1. Actividad Práctica Nº1 Representación de sistemas y controladores

Se debe redactar un informe que debe realizarse de manera individual por cada estudiante. Dicho informe debe contener:

1- todos los resultados correctos de las consignas dadas.

2- un resumen de las lecciones aprendidas, relacionadas a los Indicadores de logro de la competencia en la que cada estudiante se está formando, descritas en el Curso. https://fcefyn.aulavirtual.unc.edu.ar/course/view.php?id=2913

3- detalles de problemas que fueron resueltos, las fuentes de datos, enlaces, repositorios GitHub con las Recomendaciones finales y Conclusiones de la actividad.

Titular el archivo del informe del modo Apellido_Nombre_TPN1.pdf y subir un único archivo en la solapa correspondiente con los ejercicios resueltos.

Caso de estudio 1. Sistema de dos variables de estado

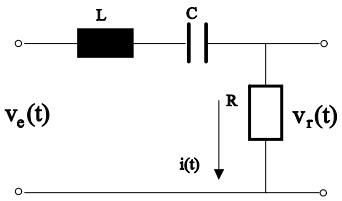

Fig. 1-1. Esquemático del circuito RLC.

Sea el sistema eléctrico de la Fig. 1-1, con las representaciones en variables de estado

$$\dot{x} = A\,x(t) + b\,u(t) \tag{1-1}$$

$$y = c^T x(t) \tag{1-2}$$

donde las matrices contienen a los coeficientes del circuito,

$$A = \begin{bmatrix} -R/L & -1/L \\ 1/C & 0 \end{bmatrix}, b = \begin{bmatrix} 1/L \\ 0 \end{bmatrix}, \tag{1-3}$$

$$c^T = \begin{bmatrix} R & 0 \end{bmatrix}. \tag{1-4}$$

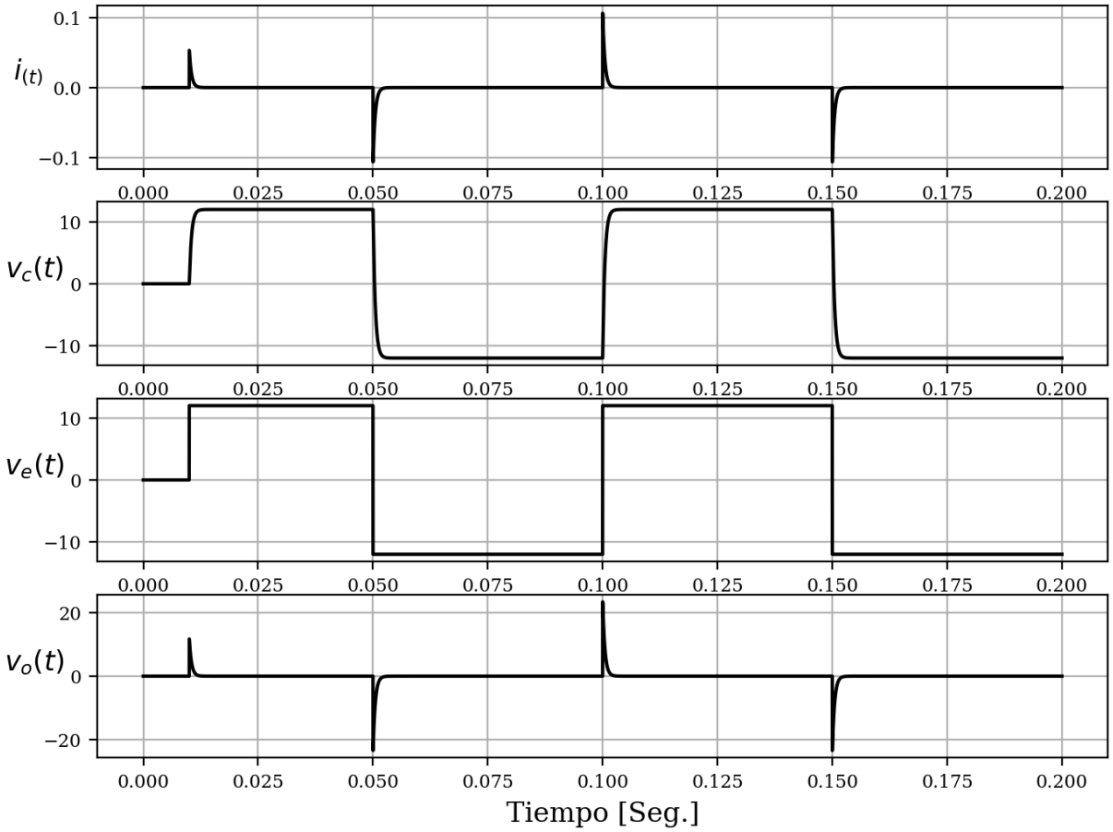

Fig. 1-2. Curvas del circuito RLC para una entrada de 12V.

