Tarea 1

July 15, 2025

Universidad Autónoma Metropolitana - Unidad Iztapalapa (UAM-I)

Maestría en Matemáticas Aplicadas e Industriales (MCMAI)

Taller de Modelado Matemático II - Parte II

Profesor: Dr. Joaquín Delgado Fernández

Alumno: Alan Badillo Salas

Julio 12, 2025. Trimestre 25-P

1 Introducción

Las redes neuronales se han desarrollado gracias al estudio bioquímico del la neurona biológica. Los resultados de los trabajos de Hodking-Huxley y FitzHugh-Nagumo han permitido modelar los mecanismos de activación que ocurren dentro de la neurona biológica mediante el balance en los depósitos de Na^+ (sodio) y K^+ (potasio) en su dinámica de impulsos nerviosos ocurridos dentro del axón (derivados de estudios en calamares gigantes).

- Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes. J. Physiol. (Lond.), 117:500-544.
- Nagumo, J., S. Arimoto, and S. Yoshizawa (1964): An active pulse transmission line simulating nerve axon, Proc IRE. 50: 2061-2070.

Es de nuestro interés poder reproducir algunos resultados del modelo de Hodkind-Huxley y FitzHugh-Nagumo, por lo que estudiaremos las ecuaciones de la dinámica de transmisión de impulsos nerviosos que se deriva del estudio del intercambio de Na^+ y K^+ en la membrana semipermeable, mediante las fugas en la membrana.

Esto se puede modelar similar a un sistema similar a un circuito eléctrico, dónde se considera la función de la membrana como un capacitor eléctrico y al flujo de iones similar a la batería o canales de corriente como se muestra en la **Figura 1**.

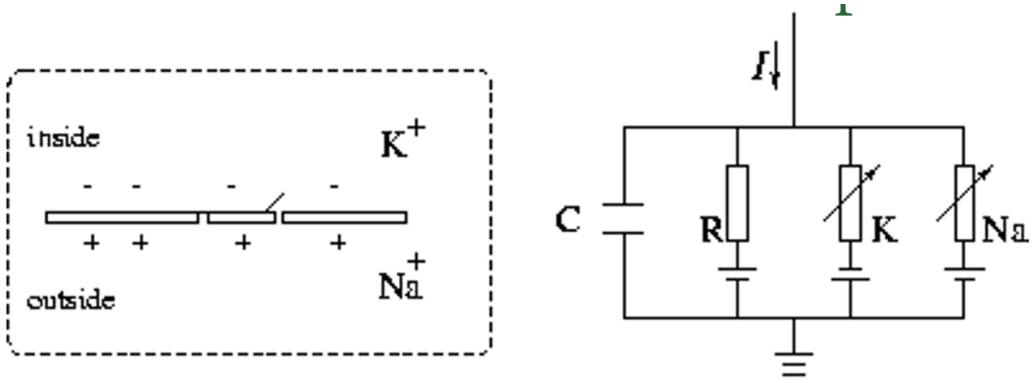


Figura 1. Comparación entre la membrana y la analogía a un capacitor y el flujo de iones en analogía a baterías o canales de corriente.

1.1 Problema

En esta tarea se nos solicita realizar un código en algún lenguaje de programación para reproducir, en la medida de lo posible, el mapa de bifurcación de la **Figura B**, que representa las regiones de activación para los parámetros del estímulo I_2 y ΔI_2 como se muestra en la **Figura A**.

En la región se varian diferentes valores de $I_2 \in [0, 8]$ y $\Delta I_2 \in [0, 6]$. Y en cada punto de la región ocurren diferentes impulsos formando trenes de picos en la región R, impulsos latentes en la región S y supresiones prontas en la región I.

