Trabajo práctico 1: Dinámica de sistemas acoplados

IGNACIO LEMBO FERRARI[1]

[1] *ignaciolembo@ib.edu.ar*

13 de septiembre de 2023.

1. DINÁMICA DE SISTEMAS ACOPLADOS

A. Modelo de Hodgkin y Huxley

El modelo de Hodgkin y Huxley consiste en un conjunto de cuatro ecuaciones diferenciales ordinarias no lineales que describen el comportamiento del potencial de una neurona

$$C\frac{dV}{dt} = I_{ext} - g_K n^4 (V - V_K) - g_{Na} m^3 h (V - V_{Na}) - g_{Cl}(V - V_{Cl}), \tag{1}$$

$$\frac{dn}{dt} = \frac{n_\infty(V) - n}{\tau_n(V)}, \tag{2}$$

$$\frac{dm}{dt} = \frac{m_\infty(V) - m}{\tau_m(V)}, \tag{3}$$

$$\frac{dh}{dt} = \frac{h_\infty(V) - h}{\tau_h(V)}, \tag{4}$$

donde τ_x y x_∞ para $x = n, m, h$ están dados por

$$x_\infty(V) = \frac{a_x}{a_x + b_x}, \qquad \tau_x = \frac{1}{a_x + b_x}, \tag{5}$$

con τ_x en milisegundos. Los valores de a_x y b_x están dados por

$$a_n = \frac{0.01(V + 55)}{1 - \exp[(-V - 55)/10]}, \tag{6}$$

$$b_n = 0.125 \exp[(-V - 65)/80], \tag{7}$$

$$a_m = \frac{0.1(V + 40)}{1 - \exp[(-V - 40)/10]}, \tag{8}$$

$$b_m = 4 \exp[(-V - 65)/18], \tag{9}$$

$$a_h = 0.07 \exp[(-V - 65)/20], \tag{10}$$

$$b_h = \frac{1}{1 + \exp[(-V - 35)/10]}, \tag{11}$$

con el potencial V expresado en milivolts. Además los potenciales de inversión y las conductancias máximas están dados por: $V_{Na} = 50$ mV, $V_K = -77$ mV, $V_{Cl} = -54.4$ mV, $g_{Na} = 120\ \mathrm{mS/cm^2}$, $g_K = 36\ \mathrm{mS/cm^2}$, $g_{Cl} = 0.3\ \mathrm{mS/cm^2}$. Además, la capacitancia de la membrana es $C = 1\ \mu\mathrm{F/cm^2}$.

B. Dinámica de dos neuronas

Se simuló la dinámica de dos neuronas de Hodgkin y Huxley (neurona 1 y neurona 2) conectadas simétricamente con interacciones sinápticas exitatorias ó inhibitorias. Para cada neurona se utilizaron las ecuaciones y los parámetros descriptos en la sección A. La corriente de interacción sináptica para cada neurona está dada por

$$I_{syn} = -g_{syn}\, s(t)(V - V_{syn}), \tag{12}$$

donde

$$\frac{ds}{dt} = \frac{s_\infty(V_{pre}) - s}{\tau}, \tag{13}$$

con $s_\infty = 0.5\ (1 + \tanh(V/5))$, $\tau = 3$ ms y donde V_{pre} refiere al potencial de la neurona presináptica, es decir, para la neurona 1 es el potencial de la neurona 2 y viceversa. Para la interacción exitatoria se utilizó $V_{syn} = 0$ y para la inhibitoria $V_{syn} = -80$ mV. Se fijó la corriente externa en $I_{ext} = 10\ \mu\mathrm{A/cm^2}$ de manera que las neuronas oscilen periódicamente.

Matemáticamente, la interacción sináptica consiste en sumar la corriente I_{syn} dada por la Ec. (12) en el miembro derecho de la Ec. (1) para cada neurona. De esta manera se tiene, para cada neurona, un sistema de 5 ecuaciones diferenciales acopladas con variables n, m, h, s, V y variable independiente el tiempo t. Esto resulta en un sistema de 10 ecuaciones diferenciales que se resuelven todas al mismo tiempo con el metodo RK4 implementado en Python.