Ítem [1] Asignar valores a R=220Ω, L=500mHy, y C=2,2μF. Obtener simulaciones que permitan estudiar la dinámica del sistema, con una entrada de tensión escalón de 12V, que cada 10ms cambia de signo.
Ítem [2] En el archivo Curvas_Medidas_RLC_2025.xls (datos en la hoja 1 y etiquetas en la hoja 2) están las series de datos que sirven para deducir los valores de R, L y C del circuito. Emplear el método de la respuesta al escalón, tomando como salida la tensión en el capacitor.
Ítem [3] Una vez determinados los parámetros R, L y C, emplear la serie de corriente desde 0.05seg en adelante para validar el resultado superponiendo las gráficas.

Caso de estudio 2. Sistema de tres variables de estado

Dadas las ecuaciones del motor de corriente continua con torque de carga T_L no nulo, con los parámetros L_{AA}=366 10^{-6}; J=5 10^{-9}; R_A=55,6; B=0; K_i=6,49 10^{-3}; K_m=6,53 10^{-3}:

$$\frac{di_a}{dt} = -\frac{R_A}{L_{AA}} i_a - \frac{K_m}{L_{AA}} \omega_r + \frac{1}{L_{AA}} v_a \qquad (1\text{-}5)$$

$$\frac{d\omega_r}{dt} = \frac{K_i}{J} i_a - \frac{B_m}{J} \omega_r - \frac{1}{J} T_L \qquad (1\text{-}6)$$

$$\frac{d\theta_t}{dt} = \omega_r. \qquad (1\text{-}7)$$

Implementar un algoritmo de simulación para inferir el comportamiento de las variables interés mediante integración Euler con $\Delta t=10^{-7}$ segundos para calcular su operación con un controlador:
Ítem [4] Obtener el torque máximo que puede soportar el motor modelado mediante las Ecs. (1-5) (1-6) y (1-7) cuando se lo alimenta con 12V, graficando para 5 segundos de tiempo la velocidad angular y corriente i_a para establecer su valor máximo como para dimensionar dispositivos electrónicos.
Ítem [5] A partir de las curvas de mediciones de las variables graficadas en la Fig. 1-3, se requiere obtener el modelo del sistema considerando como entrada un escalón de 12V, como salida a la velocidad angular, y al torque de carga T_L aplicado una perturbación. En el archivo Curvas_Medidas_Motor_2025.xls están las mediciones, en la primer hoja los valores y en la segunda los nombres. Se requiere obtener el modelo dinámico, para establecer las constantes del modelo (1-5) (1-6).

Ítem [6] Implementar un PID en tiempo discreto para que el ángulo del motor permanezca en una referencia de `1radian` sometido al torque descripto en la Fig. 1-3. (Tip: partir de K_P=0,1; K_i=0,01; K_D=5).

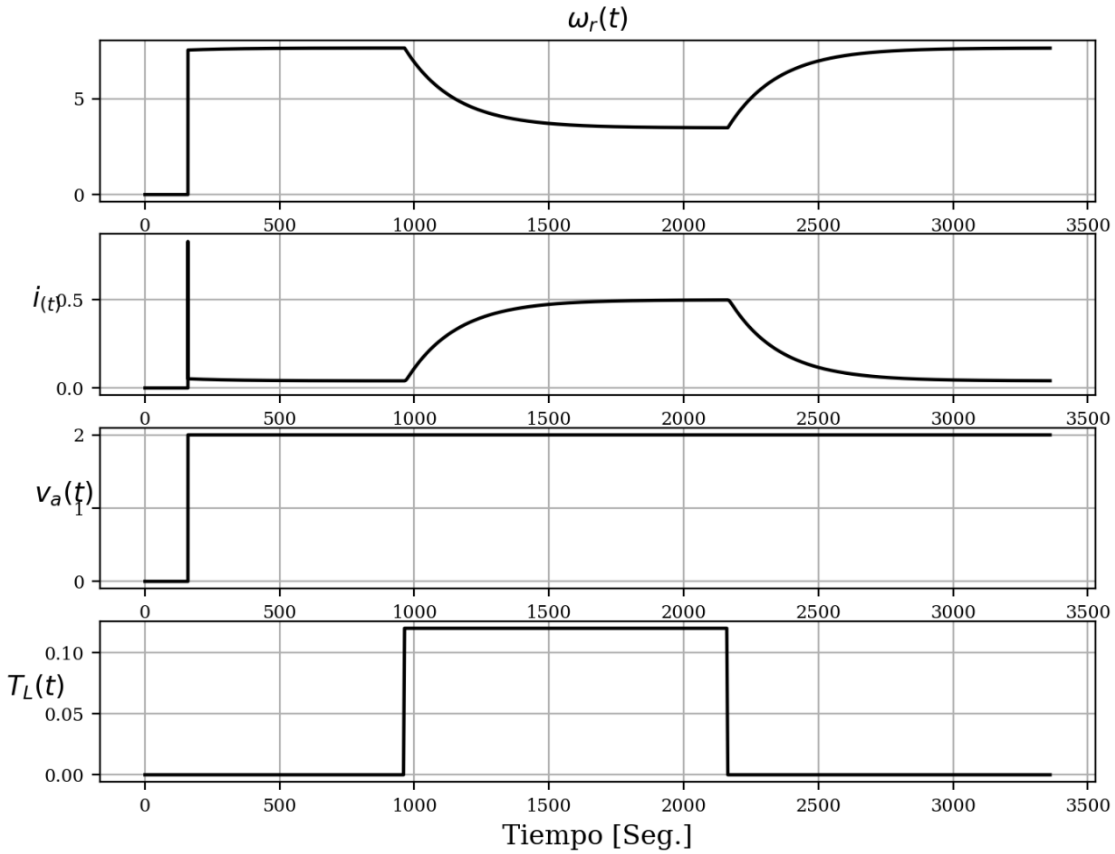

Fig. 1-3. Curvas de un motor CC para una entrada de 2V.