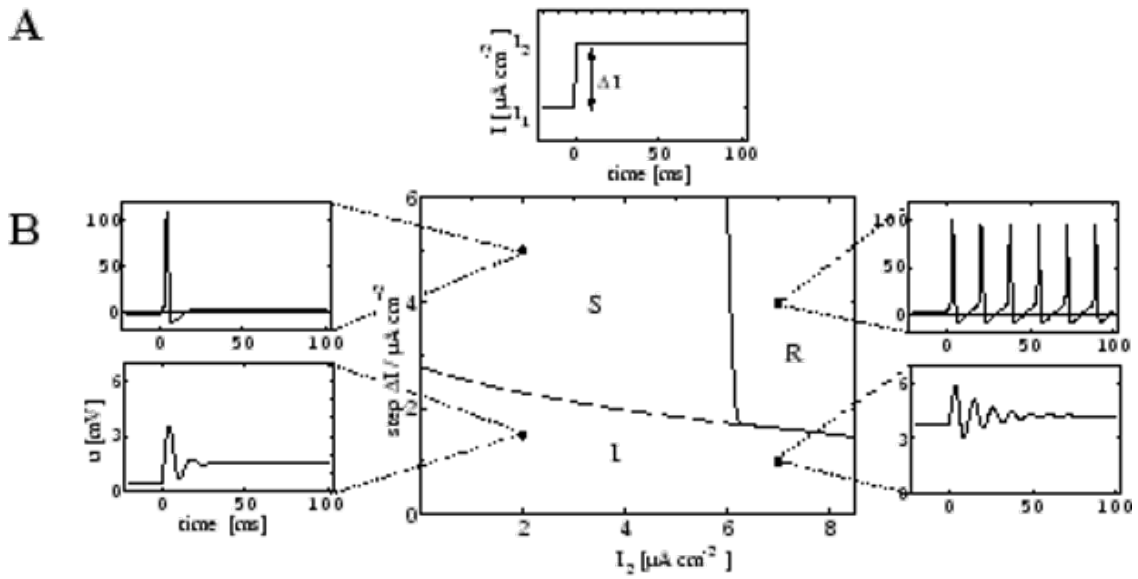


Figura 2. (A) Función de impulso asociada al tiempo $I(t)$ y su diferencial ΔI. (B) Región entre los impulsos y su diferencial, formando activaciones de tren R, latentes S o decadentes I.

1.2 Ecuaciones de la Dinámica de Hodkin-Huxley

Basados en el modelo de corriente, los valores empíricos para las conductancias g_x, los potenciales inversos E_x y las funciones $\alpha_x(u)$ y $\beta_x(u)$ con $x = \{Na(n), K(m), L(h)\}$ son:

$$C_m \frac{dV}{dt} = -g_{Na} m^3 h (V - V_{Na}) - g_K n^4 (V - V_K) - g_L (V - V_L) + I$$

$$\frac{dm}{dt} = \alpha_m(u) \cdot (1 - m) - \beta_m(u) \cdot m$$

$$\frac{dn}{dt} = \alpha_n(u) \cdot (1 - n) - \beta_n(u) \cdot n$$

$$\frac{dh}{dt} = \alpha_h(u) \cdot (1 - h) - \beta_h(u) \cdot h$$

Con los los valores empíricos reportados en la **Figura 3**.

x	E_x	g_x (*)
Na	115 mV	120 $\mathrm{mS/cm^2}$
K	-12 mV	36 $\mathrm{mS/cm^2}$
L	10.6mV	0.3$\mathrm{mS/cm^2}$

$$C_m = 1 \mu F / cm^2$$

x	$\alpha_x(u / \mathrm{mV})$	$\beta_x(u / \mathrm{mV})$
n	$(0.1 - 0.01\,u) / [\exp(1 - 0.1\,u) - 1]$	$0.125 \exp(-u / 80)$
m	$(2.5 - 0.1\,u) / [\exp(2.5 - 0.1\,u) - 1]$	$4 \exp(-u / 18)$
h	$0.07 \exp(-u / 20)$	$1 / [\exp(3 - 0.1\,u) + 1]$

Figura 3. Valores empíricos reportados por Hodkin-Huxley para las conductancias, potenciales reversos y las funciones $\alpha_x(u)$ y $\beta_x(u)$.

(*) Referidos a un potencial de reposo cero.

En la norma actual, debe restarse -65mV a los valores de E_x

2 Simulación

Para resolver este problema, se desarrolló un simulador en *Ruby* resolviendo manualmente las ecuaciones diferenciales e implementando la dinámica según los parámetros I_2 y ΔI_2.

En el simulador podemos ver 3 regiones gráficas:

- *Izquierda Arriba* - Muestra el potencial de la membrana generado por los parámetros que toman I_2 y ΔI mediante la solución de $u(t)$.
- *Izquierda Abajo* - Muestra el impulso generado en el tiempo $I(t)$
- *Derecha* - Muestra una región seleccionable para tomar los valores de los parámetros $I_2 \in [0, 8]$ y $\Delta I_2 \in [0, 6]$.

Al pulsar sobre la región derecha, se lanzarán además 10 soluciones aleatorias cercanas, para ir completando el mapa de las regiones.