Se tomó como condición inicial para la neurona 1: $V(0) = V_K$, $n(0) = n(V_K)$, $m(0) = m(V_K)$, $h(0) = h(V_K)$ y $s(0) = s(V_K)$. Para la neurona 2: $V = 0$ mV y $n = m = h = s = 0$.

En la Fig. 1 se muestra el caso en que las dos neuronas no es están acopladas ($g_{syn} = 0$ mS/cm^2). En este caso cada neurona se comporta de manera independiente y el desfasaje entre ambas señales es debido a las diferentes condiciones iniciales de cada neurona.

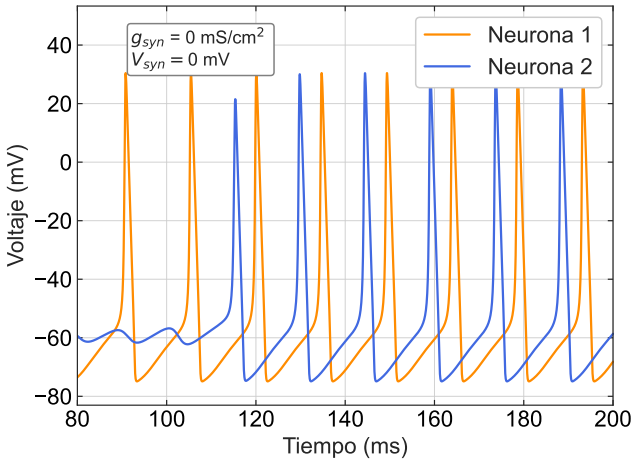

Fig. 1. Potencial de las neuronas 1 y 2 en función del tiempo sin acople sináptico $g_{syn} = 0$mS/cm^2.

Se graficó el potencial de las neuronas 1 y 2 en función del tiempo para un acople sinóptico $g_{syn} = 1$ mS/cm^2 en el caso de interacción sináptica exitatoria $V_{syn} = 0$ mV (Fig. 2) e interacción sináptica inhibitoria $V_{syn} = -80$ mV (Fig. 3). Se observa que luego de un periodo transitorio ambas neuronas se sincronizan. En el caso exitatorio se acoplan en fase y en el inhibitorio en contrafase.

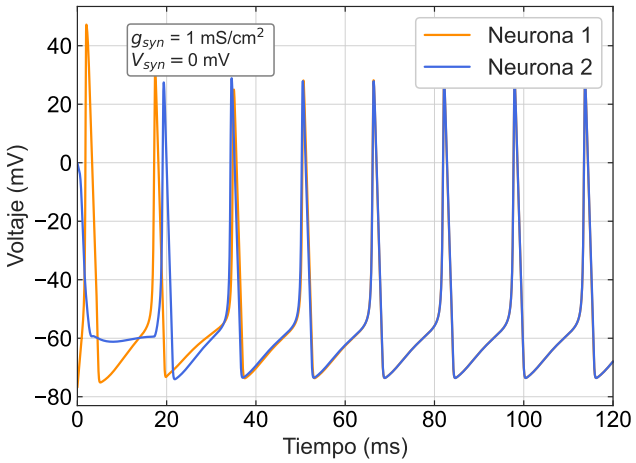

Fig. 2. Potencial de las neuronas 1 y 2 en función del tiempo con acople sináptico $g_{syn} = 1$mS/cm^2 e interacción sináptica exitatoria $V_{syn} = 0mV$.

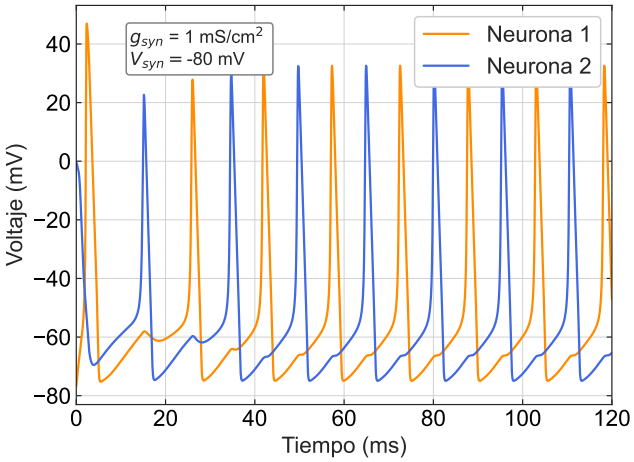

Fig. 3. Potencial de las neuronas 1 y 2 en función del tiempo con acople sináptico $g_{syn} = 1$mS/cm^2 e interacción sináptica inhibitoria $V_{syn} = -80mV$.