Ítem [7] Implementar un sistema en variables de estado que controle el ángulo del motor, para consignas de $\pi/2$ y $-\pi/2$ cambiando cada 2 segundos y que el T_L de 0,1 aparece sólo para $\pi/2$, para $-\pi/2$ es nulo. Hallar el valor de integración Euler adecuado. El objetivo es lograr la dinámica del controlador adecuada.

Ítem [8] Considerando que no puede medirse la corriente y sólo pueda medirse el ángulo, por lo que debe implementarse un observador. Obtener la simulación en las mismas condiciones que en el punto anterior, y superponer las gráficas para comparar.

Actividad Práctica Nº1 - Sistemas de Control II

Representación de sistemas y controladores

Gonzalez, Bruno

Parte I

Caso de estudio 1. Sistema de dos variables de estado

1. Ítem 1: Modelado y simulación del sistema RLC

Se asignaron los valores $R = 220\ \Omega$, $L = 500$ mH y $C = 2{,}2\ \mu$F para el circuito RLC. Se implementó una simulación en Python mediante el método de Euler, modelando el sistema en variables de estado:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t)$$

$$A = \begin{bmatrix} -\frac{R}{L} & -\frac{1}{L} \\ \frac{1}{C} & 0 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix}, \quad C_1 = \begin{bmatrix} R & 0 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad C_3 = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

Se utilizó una señal de entrada cuadrada de amplitud ±12V, con un período de 20ms, que cambia de signo cada 10ms. En la Figura 1 se muestran las respuestas obtenidas para la corriente, la tensión en el capacitor y la tensión sobre la resistencia.

```
# Parametros del circuito
R = 220          # Ohm
L = 0.5          # Henry
c = 2.2e-6       # Farad
############################################################
###          Modelado en variables de estado             ###
############################################################
# Varaibles de estado:
# x1 = i = Corriente de la malla
# x2 = Vc = Tension en el capacitor

# Matrices del sistema
A = np.array([[-R/L, -1/L],
              [1/c, 0]])
B = np.array([[1/L],
              [0]])
D= np.array([[0]])        # Matriz D = 0, ya que no hay transmision directa en el sistema

# Matriz C para la salida (Selector de salida)
C1 = np.array([[R, 0]])  # Matriz C para la salida =  Tension sobre el resistor
system1 = signal.StateSpace(A, B, C1, D)
C2 = np.array([[1, 0]])  # Selector de salida = Corriente
system2 = signal.StateSpace(A, B, C2, D)
C3 = np.array([[0, 1]])  # Selector de salida = Tension sobre el capacitor
```

```python
system3 = signal.StateSpace(A, B, C3, D)

############################################################
####              Simulacion del sistema                ###
############################################################
# Parametros del tiempo de simulacion
# Duracion total de la senal en s
total_duration = 0.100  # 100ms
dt = 10e-6  # Tiempo de muestreo (10 micros)
# Numero de pasos de simulacion
num_steps = int(total_duration / dt)
# Recalcular el tiempo para que coincida con num_steps
time = np.linspace(0, total_duration, num_steps)
############################################################
###                  Senal de entrada                   ###
############################################################
# Frecuencia y Duty Cycle de la senal en Hz
frequency_hz = 50
duty = 0.5
u_square = -12 * signal.square(2 * np.pi * frequency_hz * time, duty)
u = np.copy(u_square)
u[time < 0.01] = 0  # Los primeros 10ms (0.01s) se establecen en 0
# Inicializacion de variables
x = np.array([0, 0])  # Estado inicial [i(0), Vc(0)]
x_history = np.zeros((num_steps, 2))  # Historial de estados
y1 = np.zeros(num_steps)  # Salida del sistema 1 (VR)
y2 = np.zeros(num_steps)  # Salida del sistema 2 (i)
y3 = np.zeros(num_steps)  # Salida del sistema 3 (VC)
############################################################
###        Simulacion con Metodo de Euler               ###
############################################################

for k in range(num_steps):
    # Entrada en el instante actual
    u_k = u[k]
    # Derivada del estado: dx/dt = A*x + B*u
    dx_dt = A @ x + B.flatten() * u_k
    #B.flatten() convierte la matriz B (que es 2D) en un vector 1D para que pueda
    #multiplicarse correctamente con el escalar u_k.
    #Esto asegura que la operacion sea compatible en terminos de dimensiones.
    # Actualizacion del estado: x[k+1] = x[k] + dx/dt * dt
    x = x + dx_dt * dt
    # Guardar el estado y las salidas
    x_history[k, :] = x
    y1[k] = (C1 @ x).item()  # Extraer el valor escalar
    y2[k] = (C2 @ x).item()  # Extraer el valor escalar
    y3[k] = (C3 @ x).item()  # Extraer el valor escalar
```