El color de cada punto se calcula según el área bajo la curva medida para el impulso.

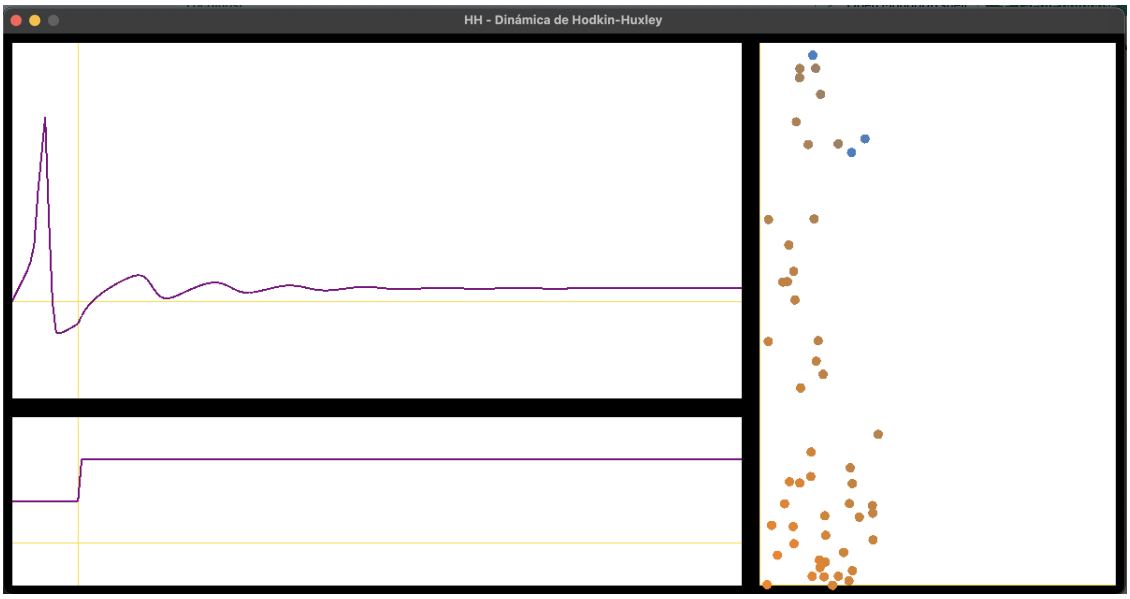
Esto permite colorear del naranja al azul cuando el tren de picos aumenta y el área bajo la curva crece.

2.1 Fase 1 - Impulsos bajos

En la región inferior izquierda, para cuando $I_2 \approx 0$ y $\Delta I_2 \approx 0$ podemos observar cómo el impulso es corto y decae inmediatamente, haciendo referencia a la región I.

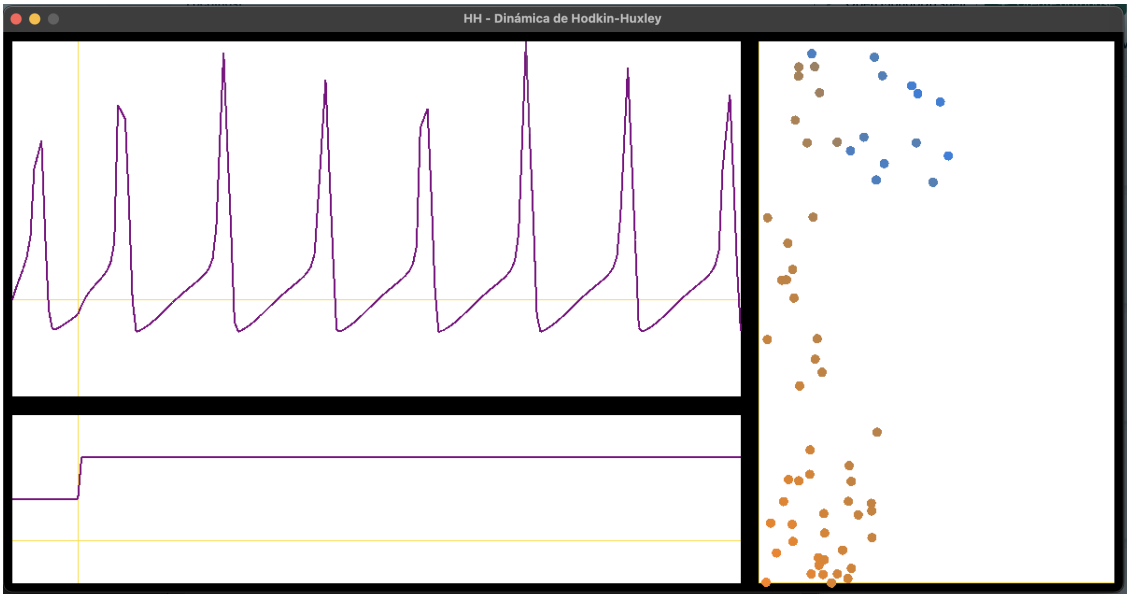
2.2 Fase 2 - Aumento de la cola de impulsos