Se estudió el desfasaje y la tasa de disparo de las neuronas en función de g_{syn} para interacciones exitatorias ó inhibitorias. Para esto, se tomó g_{syn} entre 0 y 1 mS/cm^2 y se simuló el potencial de cada neurona durante 1500 ms donde empíricamente se vio que el sistema llegaba a un estado estacionario. Para ambos estudios se tomaron los últimos diez picos de voltaje de cada neurona.

El resultado del desfasaje se muestra en la Fig. 4 donde obtuvo que para todo valor de $g_{syn} > 0$ en el caso exitatorio el desfasaje es nulo (neuronas se sincronizan en fase) y para el caso inhibitorio el desfasaje es π (neuronas se sincronizan en contrafase). Cuando $g_{syn} = 0$, no hay acople, las neuronas no se sincronizan y el desfasaje queda determinado por las condiciones iniciales de cada neurona.

En la Fig. 5 se muestran las curvas de la tasa de disparo en función de g_{syn}. Se observa que tanto para la interacción exitatoria como inhibitoria la tasa de disparo decae con g_{syn}, y con mayor pendiente (negativa) en la interacción exitatoria. Se destaca que aumentar el acople disminuye la tasa de disparo en cualquiera de las dos interacciones pero esta disminución es aún mayor cuando la interacción es exitatoria.

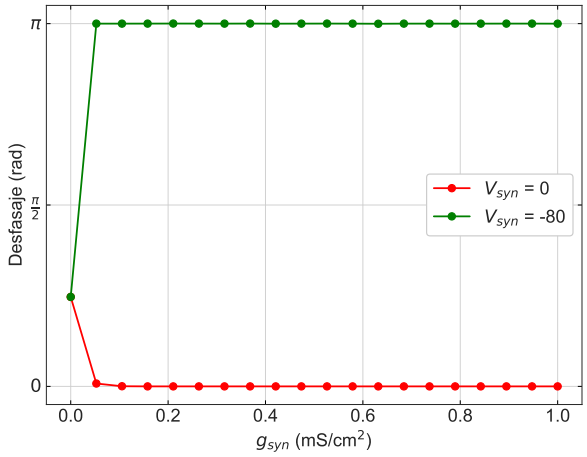

Fig. 4. Desfasaje entre las neuronas 1 y 2 en función del acople sináptico g_{syn}.

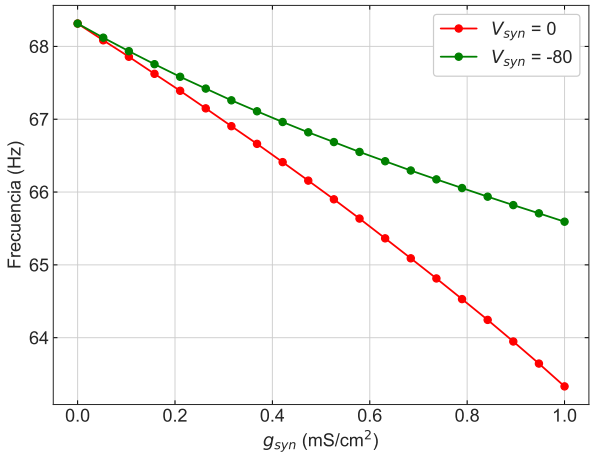

Fig. 5. Tasa de disparo de las neuronas 1 y 2 en función del acople sináptico g_{syn}.

2. SISTEMA CON DOS POBLACIONES DE NEURONAS

Se tiene un sistema con dos poblaciones de neuronas descritas por un modelo de tasa de disparo con una relación f-I semilineal

$$\begin{cases} \tau \dfrac{dh_e}{dt} = -h_e + g_{ee} f_e - g_{ei} f_i + I_e \\ \tau \dfrac{dh_i}{dt} = -h_i + g_{ie} f_e - g_{ii} f_i + I_i, \end{cases} \tag{14}$$

donde $f_a = S(h_a)$ $(a = e, i)$ con $S(x) = xH(x)$ siendo H la función de Heaviside. Dado que las actividades son $h_e, h_i \geq 0$ entonces $H(h_e) = H(h_i) = 1$. Luego, se buscan los puntos fijos del sistema

