μF/cm^2
I	Applied current (μA/cm^2)	15 nA/cm^2 $= 15 \times 10^{-3}$ μA/cm^2
I_K	Potassium channel current (μA/cm^2)	$g_{\mathrm{K}} n^4 (V - V_{\mathrm{K}})$
I_L	Leakage current (μA/cm^2)	$g_L (V - V_L)$
I_{Na}	Sodium channel current (μA/cm^2)	$g_{\mathrm{Na}} m^3 h (V - V_{\mathrm{Na}})$
V	Action potential (mV)	Initially -65 mV
V_K	Displacement from the equilibrium potential for K^+ (mV)	-77 mV
$[\mathrm{K}^+]_{\mathrm{i}}$	Potassium ion concentration inside (mM/L)	150 mM/L
$[\mathrm{K}^+]_{\mathrm{o}}$	Potassium ion concentration outside (mM/L)	5.5 mM/L
V_{Na}	Displacement from the equilibrium potential for Na^+ (mV)	50 mV
$[\mathrm{Na}^+]_{\mathrm{i}}$	Sodium ion concentration inside (mM/L)	15 mM/L
$[\mathrm{Na}^+]_{\mathrm{o}}$	Sodium ion concentration outside (mM/L)	150 mM/L
V_L	Displacement from the equilibrium potential for leakage (mV)	-54.4 mV
g_{K}	Maxium K conductance (mS/cm^2)	36 mS/cm^2
g_{Na}	Maxium Na conductance (mS/cm^2)	120 mS/cm^2
g_L	Maxium leakage conductance (mS/cm^2)	0.3 mS/cm^2
n	Potassium activation gating variable, probability of K gate being open	Initially 0.317
dn/dt	Rate of change of n (1/ms)	$\alpha_n(1-n) + \beta_n n$
m	Sodium activation gating variable, probability of Na gate being open	Initially 0.05
dm/dt	Rate of change of m (1/ms)	$\alpha_m(1-m) + \beta_m m$
h	Sodium inactivation gating variable, probability of Na gate being inactivated	Initially 0.6
dh/dt	Rate of change of h (1/ms)	$\alpha_h(1-h) + \beta_h h$
α_n	Opening rate constant (1/ms)	$\phi\left(0.01(V+55)/\left(1-\exp\left(-\frac{V+55}{10}\right)\right)\right)$
α_m	Opening rate constant (1/ms)	$\phi\left(0.1(V+40)/\left(1-\exp\left(-\frac{V+40}{10}\right)\right)\right)$
α_h	Opening rate constant (1/ms)	$\phi\left(0.07\exp\left(-(V+65)/20\right)\right)$
β_n	Closing rate constant (1/ms)	$\phi\left(0.125\exp\left(-(V+65)/80\right)\right)$
β_m	Closing rate constant (1/ms)	$\phi\left(4\exp\left(-(V+65)/18\right)\right)$
β_h	Closing rate constant (1/ms)	$\phi\left(1/\left(1+\exp\left(\frac{V+35}{10}\right)\right)\right)$
T	Temperature (°C)	6.3 °C
ϕ	Factor for temperature correction	$3^{(T-6.3)/10}$

2 Problems

1. Solve the equations for n, m and h in the range $0 - 5$ ms.
2. Create interpolations for these functions

```
from scipy.interpolate import interp1d

# Generates the data
x = np.linspace (0, 10, num=10, endpoint=True)
y = np.cos(-x**2 / 9.0)
f = interp1d(x, y, kind='cubic')

# Evaluates
print f(0.2)
```

