VS265 Problem 05: Dynamics of the membrane equation

Archit Gupta

1 Problem description

In this challenge problem we simulate the dynamics of a neuron's membrane potential. Neurons form the fundamental building blocks of computation in the brain. Following David Marr's levels of analysis, analyzing the operation of neurons from a mechanistic point of view can be thought of as a gateway to look at how computational algorithms are realized physically in the brain. We start with an electrical model of a neuron's cell membrane comprising of elements like capacitors, resistors and batteries to represent the cell's bi-lipid layer, ion-channels and their resting drift-diffusion equilibria.

With an electrical model at hand, we formulate and Ordinary Differential Equation (ODE) that governs the dynamics of our model and solve it using an ODE solver. We then look at the temporal evolution of membrane potential with injected current and analyze the contributions of various synaptic inputs.

2 Implementation

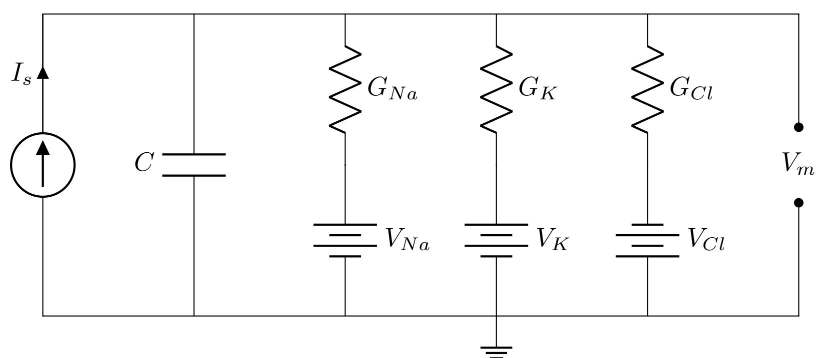

Figure 1: Electrical model of membrane potential of a single neuron.

The membrane potential can be evaluated using the electrical model shown in Fig. 1.

$$\tau\frac{dV_m}{dt} + V_m = \frac{V_r G_{leak} + V_{Na}\Delta G_{Na} + V_K \Delta G_K + V_{Cl}\Delta G_{Cl}}{G_{total}} \tag{1}$$

Here, V_m denotes the membrane potential, G_{total} is the total conductance given by $G_{total} = G_{leak} + \Delta G_{Na} + \Delta G_K + \Delta G_{Cl}$, and τ is the circuit's time constant given by $\tau = \frac{C}{G_{total}}$. We can account for externally injected current in Eq. (1) with the addition of another term giving us

$$\frac{dV_m}{dt} = -\frac{V_m}{\tau} + \frac{V_r G_{leak} + V_{Na}\Delta G_{Na} + V_K \Delta G_K + V_{Cl}\Delta G_{Cl}}{\tau G_{total}} + \frac{I_s}{C}. \tag{2}$$

Listing 1 shows a Matlab class describing the membrane dynamics equation. In addition to the capacitive and conductive elements described in Eq. (1), the current input has also been embedded into the model from Eq. (2). This equation is solved using the ODE15s solver for stiff systems. Code for solving the ODE is presented in Listing 2. Analysis of various synaptic inputs in presented in Listing 3.

3 Results

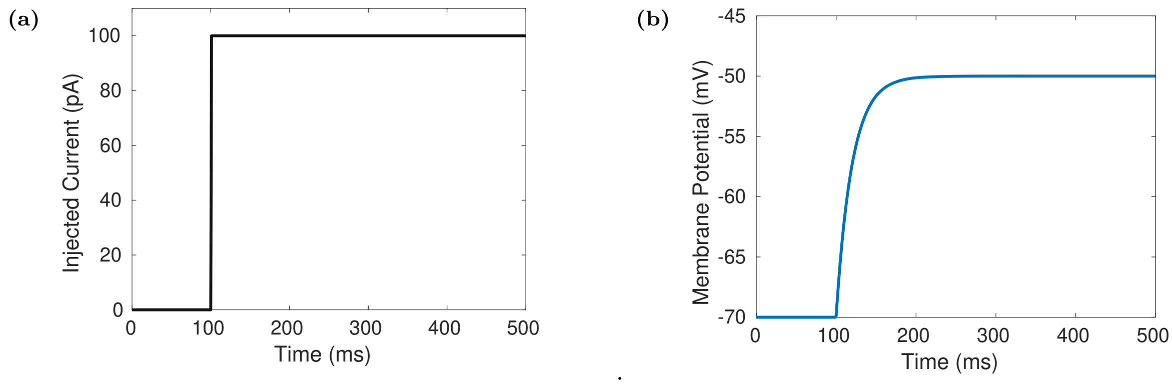

Figure 2: Simulation of membrane dynamics using Eq. (2). (a) Injected current (b) Membrane potential