Listing 1: Simulación del circuito RLC

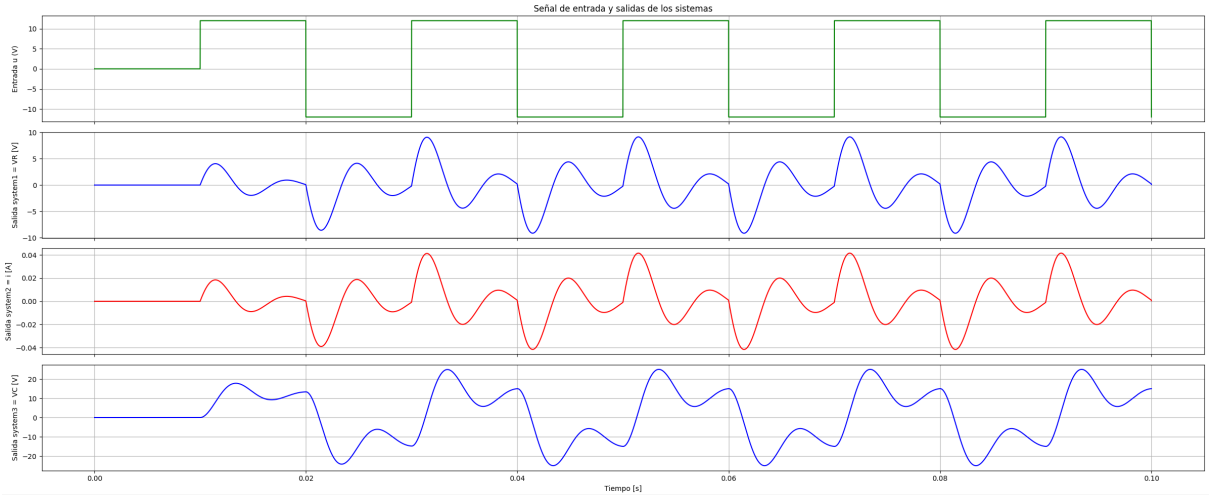

Figura 1: Simulación del circuito RLC con entrada cuadrada de 12V.

2. Ítem 2: Estimación de parámetros R, L y C

Se utilizaron los datos contenidos en el archivo `Curvas_Medidas_RLC_2025.xlsx` para modelar el sistema utilizando el método de Chen y a partir de allí estimar los parámetros del circuito con la respuesta al escalón.

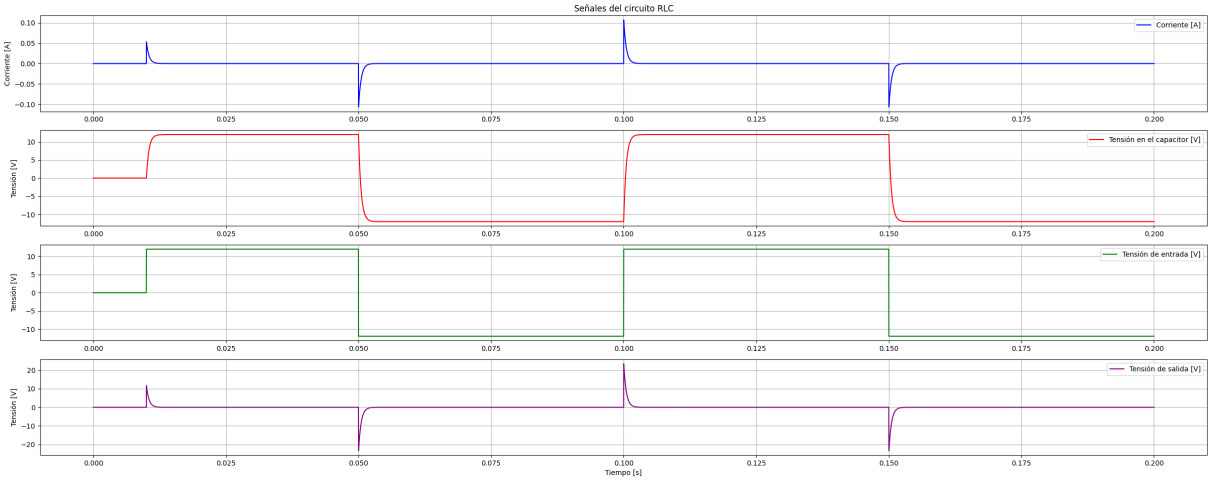

Figura 2: Curvas originales del circuito RLC extraídas del documento Curvas Medidas RLC 2025.xlsx