3. Solve the equation for V taking time steps of 0.001 ms.

4. Plot V as a function of time.

3 Solutions

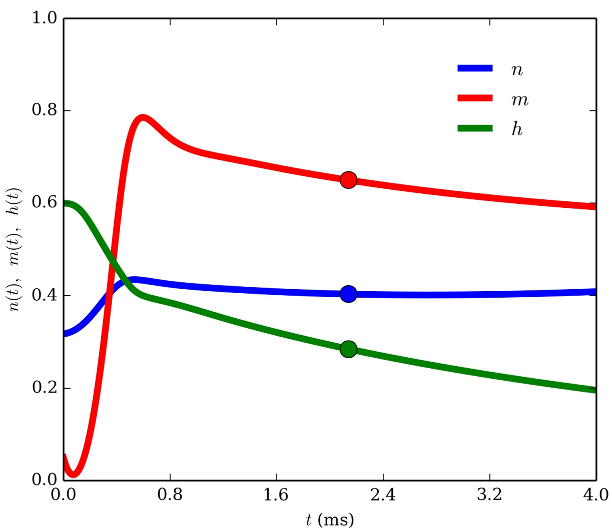
1. Solve the equations for n, m and h in the range $0 - 5$ ms

```
# load modules
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

# unit system
#
# current I : muA/cm^2
# voltage V : mV
# capacitance C : muF/cm^2
# time t : ms
# conductance = 1 / resitance S : mS/cm^2

# define all constants
T = 6.3
I = 15e-3
gK = 36.
gNa = 120.
gL = 0.3
CM = 1.0
VK = -77.
VNa = 50.
VL = -54.4

# define auxiliar constants
phi = 3 ** ((T - 6.3) / 10.)

# define auxiliar functions
def alphan (V):
    return phi * 0.01 * (V + 55) / (1 - np.exp(-0.1 * (V + 55)))

def alpham (V):
    return phi * 0.1 * (V + 40) / (1 - np.exp(-0.1 * (V + 40)))

def alphah (V):
    return phi * 0.07 * np.exp(-0.05 * (V + 65))

def betan (V):
    return phi * 0.125 * np.exp(-0.0125 * (V + 65))

def betam (V):
    return phi * 4.0 * np.exp(-0.0556 * (V + 65))

def betah (V):
    return phi / (1 + np.exp(-0.1 * (V + 35)))


# define the system of equations for the variables (V, n, m, h):
```

```python
#
# dV/dt = FV(V, n, m, h)
# dn/dt = Fn(V, n, m, h)
# dm/dt = Fm(V, n, m, h)
# dh/dt = Fh(V, n, m, h)
def F (V, m, n, h):

    IK = gK * n**4 * (V - VK)
    INa = gNa * m**3 * h * (V - VNa)
    IL = gL * (V- VL)

    FV = (I - IK - INa - IL) / CM
    Fn = alphan(V) * (1 - n) - betan(V) * (n)
    Fm = alpham(V) * (1 - m) - betam(V) * (m)
    Fh = alphah(V) * (1 - h) - betah(V) * (h)

    return FV, Fn, Fm, Fh


# integration time
tmax = 4.0
dt = 1e-3

# arrays
t = np.arange(0, tmax, step=dt)
V = np.zeros(len(t))
n = np.zeros(len(t))
m = np.zeros(len(t))
h = np.zeros(len(t))

# initial conditions
V[0] = -65
n[0] = 0.317
m[0] = 0.05
h[0] = 0.6

# integrate
for i in range(len(t) - 1):

    FV1, Fn1, Fm1, Fh1 = F(V[i], n[i], m[i], h[i])

    V1 = V[i] + 0.5 * dt * FV1
    n1 = n[i] + 0.5 * dt * Fn1
    m1 = m[i] + 0.5 * dt * Fm1
    h1 = h[i] + 0.5 * dt * Fh1

    FV2, Fn2, Fm2, Fh2 = F(V1, n1, m1, h1)

    V[i + 1] = V[i] + dt * FV1
    n[i + 1] = n[i] + dt * Fn1
    m[i + 1] = m[i] + dt * Fm1
    h[i + 1] = h[i] + dt * Fh1
```

2. Create interpolations for these functions

```
# create interpolations
nint = interp1d(t, n)
mint = interp1d(t, m)
hint = interp1d(t, h)

# define time to interpolate
# note that this time is any number in the range (0, tmax)
t0 = 2.14156
print nint(t0), mint(t0), hint(t0)

plt.plot(t, n, 'b-', t0, nint(t0), 'bo')
plt.plot(t, m, 'r-', t0, mint(t0), 'ro')
plt.plot(t, h, 'g-', t0, hint(t0), 'go')
plt.show()
```

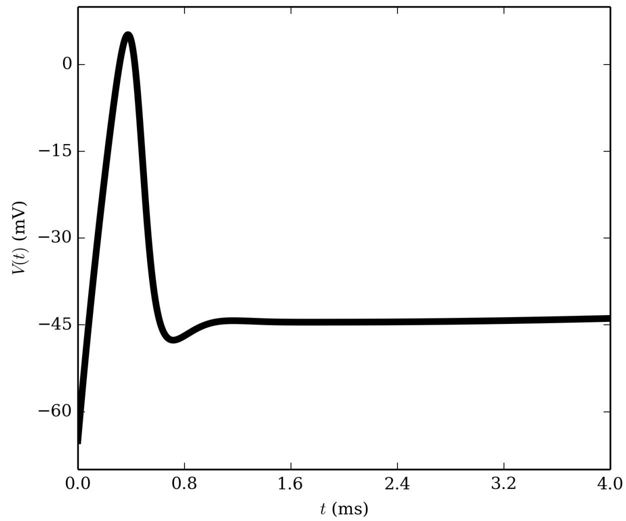


3. Solve the equation for V taking time steps of 0.001 ms.

   Already solved in the first step

4. Plot V as a function of time.

```
plt.plot(t, V, 'k-')
plt.show()
```


0
−15
−30
−45
−60
V(t) (mV)
0.0
0.8
1.6
2.4
3.2
4.0
t (ms)