Ctrl + S

Phase 2: Building the Simulator

Welcome to Phase 2 of our mission. In Phase 1, we established the theoretical framework for the Hodgkin-Huxley model. Now, our objective is to translate that theory into a functional simulator using Julia.

Mission Objectives:

1. **Implement Rate Constants:** Define the `α` (alpha) and `β` (beta) functions for each gating variable (m, h, n).
2. **Define Core Equations:** Implement the four core differential equations for V, m, h, and n.
3. **Simulate:** Use `DifferentialEquations.jl` to solve the system and simulate an action potential.
4. **Visualize:** Plot the results.

Let's begin by loading our expeditionary tools.

```
md"""
# Phase 2: Building the Simulator

Welcome to Phase 2 of our mission. In [Phase 1](https://github.com/Nirbhay0007/Neural_Spiking_Dynamics/blob/main/Dynamical_system_of_Neuron.ipynb), we established the theoretical framework for the Hodgkin-Huxley model. Now, our objective is to translate that theory into a functional simulator using Julia.

**Mission Objectives:**
1.  **Implement Rate Constants:** Define the `α` (alpha) and `β` (beta) functions for each gating variable (m, h, n).
2.  **Define Core Equations:** Implement the four core differential equations for $V$, $m$, $h$, and $n$.
3.  **Simulate:** Use `DifferentialEquations.jl` to solve the system and simulate an action potential.
4.  **Visualize:** Plot the results.

Let's begin by loading our expeditionary tools.
"""
```

2. Mission Parameters (Constants)

First, we define the fixed biophysical constants for the squid giant axon, as established by Hodgkin and Huxley.

```
md"""
## 2. Mission Parameters (Constants)

First, we define the fixed biophysical constants for the squid giant axon, as established by Hodgkin and Huxley.
"""
```

3. Rate Constants (α & β Functions)

These are the voltage-dependent rate constants that govern the opening (α) and closing (β) of the ion channel gates. The equations are taken directly from the original 1952 paper.

Note on numerical stability: The original equations for $\alpha_m(V)$ and $\alpha_n(V)$ have a $\frac{0}{0}$ indeterminate form at $V = 25$ mV and $V = 10$ mV, respectively. We must implement these as piecewise functions to handle this limit (which evaluates to 1.0 and 0.1 via L'Hôpital's rule).

```
md"""
## 3. Rate Constants ($\alpha$ & $\beta$ Functions)

These are the voltage-dependent rate constants that govern the opening ($\alpha$) and closing ($\beta$) of the ion channel gates. The equations are taken directly from the original 1952 paper.

**Note on numerical stability:** The original equations for $\alpha_m(V)$ and $\alpha_n(V)$ have a $\frac{0}{0}$ indeterminate form at $V=25$ mV and $V=10$ mV, respectively. We must implement these as piecewise functions to handle this limit (which evaluates to $1.0$ and $0.1$ via L'Hôpital's rule).
"""
```

4. Steady-State & Time Constant Functions

From the rate constants, we can derive the **steady-state activation/inactivation** (`x_∞`) and the **time constant** (τ_x) for each gate `x ∈ {m, h, n}`.

The general forms are:

$$x_\infty(V) = \frac{\alpha_x(V)}{\alpha_x(V) + \beta_x(V)}$$

```
md"""
## 4. Steady-State & Time Constant Functions

From the rate constants, we can derive the **steady-state activation/inactivation** (`x_∞`) and the **time constant** (`τₓ`) for each gate `x ∈ {m, h, n}`.

The general forms are:

```math
x_∞(V) = \frac{\alpha_x(V)}{\alpha_x(V) + \beta_x(V)}
```"""
```

4.1. Specific Gating Variable Functions (m, h, n)

Now we apply the general functions from the previous step to create specific functions for $m_\infty, \tau_m, h_\infty, \tau_h, n_\infty, \tau_n$.

```
md"""
### 4.1. Specific Gating Variable Functions (m, h, n)

Now we apply the general functions from the previous step to create specific functions for $m_\infty, \tau_m, h_\infty, \tau_h, n_\infty, \tau_n$.
"""
```

4.2. Visualize Steady-State Values

Let's plot these functions to understand their behavior. This is a classic plot in computational neuroscience.