Una vez graficadas las curvas originales se procede a calcular la función de transferencia tomando como salida la tensión sobre el capacitor:

$$G(s) = \frac{1}{CLs^2 + CRs + 1}$$

Para reducir el número de incógnitas presentes en los parámetros del circuito se procede a calcular con ayuda de las gráficas medidas el valor del capacitor a partir de la derivada de la tensión sobre el capacitor y el valor de la corriente media entre los puntos de cálculo de la derivada:

```
#############################################################
###       Calculo de dVc/dt y I promedio                  ###
###         (entre 10ms y 10.8ms)                         ###
#############################################################
t1 = 0.0105
t2 = 0.0108
# Interpolacion para obtener Vc(t1), Vc(t2)
interp_vc = interp1d(tiempo, tension_capacitor, kind='linear')
vc_t1 = interp_vc(t1)
vc_t2 = interp_vc(t2)
# Regla del paralelogramo: pendiente
dvc_dt = (vc_t2 - vc_t1) / (t2 - t1)
# Corriente promedio entre 10.5ms y 10.8ms
mask_c = (tiempo >= t1) & (tiempo <= t2)
i_avg = np.mean(corriente[mask_c])
# Calculo de C
C_est = i_avg / dvc_dt
```

Listing 2: Cálculo del Capacitor

Resultando:

$$C = 2{,}2021\mu F$$

Continuando con el proceso de cálculo se modela al sistema a partir del método de Chen:

```
#########################################################
###           Metodo de Chen (10ms a 20ms)            ###
#########################################################
# Definir los instantes t1 y t2
t1_chen = 0.0103  # En segundos
t2_chen = 0.0106
t3_chen = 0.0109


# Interpolacion para obtener y_t1 y y_t2 directamente
interp_vc = interp1d(tiempo, tension_capacitor, kind='linear', fill_value="extrapolate")
interp_vin = interp1d(tiempo, tension_entrada, kind='linear', fill_value="extrapolate")

vc_t1 = interp_vc(t1_chen)
vc_t2 = interp_vc(t2_chen)
vc_t3 = interp_vc(t3_chen)
vin_t1 = interp_vin(t1_chen)
vin_t2 = interp_vin(t2_chen)
vin_t3 = interp_vin(t3_chen)


K = 12      # Ganancia del circuito

vc_t1 = interp_vc(t1_chen)/K
vc_t2 = interp_vc(t2_chen)/K
vc_t3 = interp_vc(t3_chen)/K

Kp=K/12


# Metodo de Chen
k1=(vc_t1/Kp)-1
k2=(vc_t2/Kp)-1
k3=(vc_t3/Kp)-1

b=4*k1**3*k3-3*k1**2*k2**2-4*k2**3+k3**2+6*k1*k2*k3
alpha_1=(k1*k2+k3-np.sqrt(b))/(2*(k1**2+k2))
alpha_2=(k1*k2+k3+np.sqrt(b))/(2*(k1**2+k2))
beta=(2*k1**3+3*k1*k2+k3-np.sqrt(b))/(np.sqrt(b))

# Origen del escalon en 10ms; normalizacion de la escala de tiempo:
t1_chen = t1_chen - 0.010

T_1=-t1_chen/np.log(alpha_1)
T_2=-t1_chen/np.log(alpha_2)
T_3 = beta*(T_1-T_2)+T_1
```

Listing 3: Modelado con método de Chen

Finalmente, igualando la función de transferencia teórica con la encontrada a partir del método de Chen y utilizando como dato el valor del capacitor ya calculado se despeja y calculan los parámetros del circuito restantes:

```
# Calcular R y L con C estimada
L_est = (T_1 * T_2)/C_est
R_est = (T_1 + T_2)/C_est
```

Listing 4: Cálculo de R y L

Resultando los siguientes parámetros:

- Capacitancia: $C = 2{,}2021\ \mu\text{F}$ (a partir de $i = C \cdot dV_c/dt$)
- Inductancia: $L = 2{,}4295$ mHy
- Resistencia: $R = 223{,}77\ \Omega$

3. Ítem 3: Validación del modelo

Con los parámetros estimados, se simuló la respuesta del sistema a la entrada medida y se comparó con la tensión experimental sobre el capacitor.

```
##############################################################
###   Comparacion: Tension Calculada vs Experimental ###
##############################################################
from scipy.signal import lti, lsim

# Crear el sistema LTI a partir de los valores estimados
numerador = [1]  # Numerador de H(s) = 1 / (L C s^2 + R C s + 1)
denominador = [L_est * C_est, R_est * C_est, 1]  # Denominador de H(s)
sistema = lti(numerador, denominador)

# Simular la respuesta del sistema con la tension de entrada experimental
t_sim, vc_calculada, _ = lsim(sistema, tension_entrada, tiempo)
# figura con dos subplots
plt.figure(figsize=(12, 8))
# Subplot 1: Tension de entrada
plt.subplot(2, 1, 1)
plt.plot(tiempo, tension_entrada, label="Tension de entrada", color='green', linewidth=2)
plt.title("Tension de Entrada")
plt.xlabel("Tiempo [s]")
plt.ylabel("Tension [V]")
plt.grid()
plt.legend()
# Subplot 2: Comparacion de la tension sobre el capacitor
plt.subplot(2, 1, 2)
plt.plot(tiempo, tension_capacitor, label="Tension experimental (Excel)", color='red',
    linestyle='--', linewidth=3)
plt.plot(t_sim, vc_calculada, label="Tension calculada (Metodo de Chen)", color='blue')
plt.title("Comparacion: Tension sobre el Capacitor")
plt.xlabel("Tiempo [s]")
plt.ylabel("Tension [V]")
plt.grid()
plt.legend()
```