Fig. 2 shows the simulation results for membrane potential for an input current $I_s(t)$ shown in Fig. 2a. The membrane voltage $V_m(t)$ in response to the current is shown in Fig. 2b.

In Fig. 3, we simulate the membrane equation with different choices of the model parameters. Fig. 3a shows the membrane dynamics over a range of capacitance values. Since the time constant τ can be expressed as $\frac{C}{G_{leak}}$, reducing membrane capacitance reduces the time-constant, which in turn results in faster dynamics. We can observe that for $C = 200pF$, it take longer to charge the membrane capacitance to it final value when compared to a smaller capacitance value (say $C = 10pF$). Additionally, we observe that the steady-state membrane voltage (in the presence of injected current), is independent of the capacitance value.

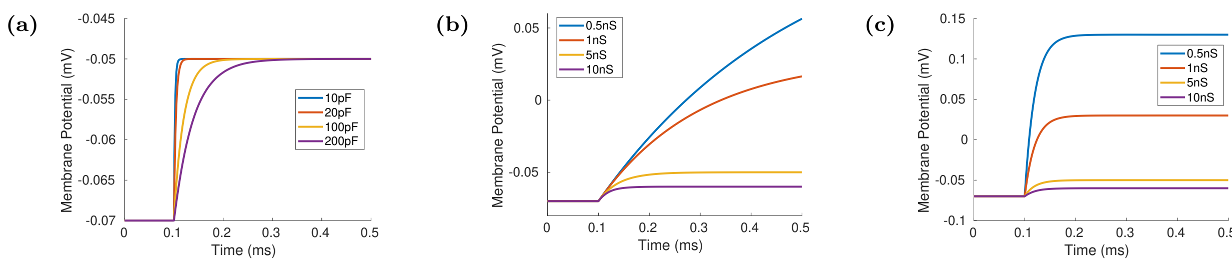

Figure 3: Varying membrane parameters and analyzing the effect on voltage dynamics. (a) Varying membrane capacitance with fixed leakage conductance G_{leak} ($5nS$). (b) Varying leakage conductance with fixed membrane capacitance C ($100pF$). (c) Simultaneous variation of C and G_{leak} such that the time constant $\tau = \frac{C}{G_{leak}}$ is preserved.

When we vary the leakage conductance G_{leak}, in Fig. 3b, we observe that the steady state membrane potential changes. A lower value of G_{leak} leads to a higher steady state membrane potential. In steady state, the membrane potential is related to the leakage conductance as $G_{leak}(V_m - V_r) = I_s$, giving us an inverse relationship between V_m and G_{leak}. Furthermore, the time constant for charging the membrane potential increases as we decrease G_{leak}, slowing down the membrane dynamics as we decrease G_{leak}.

We can also vary C and G_{leak} together, such that their ratio (τ) is preserved. As shown in Fig. 3c, in this case, it takes the same amount of time to charge the cell membrane to its final value. However, the steady-state membrane potential decreases as we increase G_{leak}, as we observed earlier in Fig. 3b.