* **m_∞ (Na⁺ Activation):** Opens very quickly as the cell depolarizes (becomes more positive).
* **h_∞ (Na⁺ Inactivation):** Is open at rest, but closes as the cell depolarizes. This is the mechanism that "shuts off" the spike.
* **n_∞ (K⁺ Activation):** Opens slowly as the cell depolarizes, allowing K⁺ to flow out and repolarize the membrane.

```
md"""
### 4.2. Visualize Steady-State Values

Let's plot these functions to understand their behavior. This is a classic plot in computational neuroscience.

* **$m_\infty$ (Na⁺ Activation):** Opens very quickly as the cell depolarizes (becomes more positive).
* **$h_\infty$ (Na⁺ Inactivation):** Is open at rest, but closes as the cell depolarizes. This is the mechanism that "shuts off" the spike.
* **$n_\infty$ (K⁺ Activation):** Opens slowly as the cell depolarizes, allowing K⁺ to flow out and repolarize the membrane.
"""
```

```
using DifferentialEquations ,Plots

```

-54.387

```
begin
    C_m = 1.0       # Membrane Capacitance (μF/cm²)
    g_Na = 120.0    # Max Sodium Conductance (mS/cm²)
    g_K = 36.0      # Max Potassium Conductance (mS/cm²)
    g_L = 0.3       # Leak Conductance (mS/cm²)
    E_Na = 50.0     # Sodium Nernst Potential (mV)
    E_K = -77.0     # Potassium Nernst Potential (mV)
    E_L = -54.387   # Leak Nernst Potential (mV)
end
```

```
β_n (generic function with 1 method)
```

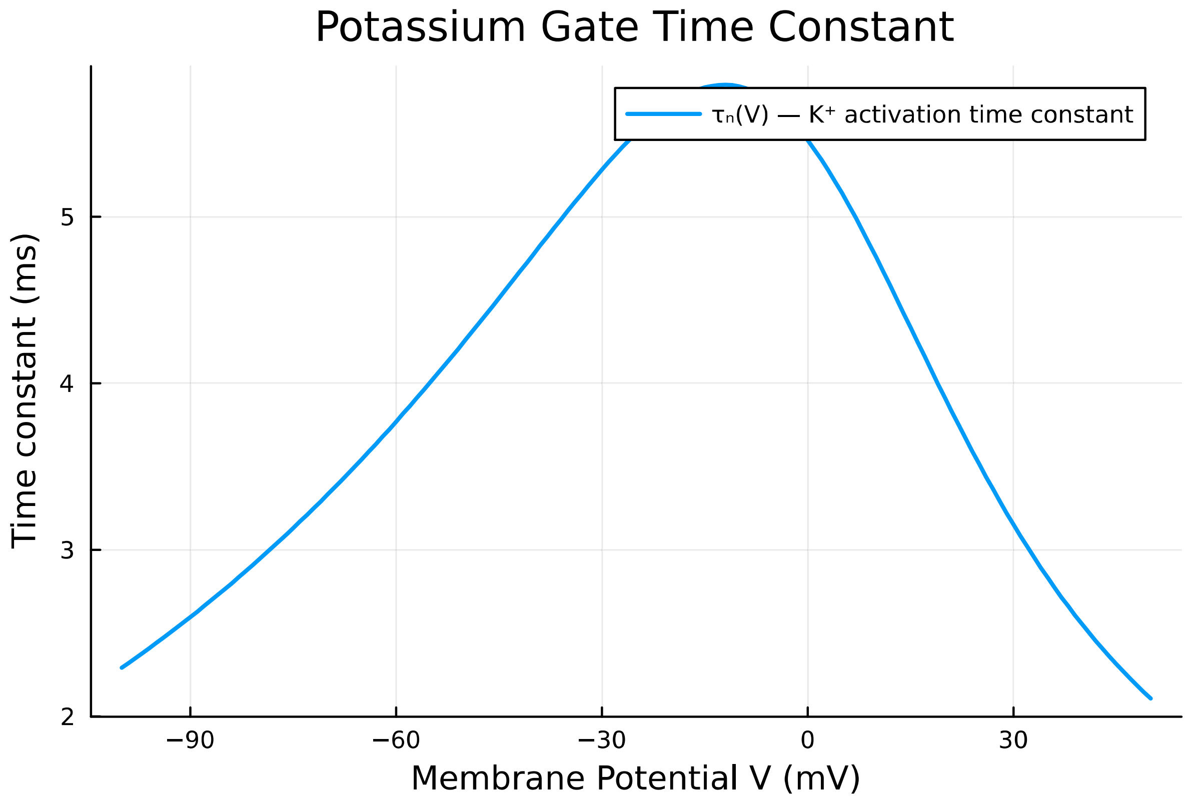
```julia
    # Rate constants for gating variables (Hodgkin-Huxley, Squid Giant Axon)
    # ------------------------------------------------------------
    # Piecewise-limit safeguard for αₘ(V) and αₙ(V)
begin
    α_m(V) = abs(25.0 - V) < 1e-6 ? 1.0 : 0.1 * (25.0 - V) / (exp((25.0 - V) / 10.0) - 1.0)
    β_m(V) = 4.0 * exp(-V / 18.0)

    α_h(V) = 0.07 * exp(-V / 20.0)
    β_h(V) = 1.0 / (exp((30.0 - V) / 10.0) + 1.0)

    α_n(V) = abs(10.0 - V) < 1e-6 ? 0.1 : 0.01 * (10.0 - V) / (exp((10.0 - V) / 10.0) - 1.0)
    β_n(V) = 0.125 * exp(-V / 80.0)

end
```

```
time_constant (generic function with 1 method)
```

```julia
begin
steady_state(α, β, V) = α(V) / (β(V) + α(V))
time_constant(α, β, V) = 1.0 / (α(V) + β(V))
end
```