Listing 5: Comparación entre sistema original y sistema calculado

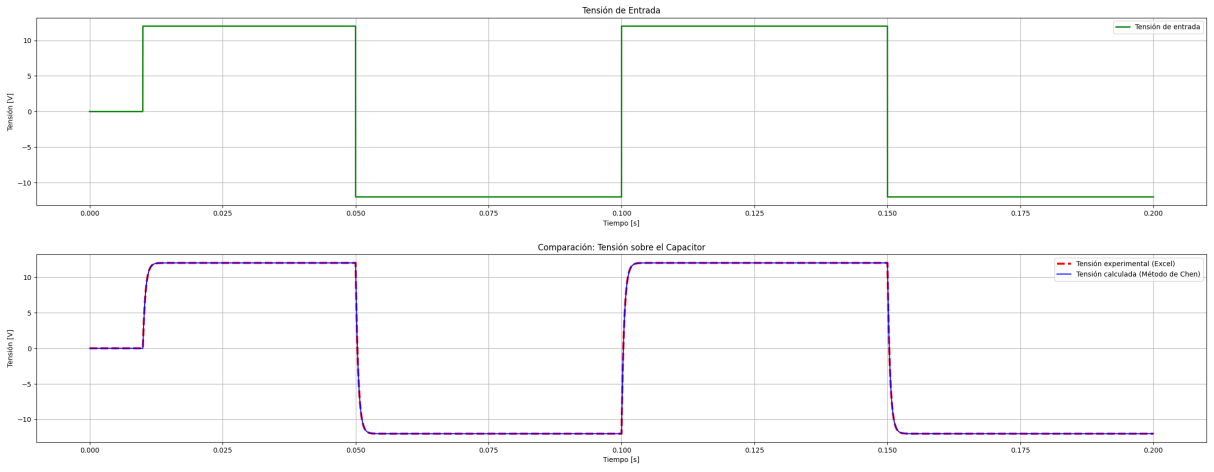

Figura 3: Comparación entre la tensión calculada (modelo) y la medida (experimental).

En la Figura 3 se observa una buena superposición de las curvas, validando el modelo identificado y sus parámetros calculados.

Parte II

Caso de estudio 2. Sistema de tres variables de estado

4. Ítem 4: Simulación del motor de corriente continua

En este ítem se simula un motor de corriente continua a partir de sus ecuaciones dinámicas, utilizando el método de Euler con un paso de integración $\Delta t = 10^{-7}\ s$, durante 5 segundos de operación, con una entrada constante de 12V.

Los parámetros utilizados del sistema son:

- $L_{AA} = 366\ \mu H$
- $J = 5 \cdot 10^{-9}\ kg \cdot m^2$
- $R_A = 55{,}6\ \Omega$
- $B = 0$ (sin fricción viscosa)
- $K_i = 6{,}49 \cdot 10^{-3}\ Nm/A$
- $K_m = 6{,}53 \cdot 10^{-3}\ V \cdot s/rad$
- $V_A = 12\ V$

El modelo empleado es el siguiente:

$$\begin{aligned} \frac{di_a}{dt} &= \frac{-R_A i_a - K_m \omega + V_A}{L_{AA}} \\ \frac{d\omega}{dt} &= \frac{K_i i_a - B\omega - T_L}{J} \\ \frac{d\theta}{dt} &= \omega \end{aligned}$$

El siguiente código implementa la integración de estas ecuaciones utilizando el método de Euler:

```
# Metodo de Euler
for k in range(N - 1):
    di_a = (-R_A * i_a[k] - K_m * omega[k] + V_A) / L_AA
    domega = (K_i * i_a[k] - B_m * omega[k] - T_L) / J
    dtheta = omega[k]

    i_a[k + 1] = i_a[k] + dt * di_a
    omega[k + 1] = omega[k] + dt * domega
    theta[k + 1] = theta[k] + dt * dtheta
```

Listing 6: Simulación del motor CC mediante método de Euler

La Figura 4 muestra evolución de corriente de armadura, velocidad angular y posición angular en el tiempo para una tensión de entrada de 12V.