$$\begin{cases} \tau \dfrac{dh_e}{dt} = -h_e + g_{ee} h_e - g_{ei} h_i + I_e \\ \tau \dfrac{dh_i}{dt} = -h_i + g_{ie} h_e - g_{ii} h_i + I_i \end{cases} \tag{15}$$

el cual se puede escribir de forma matricial como

$$\tau \frac{d\vec{h}}{dt} = \boldsymbol{G}\vec{h} + \vec{I} \tag{16}$$

con

$$\vec{h} = \begin{pmatrix} h_e \\ h_i \end{pmatrix}, \quad \boldsymbol{G} = \begin{pmatrix} g_{ee} - 1 & -g_{ei} \\ g_{ie} & -1 - g_{ii} \end{pmatrix}, \quad \vec{I} = \begin{pmatrix} I_e \\ I_i \end{pmatrix}$$

En el punto fijo tenemos que $\frac{d\vec{h}}{dt} = 0$, es decir, estamos buscando las nulclinas para h_e y h_i. Pidiendo esta condición, el sistema (16) tendrá solución cuando $\det(\boldsymbol{G}) \neq 0$ y resolviendo el sistema de ecuaciones, los valores de las actividades en el punto fijo h_e^* y h_i^* resultan

$$\begin{cases} h_e^* = \dfrac{(1 + g_{ii}) I_e - g_{ei} I_i}{\det(\boldsymbol{G})} \\ h_i^* = \dfrac{g_{ie} I_e - (g_{ee} - 1) I_i}{\det(\boldsymbol{G})} \end{cases} \tag{17}$$

Ahora como $h_e, h_i > 0$, si $\det(\boldsymbol{G}) > 0$ tenemos las siguientes condiciones

$$\begin{cases} I_i < I_e \dfrac{1 + g_{ii}}{g_{ei}} \\ I_i < I_e \dfrac{g_{ie}}{g_{ee} - 1} \end{cases} \tag{18}$$

Por otro lado, si $\det(\boldsymbol{G}) < 0$ tenemos las siguientes condiciones

$$\begin{cases} I_i > I_e \dfrac{1 + g_{ii}}{g_{ei}} \\ I_i > I_e \dfrac{g_{ie}}{g_{ee} - 1} \end{cases} \tag{19}$$

La estabilidad del punto de equilibrio está dada por la parte real de los autovalores del jacobiano $\boldsymbol{J}$ del sistema (14) evaluado en $\vec{h}^*$. Solo en el caso que ambos autovalores tengan parte real negativa, el punto fijo es estable. El jacobiano asociado al sistema (14) evaluado en $\vec{h}^*$ es

$$\boldsymbol{J}|_{\vec{h}^*} = \frac{1}{\tau} \begin{pmatrix} g_{ee} - 1 & -g_{ei} \\ g_{ie} & -1 - g_{ii} \end{pmatrix},$$

cuyos autovalores son

$$\lambda_{1,2} = \frac{1}{2\tau}\left(\mathrm{Tr}(J) \pm \sqrt{(\mathrm{Tr}(J))^2 - 4\,\det(J)}\right). \quad \textbf{(20)}$$

En este caso $\boldsymbol{J}$ coincide con $\boldsymbol{G}$ porque el sistema es lineal en h_e, h_i.

Se analiza la Ec. (20) por casos. Si $\det(\boldsymbol{J}) > 0$ el valor dentro de la raíz es siempre menor a $\mathrm{Tr}(\boldsymbol{J})$. Luego, esto implica que, si los autovalores son reales, es decir,

$$\mathrm{Tr}(\boldsymbol{J})^2 > 4\,\det(\boldsymbol{J}), \quad \textbf{(21)}$$

si $\mathrm{Tr}(\boldsymbol{J}) > 0$ entonces $\lambda_{1,2} > 0$ y si $\mathrm{Tr}(\boldsymbol{J}) < 0$ entonces $\lambda_{1,2} < 0$. Si los autovalores son complejos el signo de la traza directamente es el signo de la parte real del autovalor. Por lo tanto, para que la solución sea estable cuando $\det(\boldsymbol{J}) > 0$, $\mathrm{Tr}(\boldsymbol{J}) < 0$.