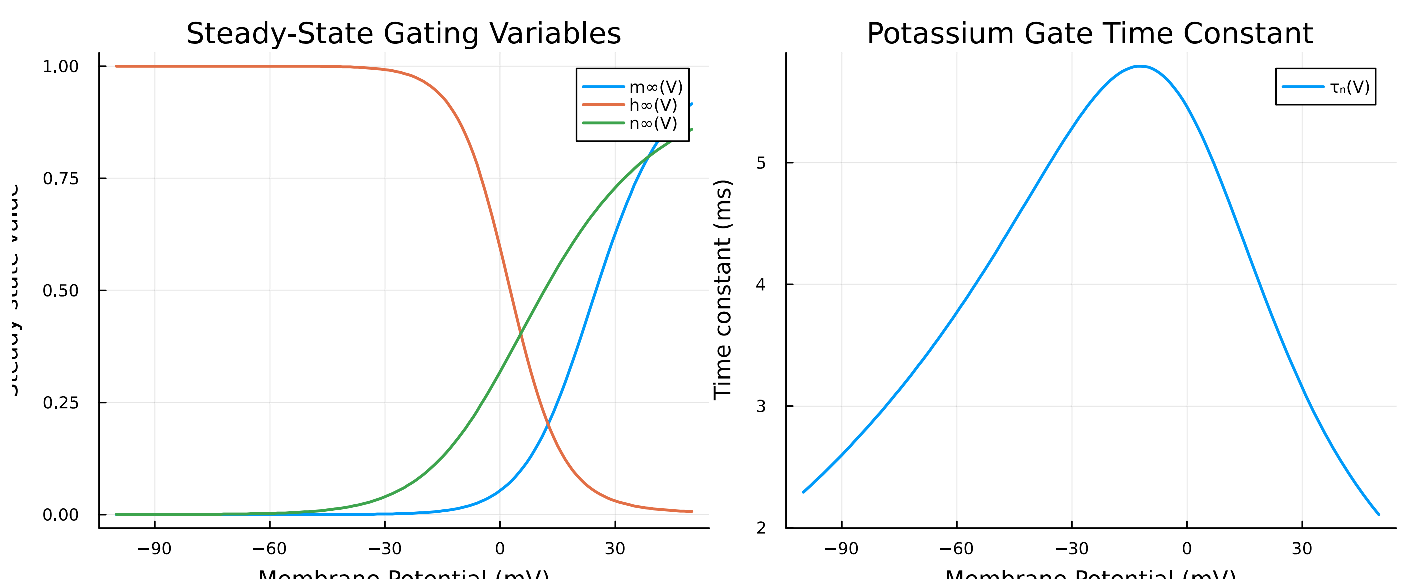
```
tau_n (generic function with 1 method)
```

```julia
begin
m_inf(V) = steady_state(α_m, β_m, V)
tau_m(V) = time_constant(α_m, β_m, V)

h_inf(V) = steady_state(α_h, β_h, V)
tau_h(V) = time_constant(α_h, β_h, V)

n_inf(V) = steady_state(α_n, β_n, V)
tau_n(V) = time_constant(α_n, β_n, V)
end
```




```
begin

# Voltage range for evaluation (mV)
Vs = -100:1:50

# Compute gating values
m_vals = [m_inf(V) for V in Vs]
h_vals = [h_inf(V) for V in Vs]
n_vals = [n_inf(V) for V in Vs]
tau_n_vals = [tau_n(V) for V in Vs]

# ------------------------------
# Plot 1: Steady-state activation & inactivation
# ------------------------------
plot(Vs, m_vals, label="m∞(V) – Na⁺ activation", lw=2, grid=true)
plot!(Vs, h_vals, label="h∞(V) – Na⁺ inactivation", lw=2)
plot!(Vs, n_vals, label="n∞(V) – K⁺ activation", lw=2)
xlabel!("Membrane Potential V (mV)")
ylabel!("Steady-state value")
title!("Steady-State Gating Variables")
plot!(legend=:bottomright)

# ------------------------------
# Plot 2: Time constant τₙ(V)
# ------------------------------
plot(Vs, tau_n_vals,
    label="τₙ(V) – K⁺ activation time constant",
    lw=2,
    xlabel="Membrane Potential V (mV)",
    ylabel="Time constant (ms)",
    title="Potassium Gate Time Constant",
    grid=true)
```

```
end
```




```
begin
p1 = plot(Vs, m_vals, label="m∞(V)", lw=2)
plot!(p1, Vs, h_vals, label="h∞(V)", lw=2)
plot!(p1, Vs, n_vals, label="n∞(V)", lw=2)
xlabel!(p1, "Membrane Potential (mV)")
ylabel!(p1, "Steady-state value")
title!(p1, "Steady-State Gating Variables")

p2 = plot(Vs, tau_n_vals,
          label="τₙ(V)",
          lw=2,
          xlabel="Membrane Potential (mV)",
          ylabel="Time constant (ms)",
          title="Potassium Gate Time Constant")

plot(p1, p2, layout=(1,2), size=(1000,400))

end
```