Cuando en la región se aumenta a ΔI_2 y se mantiene a $I_2 \approx (0, 3)$ el comportamiento del impulso forma colas parecidos a trenes que decaen rápidamente, haciendo referencia a la región S y mostrando un color más ocre.



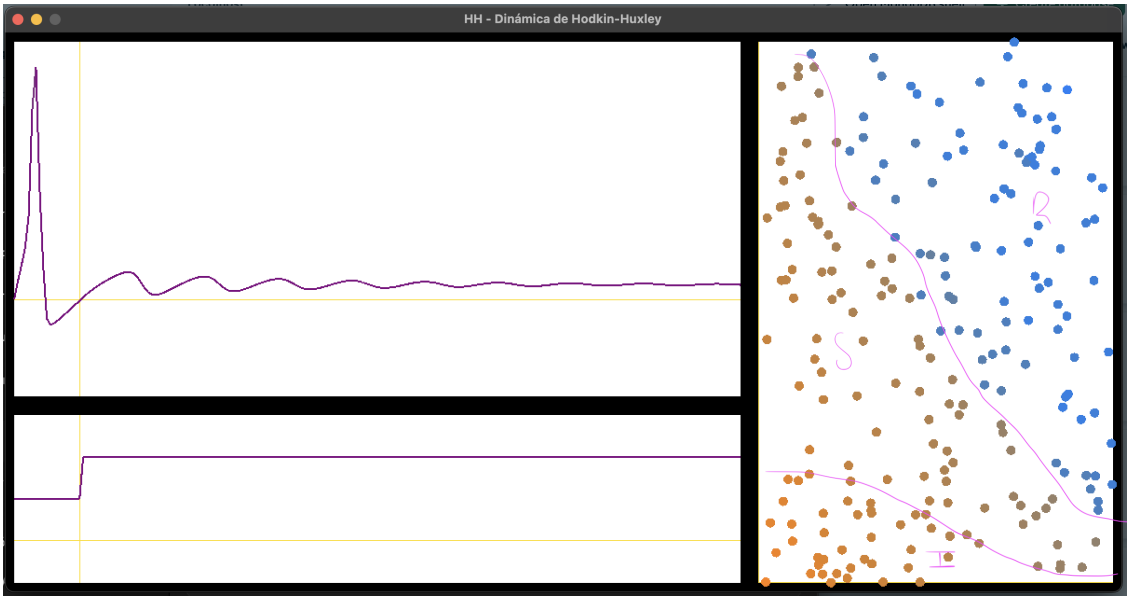
2.3 Fase 3 - Tren de picos

Cuando ΔI_2 crece ($\Delta I_2 \approx 6$), los picos ya no decaen formando así un tren de picos.



2.4 Región completa

Con suficientes simulaciones se puede observar en qué puntos se mantendrá el tren de picos y en cuáles decaerá, formando así un comportamiento similar al reportado en el mapa de HH.



3 Conclusiones

En esta tarea hemos abordado un modelo que ha permitido el desarrollo de las redes neuronales mediante mecanismos biológicos asimilados como modelos eléctricos y ajustados empíricamente para producir activaciones cortas y frecuentes.

Esto nos permite madurar el entendimiento de cómo los modelos bioquímicos usan modelos matemáticos complicados y dinámicas complejas para poder hacer intercambios de señales y energía mediante bombas y depósitos de Na^+ y K^+, formando así instrumentos naturales donde reside la inteligencia natural, que reconstruimos mediante modelos matemáticos, para llegar a una inteligencia artificial.