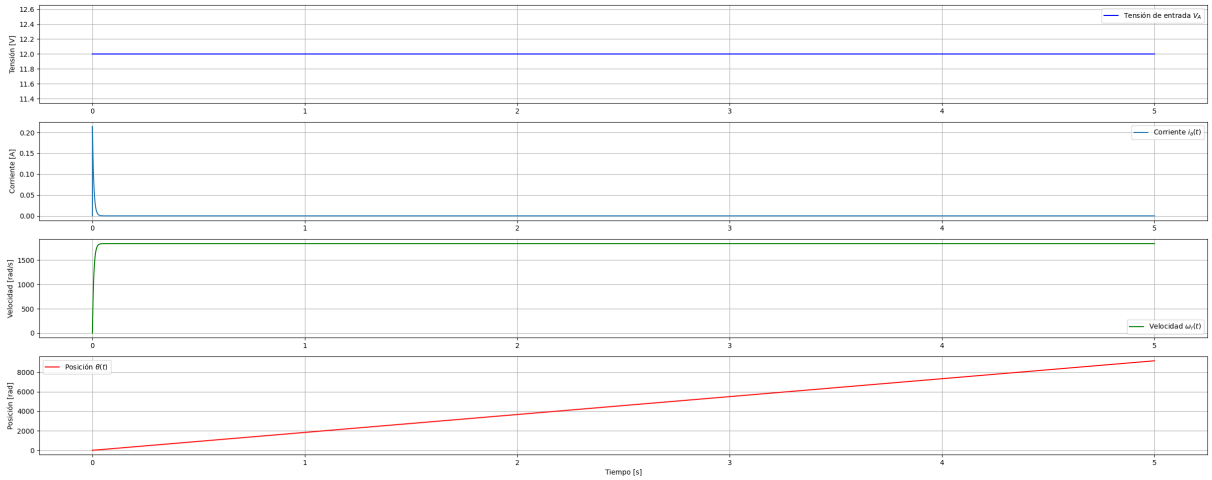

Figura 4: Simulación del motor CC para una entrada de 12V.

Como resultado de la simulación, se observa el comportamiento típico de arranque del motor: una corriente de armadura que alcanza un pico inicial de aproximadamente 0,21 A, producto de la acción de la inductancia frente a un cambio brusco de tensión, y una velocidad angular que crece progresivamente hasta alcanzar un régimen permanente.

5. Ítem 5: Identificación del modelo del motor y validación

El desarrollo de este ítem se llevó a cabo mediante un script en Python (ver archivo `item5.py`) que procesa los datos experimentales provistos en el archivo `Curvas_Medidas_Motor_2025.xls`

Visualización de señales medidas

Inicialmente se realiza el ploteo de las señales de tensión, corriente, velocidad angular y torque registradas experimentalmente:

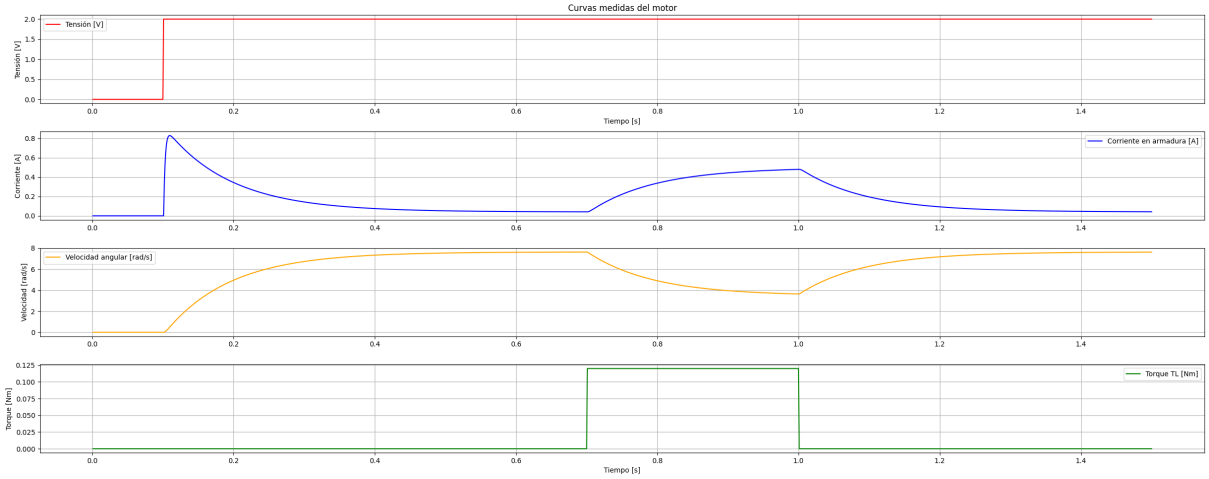

Figura 5: Curvas medidas del motor: tensión, corriente de armadura, velocidad angular y torque.

Modelo teórico del sistema y funciones de transferencia

Partiendo de las ecuaciones dinámicas del motor, se derivaron las funciones de transferencia teóricas:

$$\frac{\omega(s)}{V_a(s)} = \frac{K_i}{J_m L_a s^2 + (B_m L_a + R_a J_m)s + (B_m R_a + K_i K_m)}$$

$$\frac{\omega(s)}{T_L(s)} = \frac{(L_a/R_a)s + 1}{R_a J_m L_a s^2 + (B_m L_a + R_a J_m)s + (B_m R_a + K_i K_m)}$$

Aplicación del método de Chen para estimar modelo

Para obtener el modelo del sistema se aplicó el método de chen con el objetivo de encontrar las funciones de transferencia que toman como salida a la velocidad angular y como entrada a la tensión de alimentación y al torque TL. El procedimiento consiste en analizar la respuesta temporal de la velocidad angular ante escalones de entrada (en tensión o torque) y obtener tres puntos característicos de la dinámica del sistema. A partir de los valores de salida en esos instantes, se calculan los parámetros del método, con los que se estima la respuesta. Esto permite e obtener las funciones de transferencia modelo.

