## Table of Contents



```
function [time, V_membrane, I_d, I_C, I_Na, I_K, I_L, g_Na, g_K, g_L] =
 HHSimulate(num_exc, durations, delay, amplitude, if_plot)
```

# Introduction

In this section of the code, the hyperparameters regarding the update step of the differential equations and length of the simulation is set where t = 10 seconds and dt is 1 milliseconds.

```
simulation_time_in_samples = 1e5;
dt = 1e-3;
```

# Constants

Current is defined in microamperes, time is defined milliseconds. Hence, using the relation q = CV and q = it, capacitance per unit area must be defined in microFarads. First Nernst potentials are hardcoded, then they are updated with respect to the resting voltage. Finally, maximum channel conductances are hard coded.

```
C_m = 1;
% mV %
E_Na = -115; %Sodium nernst is positive
E_K = 12;
E_l = -10.613;
% mS
g_na_bar = 120;
g_k_bar = 36;
g_l_bar = 0.3;

% Resting referenced Nernst potential corrections
V_rest = -70;
E_Na = V_rest - E_Na;
E_l = V_rest - E_l;
E_K = V_rest - E_K;
```

# Vector Initializations

The state variables are initialized with zero first.

```
vm = zeros(1,simulation_time_in_samples);
Delta_vm = zeros(size(vm));
```

# Design stimulation

Depending on the stimulation parameters defined in time, the stimulation current is designed by converting time instance values to the discrete sample values and filling the corresponding sample values with the stimulation intensity.

```
if ischar(num_exc)
    num_exc = str2double(num_exc);
end
excitation_current = amplitude; %uA
I_d = zeros(size(vm));
if num_exc == 1
    duration_in_sample = durations(1) / dt;
    I_d(1:duration_in_sample) = excitation_current(1);
else
    duration1_in_sample = durations(1) / dt;
    duration2_in_sample = durations(2) / dt;
    delay_in_sample = delay/dt;
    I_d(1:duration1_in_sample) = excitation_current(1);
    I_d(duration1_in_sample+delay_in_sample:duration1_in_sample
+delay_in_sample+duration2_in_sample) = excitation_current(1);
    if length(excitation_current) == 2
    I_d(duration1_in_sample+delay_in_sample:duration1_in_sample
+delay_in_sample+duration2_in_sample) = excitation_current(2);
    end
end
```

# Currents

Current vectors are initialized with zero at first. Then, all the activation, inactivation parameters and channel conudctances are initialized with parameters computed for resting potential, *i.e.* 0 deviation from resting potential.

```
I_Na = zeros(size(vm));
I_K = zeros(size(vm));
I_L = zeros(size(vm));
I_C = zeros(size(vm));
I_total = zeros(size(vm));

[a_mi,b_mi,a_hi,b_hi,mi,tau_m,hi,tau_h] = calculate_na_params(0);
[a_ni,b_ni,ni,tau_n] = calculate_k_params(0);

n = ni*ones(size(vm));
m = mi*ones(size(vm));
h = hi*ones(size(vm));

% Channel Conductances
```

```
g_Na = g_na_bar*mi^3*hi*ones(size(vm));
g_K = g_k_bar*ni^4*ones(size(vm));
g_L = g_l_bar*ones(size(vm));

a_n = a_ni*ones(size(vm));
a_m = a_mi*ones(size(vm));
a_h = a_hi*ones(size(vm));
b_n = b_ni*ones(size(vm));
b_h = b_hi*ones(size(vm));
b_m = b_mi*ones(size(vm));
```

# Define membrane voltage.

Membrane voltage is initialized with action potential.

```
V_membrane = V_rest*ones(size(vm));
```

# Action potential generation.

```
for i = 1:simulation_time_in_samples-1
```

# Membrane voltage

Calculate membrane voltage with respect to the extracellular potential.

```
    V_membrane(i) = vm(i) + V_rest;
```

# Conductance

Calculate conductances with the current activation and inactivation parameters.

```
    g_Na(i) = g_na_bar*m(i)^3*h(i);
    g_K(i) = g_k_bar*n(i)^4;
    g_L(i) = g_l_bar; % does not change
```

# Currents

Assuming the Nernst potential remains constant over the time, calculate the channel currents. Using these and the stimulation current, calculate the capacitive current.

```
    I_Na(i) = g_Na(i) * (V_membrane(i)-E_Na);
    I_K(i) = g_K(i) * (V_membrane(i)-E_K);
    I_L(i) = g_L(i) * (V_membrane(i)-E_l);
    I_C(i) = I_d(i) - (I_Na(i) + I_K(i) + I_L(i));
    I_total(i) = I_d(i);
```

# Voltage change

Calculate the change in membrane potential using the capacitor charge equation.

```
    Delta_vm(i) = dt * I_C(i) / C_m;
```

```
    vm(i+1) = vm(i) + Delta_vm(i);
```

## Activation and inactivation parameters

Calculate the parameters $\alpha, \beta$ and m, n, h.

```
    [a_m(i),b_m(i),a_h(i),b_h(i),~,mi,~,hi] = calculate_na_params(vm(i));
    [a_n(i),b_n(i),ni,~] = calculate_k_params(vm(i));
```

## Parameter update

Update the variables that control the channel conductances.

```
    m(i+1) = m(i) + dt * (a_m(i)*(1 - m(i)) - b_m(i)*m(i));
    n(i+1) = n(i) + dt * (a_n(i)*(1 - n(i)) - b_n(i)*n(i));
    h(i+1) = h(i) + dt * (a_h(i)*(1 - h(i)) - b_h(i)*h(i));

end
```

## Visualization

```
time = (1:simulation_time_in_samples)*dt;
if if_plot == 1
% figure
plot(time,V_membrane,'LineWidth',2)
ylabel('Voltage(mV)')
hold on
yyaxis right
plot(time,I_d,'LineWidth',3)
ylabel('Current({\mu}A)')
xlabel('Time(ms)')
legend('Membrane Potential','Excitation
 Current','Location','northeastoutside')
end

end
```

*Published with MATLAB® R2022b*