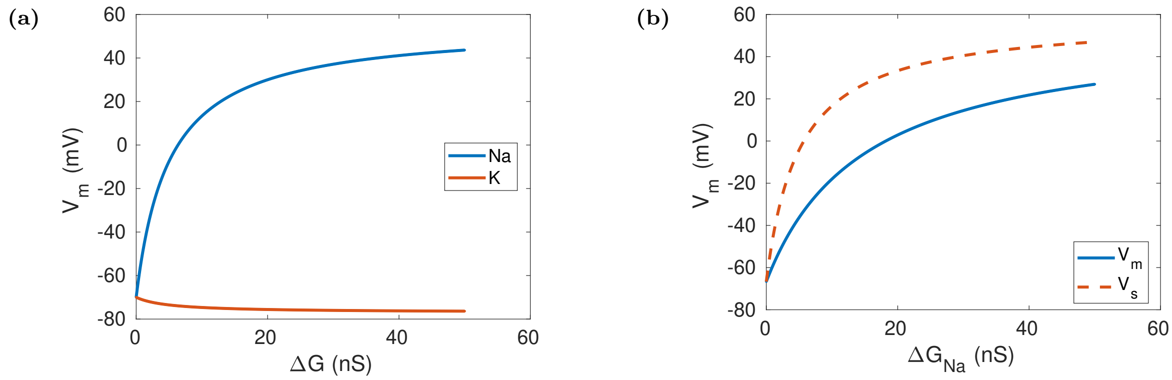

Figure 4: Role of synaptic input in membrane excitability (a) Sweeping Sodium (Na^+) and Potassium (K^+) conductance and measuring steady-state membrane potential (V_m). (b) Measuring steady-state membrane potential V_m as we sweep G_{Na} with Chloride Cl^- conductance increased by $10nS$. V_s (dotted) showing the superposition of contribution from Cl^- and Na^+ conductance individually.

In Fig. 4, we explore the role of synaptic transmission in the excitability of the cell membrane. Fig. 4a shows the steady state membrane potential as we sweep the Sodium (Na^+) and Potassium (K^+) conductance between $0nS$ and $50nS$. The reversal potentials for Na^+ and K^+ are 55mV and -77mV respectively. At higher values of conductances, Na^+ and K^+ drive the steady state membrane voltage to their respective reversal potentials.

We also explore the role of Cl^- in Fig. 4b. If we increase Cl^- conductance by 10nS ($\Delta G_{Cl} = 10nS$), we observe a change in the response of steady state membrane potential with respect to sodium conductance (ΔG_{Na}). Interestingly, this change in response is more than what we would expect from a linear superposition of the contributions of Cl^- and Na^+ as shown in Fig. 4b (dotted line, V_s).

We can express the steady-state membrane potential $V_{m,ss}$ by rewriting Eq. (1) as

$$V_{m,ss} = \frac{V_r G_{leak} + V_{Na}\Delta G_{Na} + V_{Cl}\Delta G_{Cl}}{G_{leak} + \Delta G_{Na} + \Delta G_{Cl}}. \tag{3}$$

Here, we have assumed that $\Delta G_K = 0$ and only the synaptic input from Na^+ and Cl^- channels drives the membrane's steady state. In the steady-state formulation, ΔG_{Cl} appears in both the numerator and denominator in Eq. (3), and therefore, it non-linearly nullifies the effect of Na^+ contribution to lower the membrane's excitability. The primary driver of this non-linearity is the contribution of ΔG_{Cl} to G_{total} in the denominator of Eq. (3).

Cl^- conductance, therefore, plays an interesting, non-linear role in modulating the response of the cell. By activating $Cl-$ channels via GABAergic synapses, the overall excitability is reduced[1].

[1] Excitation of the cell membrane leads to depolarization which in turn is hindered by the activation of $Cl-$ channels.