4 Anexo

A continuación se deja el código del simulador desarrollado.

```
# Universidad Autónoma Metropolitana
# Unidad Iztapalapa
# Maestría en Matemáticas Aplicadas e Industriales
# Profesor: Joaquín Delgado Fernández

# Alan Badillo Salas (alan@nomadacode.com)
# Julio, 2025

# Simulador de la Dinámica del Modelo de Hodkin-Huxley

require 'ruby2d'
require_relative 'curve_box'
require_relative 'point_box'
```

```
$w = 1200
$h = 600

set width: $w
set height: $h
set title: 'HH - Dinámica de Hodkin-Huxley'

Cm = 1;
ENa = 115; gNa = 120;
EK = -12; gK = 36;
EL = 10.6; gL = 0.3;

an = ->(u) { (0.1 - 0.01 * u) / (Math.exp(1 - 0.1 * u) - 1) }
am = ->(u) { (2.5 - 0.1 * u) / (Math.exp(2.5 - 0.1 * u) - 1) }
ah = ->(u) { 0.07 * Math.exp(-u / 20) }

bn = ->(u) { 0.125 * Math.exp(-u / 80) }
bm = ->(u) { 4 * Math.exp(-u / 18) }
bh = ->(u) { 1 / (Math.exp(3 - 0.1 * u) + 1) }

$dI2 = 0.5   # 0..6 (0.1)
$a = 0     # 0..15 (1)
$I2 = 0.5  # 0..8 (0.1)

Unit = ->(t) { (t >= 0) ? 1.0 : 0.0 }

J = ->(t) { $I2 + $dI2 * Unit.call(t - $a) }

Du = ->(t, u, m, n, h) { (-gNa * m ** 3 * h * (u - ENa) - gK * n ** 4 * (u - EK) - gL * (u - EL)
Dn = ->(t, u, m, n, h) { an.call(u) * (1 - n) - bn.call(u) * n }
Dm = ->(t, u, m, n, h) { am.call(u) * (1 - m) - bm.call(u) * m }
Dh = ->(t, u, m, n, h) { ah.call(u) * (1 - h) - bh.call(u) * h }

$tmin = -10
$tmax = 100

$ymin = -30
$ymax = 80

def solver(t)

    t0 = $tmin
    dt = 0.01

    st = t0

    u = 0.0
    m = 0.0
```

```
    n = 0.0
    h = 0.0

    while st < t
        du_val = Du.call(st, u, m, n, h)
        dn_val = Dn.call(st, u, m, n, h)
        dm_val = Dm.call(st, u, m, n, h)
        dh_val = Dh.call(st, u, m, n, h)

        u += dt * du_val
        n += dt * dn_val
        m += dt * dm_val
        h += dt * dh_val

        st += dt
    end

    #print("#{st} #{u} ")

    u

end

curve1 = CurveBox.new(
  ox: 10,
  oy: 10,
  sx: 780,
  sy: 380,
  tmin: $tmin,
  tmax: $tmax,
  ymin: $ymin,
  ymax: $ymax,
  f: method(:solver)
)

curve2 = CurveBox.new(
  ox: 10,
  oy: 410,
  sx: 780,
  sy: 180,
  tmin: $tmin,
  tmax: $tmax,
  ymin: -0.5,
  ymax: 1.5,
  f: J
)

$points = []
```

```
# $points << [4, 4]

curve3 = PointBox.new(
  ox: 810,
  oy: 10,
  sx: 380,
  sy: 580,
  tmin: 0,
  tmax: 8,
  ymin: 0,
  ymax: 6,
  p: $points
)

curve1.draw
curve2.draw
curve3.draw

on :mouse_down do |event|
  if event.x >= 810 && event.x <= 1180 && event.y >= 10 && event.y <= 580
    # print("(#{event.x}, #{event.y}) ")
    xr = (event.x - 810.0) / (1180.0 - 810.0)
    yr = 1 - (event.y - 10.0) / (580.0 - 10.0)

    x = xr * 8
    y = yr * 6

    (1..10).step(1) do |i|

      $I2 = x + rand * 2 - 1
      $dI2 = y + rand * 2 - 1

      c = 0

      delta = ($tmax - $tmin) / 100.0

      ($tmin...$tmax).step(delta) do |t|
        c += solver(t)
      end

      $points << [$I2, $dI2, (c - 100.0) / 1000.0]

    end

    $I2 = x
    $dI2 = y
```

```
    c = 0

    delta = ($tmax - $tmin) / 100.0

    ($tmin...$tmax).step(delta) do |t|
      c += solver(t)
    end

    $points << [$I2, $dI2, (c - 100.0) / 1000.0]

    curve1.draw

    curve3.p = $points

    curve3.draw
  end
end

show
```