Por otro lado, cuando $\det(\boldsymbol{J}) < 0$ los autovalores serán reales y uno será positivo y otro negativo, lo que implica que la solución no puede ser estable.

Entonces, se mostró que $\mathrm{Tr}(\boldsymbol{J}) < 0$ y $\det(\boldsymbol{J}) > 0$ implica estabilidad. La condición para este problema entonces está dada por

$$\begin{cases} \mathrm{Tr}(\boldsymbol{J}) = g_{ee} - g_{ii} - 2 < 0, \\ \det(\boldsymbol{J}) = (g_{ee} - 1)(-1 - g_{ii}) + g_{ei}g_{ie} > 0. \end{cases} \quad \textbf{(22)}$$

Además, si los autovalores son reales y negativos, el comportamiento de la solución es exponencial decreciente hacia el equilibrio, mientras que si son complejos el comportamiento cerca del equilibrio será oscilatorio modulado por una exponencial decreciente.

Se destaca que la estabilidad del punto solo depende de las constantes de interacción sináptica mientras que los valores de $\vec{h}^*$ dependen de estas constantes y de las corrientes I_e e I_i.

A. APÉNDICE

Código 1. Implementación del método Runge-Kutta 4.

```
import numpy as np

def rk4(funcs, y0, t0, dt):
    k1 = np.array([func(t0, y0) for func in funcs])
    k2 = np.array([func(t0 + dt/2, y0 + dt/2 * k1) for func in funcs])
    k3 = np.array([func(t0 + dt/2, y0 + dt/2 * k2) for func in funcs])
    k4 = np.array([func(t0 + dt, y0 + dt * k3) for func in funcs])

    y_new = y0 + dt/6 * (k1 + 2*k2 + 2*k3 + k4)
    return y_new
```

Código 2. Implementación del modelo de una neurona de Hodgkin y Huxley.

```
import numpy as np

#Constantes
gna = 120 #mS/cm2
gk = 36 #mS/cm2
gcl = 0.3 #mS/cm2
Vna = 50 #mV
Vk = -77 #mV
Vcl = -54.4 #mV
C = 1 #muF/cm2
Iext = 10 #muA/cm2
gsyn = 4
ts = 3 #ms
Vsyn = -80 #mV /-80 mV
tau = 3 #ms

# Definicion de las ecuaciones diferenciales
def n0(V):
    an = 0.01*(V+55)/(1-np.exp((-V-55)/10))
    bn = 0.125*np.exp((-V-65)/80)
    return an/(an+bn)
def tn(V):
    an = 0.01*(V+55)/(1-np.exp((-V-55)/10))
    bn = 0.125*np.exp((-V-65)/80)
    return 1/(an+bn)
def m0(V):
    am = 0.1*(V+40)/(1-np.exp((-V-40)/10))
    bm = 4*np.exp((-V-65)/18)
    return am/(am+bm)
def tm(V):
    am = 0.1*(V+40)/(1-np.exp((-V-40)/10))
    bm = 4*np.exp((-V-65)/18)
    return 1/(am+bm)
```

```
def h0(V):
    ah = 0.07*np.exp((-V-65)/20)
    bh = 1/(1+np.exp((-V-35)/10))
    return ah/(ah+bh)
def th(V):
    ah = 0.07*np.exp((-V-65)/20)
    bh = 1/(1+np.exp((-V-35)/10))
    return 1/(ah+bh)
def s0(V_pre):
    return 0.5*(1+np.tanh(V_pre/5))
def f_s(s, Vpre):
    return (s0(Vpre) - s)/ts
def f_n(n, V):
    return (n0(V) - n)/tn(V)
def f_m(m, V):
    return (m0(V) - m)/tm(V)
def f_h(h, V):
    return (h0(V) - h)/th(V)
def f_V(s, n, m, h, V):
    return (Iext - gk*(n**4)*(V-Vk) - gna*(m**3)*(h)*(V-Vna) -
    gcl*(V-Vcl) - gsyn*(s)*(V-Vsyn))/C
funcs = [
    lambda t, y: f_s(y[0], y[9]) ,
    lambda t, y: f_s(y[1], y[8]) ,
    lambda t, y: f_n(y[2], y[8]) ,
    lambda t, y: f_n(y[3], y[9]) ,
    lambda t, y: f_m(y[4], y[8]) ,
    lambda t, y: f_m(y[5], y[9]) ,
    lambda t, y: f_h(y[6], y[8]) ,
    lambda t, y: f_h(y[7], y[9]) ,
    lambda t, y: f_V(y[0], y[2], y[4], y[6], y[8])  ,
    lambda t, y: f_V(y[1], y[3], y[5], y[7], y[9])  ,
]
```