4 Appendix: Code

Listing 1: Matlab Code for Membrane model

```
classdef membrane_model < handle
    % Class definition for membrane potential.

    % Model parameters defined as properties
    properties (Access = protected)
        G_leak = 5e-9;   % Total leakage conductance
        C = 100e-12;     % Membrane capacitance
        dG_Na = 0;       % Change in Na channel conductance
        dG_K  = 0;       % Change in K channel conductance
        dG_Cl = 0;       % Change in Cl channel conductance
        V_Na  = 0.055;       % Na reversal potential
        V_K   = -0.077;      % K reversal potential
        V_Cl  = -0.065;      % Cl reversal potential
        V_r   = -0.07;       % Membrane's resting potential
        Vm    = -0.07;       % Current membrane potential
    end

    % Class functions
    methods (Access = public)

        % Class constructor
        function mod = membrane_model(m_c, m_g_leak)
            if nargin > 1
                mod.C = m_c;
                mod.m_g_leak = m_g_leak;
            end
        end

        function set_C(model, new_C)
            model.C = new_C;
        end

        function set_G_leak(model, new_G_leak)
            model.G_leak = new_G_leak;
        end

        function set_dG_Na(model, new_dG_Na)
            model.dG_Na = new_dG_Na;
        end

        function set_dG_K(model, new_dG_K)
            model.dG_K = new_dG_K;
        end

        function set_dG_Cl(model, new_dG_Cl)
            model.dG_Cl = new_dG_Cl;
        end

        function [tau, ch_contrib] = get_channel_contrib(model)
            % Get the steady state membrane potential
            % for the current set of parameters.
            contrib_Na = model.V_Na*model.dG_Na;
            contrib_K  = model.V_K*model.dG_K;
            contrib_Cl = model.V_Cl*model.dG_Cl;

            G_total = model.G_leak + model.dG_Na + model.dG_K + model.dG_Cl;
            ch_contrib = (model.V_r*model.G_leak + contrib_Na + ...
                contrib_K + contrib_Cl)/G_total;
            tau = model.C/G_total;
        end

        function vm_val = get_Vm(model)
            vm_val = model.Vm;
        end

        function vm_ss = get_steady_state_vm(model)
            [~, vm_ss] = model.get_channel_contrib();
        end

        function i_contrib = get_current_input(model, t)
            i_contrib = zeros(size(t));
            i_contrib(t > 0.1) = 1e-10;
        end

        function dvdt_val = dvdt(model, t, vm)
            i_contrib = model.get_current_input(t);
            [tau, ch_contrib] = model.get_channel_contrib();
            dvdt_val = ((-vm + ch_contrib)/tau) + (i_contrib/model.C);
        end
    end
end
```

Listing 2: Matlab Code for simulating the Membrane model as an ODE

```
%% Dynamics of membrane equation

% 1. Setup the membrane model for simulation
%    Note that the current source has been built into the model.
% Model parameters
mod = membrane_model();

% Set up an ODE solver to solve the differential equation
[t_pts, y_vals] = ode15s(@mod.dvdt, [0:0.001:0.5], [-0.07]);

% Get the input current for the ode time-points
input_current = mod.get_current_input(t_pts);

%% Plots
% First plot the current input
figure();
plot(1000 * t_pts, 1000 * y_vals, 'LineWidth', 2.4);
set(gca, 'FontSize', 16);
xlabel('Time (ms)');
ylabel('Membrane Potential (mV)');

figure();
plot(1000 * t_pts, 1e12*input_current, 'LineWidth', 2.4, 'Color', ...
    [0.5, 0.5, 0.5, 0.5]);
xlabel('Time (ms)');
ylabel('Injected Current (pA)');
set(gca, 'FontSize', 16);
ylim([0, 1.1e12 * max(input_current)]);

% Now we change the model parameters and try running the membrane dynamics again.
n_vals_to_try = 4;
C_vals = [10, 20, 100, 200] * 1e-12;
G_leak_vals = [0.5, 1, 5, 10] * 1e-9;

%% Plot results for the different capacitance values
cvar_tpts = cell(n_vals_to_try,1);
cvar_yvals = cell(n_vals_to_try,1);
figure();
hold on;
for c_idx = 1:n_vals_to_try
    % Change the model parameters
    mod.set_C(C_vals(c_idx));
    [cvar_tpts{c_idx}, cvar_yvals{c_idx}] = ode15s(@mod.dvdt, ...
        [0:0.001:0.5], [-0.07]);
    plot(cvar_tpts{c_idx}, cvar_yvals{c_idx}, 'LineWidth', 2.4);
end
set(gca, 'FontSize', 16);
xlabel('Time (ms)');
ylabel('Membrane Potential (mV)');
legend('10pF', '20pF', '100pF', '200pF', 'location', 'best');


%% Plot results for the different leakage impedance values
gvar_tpts = cell(n_vals_to_try,1);
gvar_yvals = cell(n_vals_to_try,1);
figure();
hold on;
for g_idx = 1:n_vals_to_try
    mod.set_G_leak(G_leak_vals(g_idx));
    [gvar_tpts{c_idx}, gvar_yvals{c_idx}] = ode15s(@mod.dvdt, ...
        [0:0.001:0.5], [-0.07]);
    plot(gvar_tpts{c_idx}, gvar_yvals{c_idx}, 'LineWidth', 2.4);
end
set(gca, 'FontSize', 16);
xlabel('Time (ms)');
ylabel('Membrane Potential (mV)');
legend('0.5nS', '1nS', '5nS', '10nS', 'location', 'best');

%% Change C and G in a way such that tau is preserved
cgvar_tpts = cell(n_vals_to_try,1);
cgvar_yvals = cell(n_vals_to_try,1);
figure();
hold on;
for g_idx = 1:n_vals_to_try
    mod.set_C(C_vals(g_idx));
    mod.set_G_leak(G_leak_vals(g_idx));
    [cgvar_tpts{c_idx}, cgvar_yvals{c_idx}] = ode15s(@mod.dvdt, ...
        [0:0.001:0.5], [-0.07]);
    plot(cgvar_tpts{c_idx}, cgvar_yvals{c_idx}, 'LineWidth', 2.4);
end
set(gca, 'FontSize', 16);
xlabel('Time (ms)');
ylabel('Membrane Potential (mV)');
legend('0.5nS', '1nS', '5nS', '10nS', 'location', 'best');
```