I_ext (generic function with 1 method)

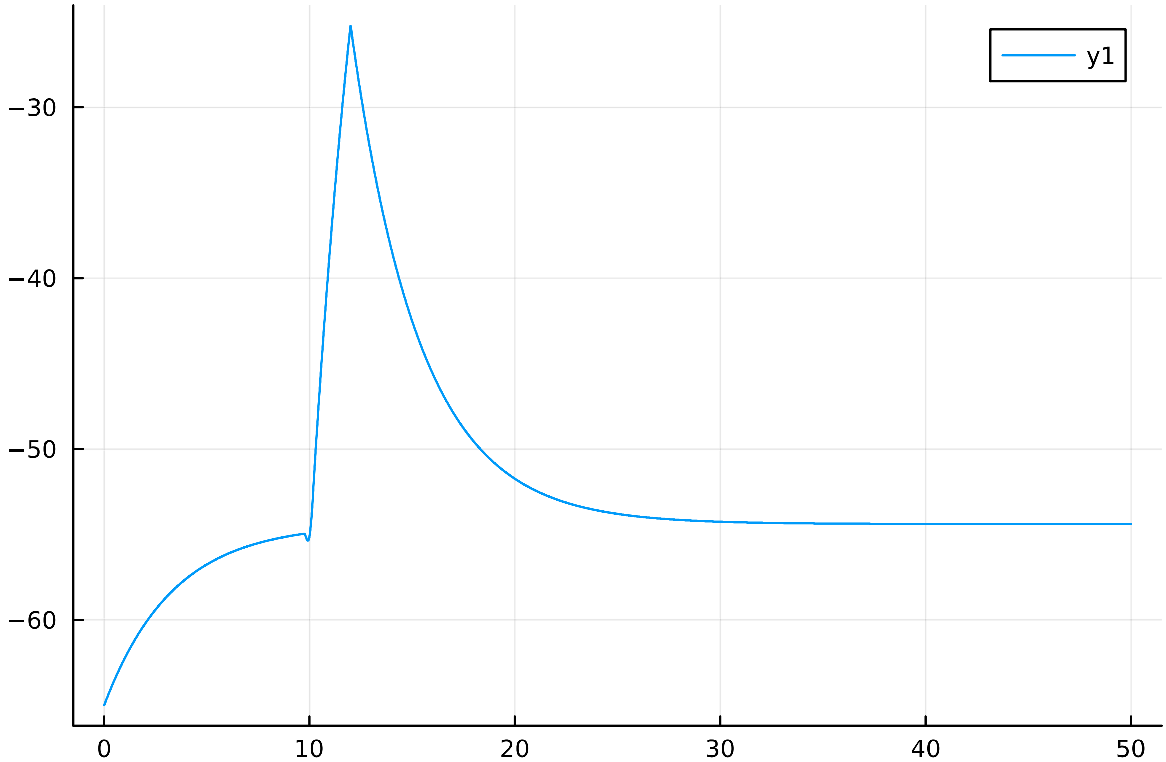
```
function I_ext(t)
    if t > 10 && t < 12
        return 20.0  # try 20 microamp/cm2
    else
        return 0.0
    end
end
```

```
reduced_hh! (generic function with 1 method)
```

```julia
begin
function reduced_hh!(du,u,p,t)  # we are defing a funtion for the solver
    V=u[1]                      # Membrane oltage
    n=u[2]                      # Potassium Voltage  u  -----> [V ,    n]
    INa=g_Na * (m_inf(V)^3)*h_inf(V)*(E_Na-V)
    IK=g_K * (n^4) * ( E_K-V)
    IL = g_L * (E_L  - V  )



    dV = (INa + IL+IK+ I_ext(t))/C_m
    dn = (n_inf(V) - n) / tau_n(V)
    du[1]=dV
    du[2]=dn
end
end
```

```
(0.0, 50.0)
```

```julia
begin
    V0 = -65.0   # mV    initial Voltage  V0

n0 = n_inf(V0)
u0 = [ V0,n0]
tspan = (0.0, 50.0)   # ms

end
```




```

begin
    prob = ODEProblem(reduced_hh!, u0, tspan)
sol = solve(prob,saveat=0.01)
plot(sol.t, sol[1,:])
end
```

```
[0.00147123, 0.00147123, 0.00147125, 0.00147128, 0.00147133, 0.00147139, 0.00147146, 0.001471
```