Código 3. Simulación de una interacción exitatoria o inhibitoria entre dos neuronas de Hodgkin y Huxley para un dado valor de g_{syn}.

```
import numpy as np
import matplotlib.pyplot as plt
import HH_neuron as hh
import rk4 as rk4
import seaborn as sns
#sns.set(context='paper')
sns.axes_style("whitegrid")
sns.set_style("ticks")

#Parametros de la simulacion: y = [s1, s2, n1, n2, m1, m2, h1, h2, V1, V2]
y_initial = np.array([hh.s0(hh.Vk), 0, hh.n0(hh.Vk), 0, hh.m0(hh.Vk), 0,
hh.h0(hh.Vk), 0, hh.Vk, 0])
t_initial = 0.0 #ms
```

```
t_final = 500 #ms
time_step = 0.01 #ms

hh.gsyn = 1 #mS/cm2
hh.Vsyn = 0 #mV

#Simulacion
steps = int(t_final/time_step)
t_values = np.empty(steps)
V1_values = np.empty(steps)
V2_values = np.empty(steps)

t = t_initial
y = y_initial

for i in range(steps):
    # Aplicar un paso de Runge-Kutta de orden 4
    y = rk4.rk4(hh.funcs, y, t, time_step)

    t_values[i] = t
    V1_values[i] = y[8]
    V2_values[i] = y[9]

    t += time_step

#Ploteo
plt.plot(t_values, V1_values, color='darkorange', label = "Neurona 1")
plt.plot(t_values, V2_values, color='royalblue', label = "Neurona 2")
plt.xlabel("Tiempo (ms)", fontsize=15)
plt.ylabel("Voltaje (mV)", fontsize=15)
plt.xticks(rotation=0, fontsize=15, color='black')
plt.yticks(fontsize=15, color='black')
plt.tick_params(direction='in', top=True, right=True, left=True, bottom=True)
plt.legend(fontsize=15, framealpha=1, loc = 'upper right')
plt.xlim(0, 120)
plt.grid(True, linewidth=0.5, linestyle='-', alpha=0.9)
plt.annotate('$g_{syn}$ = ' + str(hh.gsyn) + ' mS/cm$^2$ \n$V_{syn}$ = $'
+ str(hh.Vsyn) + ' mV', xy=(0.1, 0.85), xycoords='axes fraction', fontsize=12,
bbox=dict(boxstyle='round,pad=0.2', edgecolor='grey', facecolor='white'))

plt.savefig(f"../Redes-Neuronales/Practica_2/resultados/V_vs_t_{hh.gsyn}_
{hh.Vsyn}.pdf")
plt.savefig(f"../Redes-Neuronales/Practica_2/resultados/V_vs_t_{hh.gsyn}_
{hh.Vsyn}.png", dpi=600)
```