Listing 3: Matlab Code for steady-state analysis of Membrane potential and contribution of synaptic inputs

```
%% Dynamics of membrane equation

% 1. Setup the membrane model for simulation
%    Note that the current source has been built into the model.
% Model parameters
mod = membrane_model();

% Set up an ODE solver to solve the differential equation
[t_pts, y_vals] = ode15s(@mod.dvdt, [0:0.001:0.5], [-0.07]);

% Get the input current for the ode time-points
input_current = mod.get_current_input(t_pts);

% Change the synaptic input and anlyze the resting membrane potential
dG_Na = [0:0.1:50]*1e-9;
dG_K = [0:0.1:50]*1e-9;
n_Na_vals = length(dG_Na);
n_K_vals  = length(dG_K);

vm_vals_Na = zeros(n_Na_vals,1);
for v_idx = 1:n_Na_vals
    mod.set_dG_Na(dG_Na(v_idx))
    vm_vals_Na(v_idx) = mod.get_steady_state_vm();
end
mod.set_dG_Na(0);

vm_vals_K = zeros(n_K_vals,1);
for v_idx = 1:n_K_vals
    mod.set_dG_K(dG_K(v_idx));
    vm_vals_K(v_idx) = mod.get_steady_state_vm();
end
mod.set_dG_K(0);

figure();
plot(1e9*dG_Na, 1e3*vm_vals_Na, 'Marker', '.', 'LineWidth', 2.4);
hold on;
plot(1e9*dG_K, 1e3*vm_vals_K, 'Marker', '.', 'LineWidth', 2.4);
xlabel('\Delta{G} (nS)');
ylabel('V_{m} (mV)');
set(gca, 'FontSize', 16);
legend('Na', 'K', 'location', 'best');

% Analyzing chloride channel contribution. Set the Chloride channel
% conductance and measure the output corresponding to this
no_cl_vm = mod.get_steady_state_vm();
mod.set_dG_Cl(10e-9);
cl_only_vm = mod.get_steady_state_vm();

vm_vals_Na__with_Cl = zeros(n_Na_vals,1);
for v_idx = 1:n_Na_vals
    mod.set_dG_Na(dG_Na(v_idx))
    vm_vals_Na__with_Cl(v_idx) = mod.get_steady_state_vm();
end
mod.set_dG_Na(0);

superposition_vm = cl_only_vm + vm_vals_Na - no_cl_vm;
figure();
plot(1e9*dG_Na, 1e3*vm_vals_Na__with_Cl, 'Marker', '.', 'LineWidth', 2.4);
hold on;
l_plot = plot(1e9*dG_Na, 1e3*superposition_vm, 'LineStyle', ...
    '--', 'LineWidth', 2.4);
% Make the line a little transparent
original_color = l_plot.Color;
% l_plot.Color = [original_color(:), 0.5];
xlabel('\Delta{G_{Na}} (nS)');
ylabel('V_{m} (mV)');
set(gca, 'FontSize', 16);
legend('V_{m}', 'V_{s}', 'location', 'best');
```