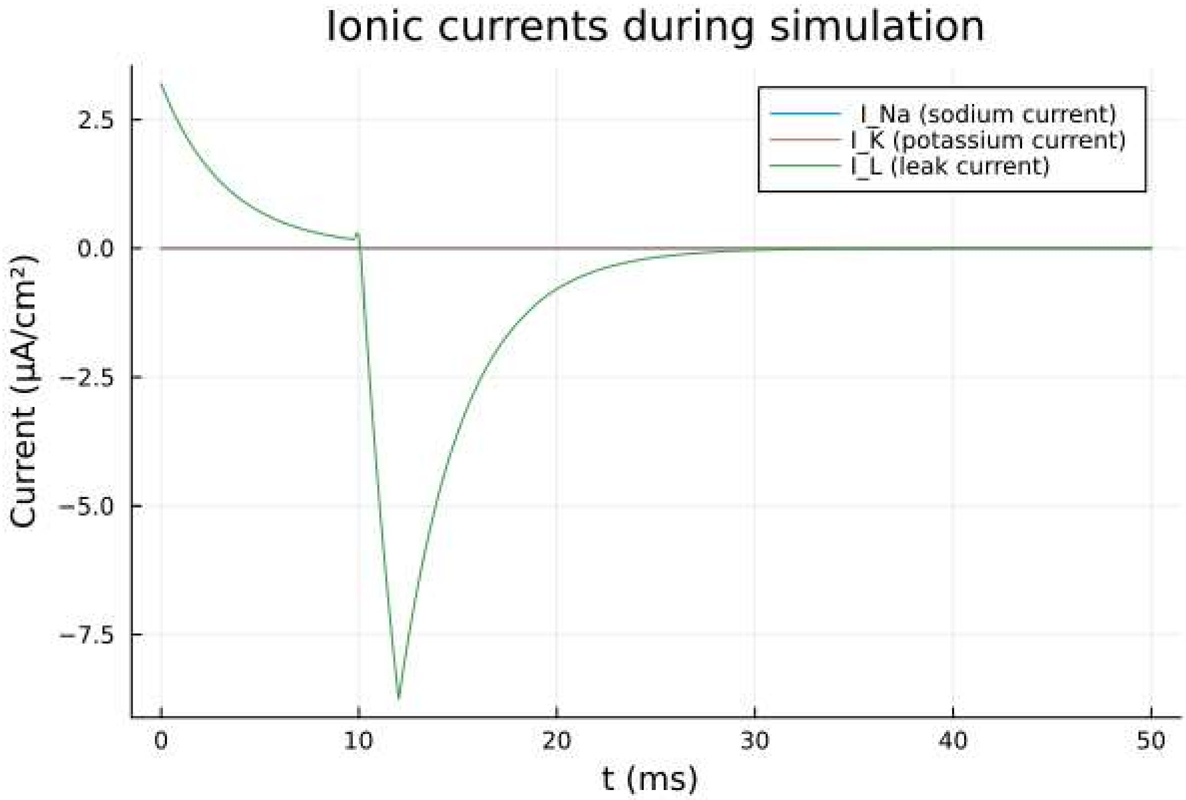
```
begin
    ts = sol.t          # vector of all time points , that solver produced
    Vs3 = sol[1,:]       # Voltage V(t) for th entire simulation
    ns = sol[2,:]       # potassium gate n(t) for the solution
end
```

```
[3.1839, 3.17436, 3.16485, 3.15537, 3.14592, 3.1365, 3.1271, 3.11774, 3.1084, 3.09908, 3.0898,
```

```
begin




    INa_vals = [g_Na * (m_inf(V)^3) * h_inf(V) * (E_Na - V ) for V in Vs3]
    # g_Na  maximum sodium   conductance , ( how easily an sodium ion can flow through these channel whey the gates a opened completly)
    #  m and h gates control the Na+ channel
    # m has three activation gates
    # h has 1 inactivation gate
    # (V - E_Na ) drives na+ towards equilibrium






    IK_vals = [g_K * (n^4)  * (E_K - V ) for (V,n) in zip(Vs3 , ns )]

    # g_k  maximum potassium    conductance , ( how easily an potassium ion can flow through these channel whey the gates a opened completly)
    # n is 4 activation gates the control the ion flow

    # ( V- E_K) drive k+ ions towards equilibrium




    IL_vals = [g_L * (E_L - V ) for V in Vs3]

    # It is a leak current - think of it like a channel that is alawys open (depending on the voltage)

    # g_L = Leak conductance - how leaky the voltage is , when no channels are open
    # V = Membrane potential at a given point
    # E_L = Equilibrium voltage ( voltage at which there is no net flow)

end
```

Now we are going to plot the three graph

```
begin
    md"""
    ### Now we are going to plot the three graph
    """

end
```




```
begin
    p = plot(ts,INa_vals,
            label = " I_Na (sodium current)" ,
            xlabel = "t (ms) " , ylabel = "Current (μA/cm²)")
    plot!(p,ts , IK_vals,label="I_K (potassium current) ")
    plot!(p, ts, IL_vals, label="I_L (leak current)")
title!(p, "Ionic currents during simulation")
p


end
```