Código 4. Código para calcular el desfasaje entre dos neuronas de Hodgkin y Huxley y la tasa de disparo, para un dado valor de g_{syn} en una interacción exitatoria o inhibitoria

```
import numpy as np
import matplotlib.pyplot as plt
```

```
import HH_neuron as hh
import rk4 as rk4
import seaborn as sns
from scipy.signal import find_peaks

#sns.set(context='paper')
sns.axes_style("whitegrid")
sns.set_style("ticks")
fig1, ax1 = plt.subplots(figsize=(8,6))
fig2, ax2 = plt.subplots(figsize=(8,6))

for Vsyn in [0, -80]: #mV
    gsyns = np.linspace(0, 1, 20) #mS/cm2
    f = []
    shift = []
    hh.Vsyn = Vsyn
    print(hh.Vsyn)
    for gsyn in gsyns:
        hh.gsyn = gsyn
        print(hh.gsyn)
        # Valores iniciales: y = [s1, s2, n1, n2, m1, m2, h1, h2, V1, V2]
        y_initial = np.array([hh.s0(hh.Vk), 0, hh.n0(hh.Vk), 0, hh.m0(hh.Vk), 0,
        hh.h0(hh.Vk), 0, hh.Vk, 0])
        t_initial = 0.0 #ms
        t_final = 1500 #ms
        time_step = 0.01 #ms

        #Simulacion
        steps = int(t_final/time_step)
        t_values = np.empty(steps)
        V1_values = np.empty(steps)
        V2_values = np.empty(steps)

        t = t_initial
        y = y_initial

        for i in range(steps):
            # Aplicar un paso de Runge-Kutta de orden 4
            y = rk4.rk4(hh.funcs, y, t, time_step)

            t_values[i] = t
            V1_values[i] = y[8]
            V2_values[i] = y[9]

            t += time_step

        peaks1, _ = find_peaks(V1_values, height=0)
        peaks2, _ = find_peaks(V2_values, height=0)

        # Encontrar los picos dentro del rango de interes
```

```
        peaks1 = peaks1[-10:]
        peaks2 = peaks2[-10:]
        T1 = (t_values[peaks1[1:]] - t_values[peaks1[:-1]])
        T2 = (t_values[peaks2[1:]] - t_values[peaks2[:-1]])

        T = (np.concatenate((T1,T2))).mean()
        f.append(1.0/(T*1e-3)) #Hz
        Tdiff = np.abs(t_values[peaks1] - t_values[peaks2])
        Tshift = Tdiff.mean()
        shift.append((Tshift%T)/T * 2 * np.pi)

    if(Vsyn == 0):
        color = 'r'
    else:
        color = 'g'

    # Ploteo para la figura 1
    ax1.plot(gsyns, f, "-o", color= color,  label = "$V_{syn}$ = " + str(Vsyn) )
    ax1.set_xlabel("$g_{syn}$ (mS/cm$^2$)", fontsize=15)
    ax1.set_ylabel("Frecuencia (Hz)", fontsize=15)
    ax1.tick_params(direction='in', top=True, right=True, left=True, bottom=True)
    ax1.legend(fontsize=15, framealpha=1)
    ax1.tick_params(axis='x',rotation=0, labelsize=18, color='black')
    ax1.tick_params(axis='y', labelsize=18, color='black')
    #ax1.set_xlim(0, 200)
    ax1.grid(True, linewidth=0.5, linestyle='-', alpha=0.9)
    #ax1.annotate('$g_{syn}$ = ' + str(hh.gsyn) + ' mS/cm$^2$ \n$V_{syn} = $'
    + str(hh.Vsyn) + ' mV', xy=(0.1, 0.825), xycoords='axes fraction',
    fontsize=12, bbox=dict(boxstyle='round,pad=0.2', edgecolor='grey',
    facecolor='white'))
    # Guardar figura 1
    fig1.savefig(f"../Redes-Neuronales/Practica_2/resultados/frecuencia_vs_gsyn.pdf")
    fig1.savefig(f"../Redes-Neuronales/Practica_2/resultados/frecuencia_vs_gsyn.png",
    dpi=600)

    # Ploteo para la figura 2
    ax2.plot(gsyns, shift, "-o", color= color, label = "$V_{syn}$ = " + str(Vsyn))
    ax2.set_xlabel("$g_{syn}$ (mS/cm$^2$)", fontsize=15)
    ax2.set_ylabel("Desfasaje (rad)", fontsize=15)
    ax2.tick_params(direction='in', top=True, right=True, left=True, bottom=True)
    ax2.set_yticks([0, np.pi/2, np.pi])
    ax2.set_yticklabels(["$0$",  r"$\frac{\pi}{2}$", r"$\pi$"])
    ax2.legend(fontsize=15, framealpha=1)
    ax2.tick_params(axis='x',rotation=0, labelsize=18, color='black')
    ax2.tick_params(axis='y', labelsize=18, color='black')
    ax2.grid(True, linewidth=0.5, linestyle='-', alpha=0.9)
    # Guardar figura 2
    fig2.savefig(f"../Redes-Neuronales/Practica_2/resultados/desfasaje_vs_gsyn.pdf")
    fig2.savefig(f"../Redes-Neuronales/Practica_2/resultados/desfasaje_vs_gsyn.png",
    dpi=600)
```