Cálculo de parámetros para $\frac{\omega(s)}{V_a(s)}$

```
##############################################
# Aplicaci n pr ctica del M todo de Chen #
##############################################
#########        wr/VA:       ###############
# Definir los instantes t1 y t2
t1_chen = 0.102  # En segundos
t2_chen = 0.103
t3_chen = 0.104

# Interpolacion para obtener y_t1 y y_t2 directamente
interp_wr = interp1d(tiempo, Velocidad_angular, kind='linear', fill_value="extrapolate")
interp_va = interp1d(tiempo, Tensi n, kind='linear', fill_value="extrapolate")

wr_t1 = interp_wr(t1_chen)
wr_t2 = interp_wr(t2_chen)
wr_t3 = interp_wr(t3_chen)

va_t1 = interp_va(t1_chen)
va_t2 = interp_va(t2_chen)
va_t3 = interp_va(t3_chen)

K = 7.6246
K1C = K / 2  # Ganancia del sistema

#Parametros del Metodo de Chen:
k1=(wr_t1/K1C)-1
k2=(wr_t2/K1C)-1
k3=(wr_t3/K1C)-1

be =  4 * (k1**3) * k3 - 3 * (k1**2) * (k2**2) - 4 * (k2**3) + (k3**2) + 6 * k1 * k2 * k3

if be < 0:
    alfa1 = (k1 * k2 + k3 - np.sqrt(be)) / (2 * (k1**2 + k2))
    alfa2 = (k1 * k2 + k3 + np.sqrt(be)) / (2 * (k1**2 + k2))
else:
    alfa1 = (k1 * k2 + k3 - np.sqrt(be)) / (2 * (k1**2 + k2))
```

```
    alfa2 = (k1 * k2 + k3 + np.sqrt(be)) / (2 * (k1**2 + k2))
beta = (2 * k1**3 + 3 * k1 * k2 + k3 - np.sqrt(be)) / np.sqrt(be)

t1_chen_n = t1_chen - 0.100 #normalizacion de la escala de tiempo.

T1 = np.real(-t1_chen_n / np.log(alfa1))
T2 = np.real(-t1_chen_n / np.log(alfa2))
T3 = np.real( beta * (T1 - T2) + T1)
```

Listing 7: Modelo Wr/Va ajustado con el método de Chen

Dando como resultado la siguiente función de transferencia:

Función de transferencia FT:

$$\frac{3.8123}{(0.00468923961841731 \cdot s + 1) \cdot (0.0885344365467611 \cdot s + 1)}$$

Simulando la función de transferencia obtenida y comparándola con la respuesta original del motor frente al escalón de tensión se observa:

(Se debe ignorar la presencia de una perturbación debido al torque de carga en la curva original.

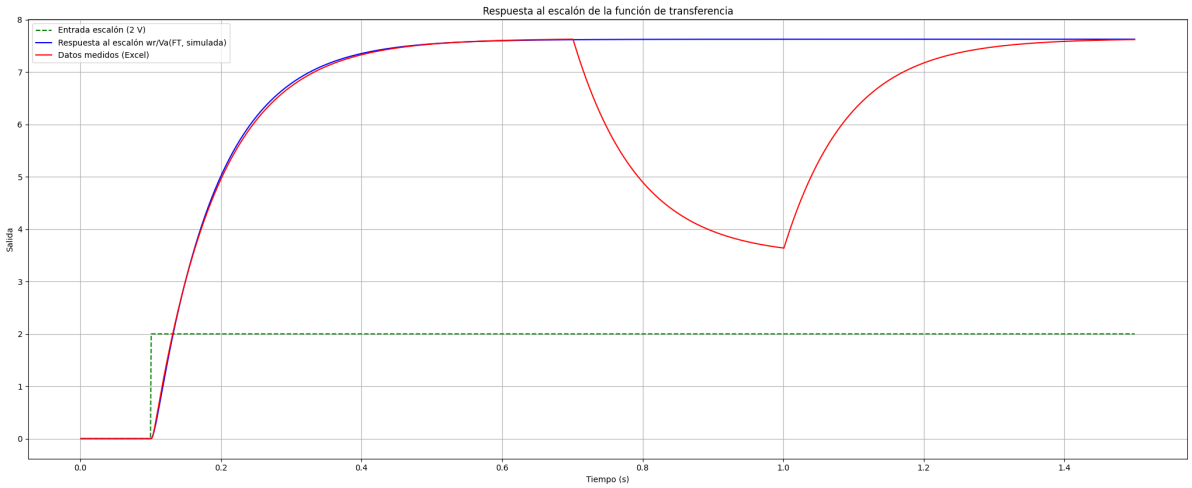

Figura 6: Respuesta al escalón de la respuesta encontrada comparada con la curva original de velocidad angular

Cálculo de parámetros para $\frac{\omega(s)}{T_L(s)}$

```
##############################################
# Aplicaci n pr ctica del M todo de Chen #
##############################################
#########        wr/TL:        ###############

t1_chen = 0.702  # En segundos
t2_chen = 0.712
t3_chen = 0.722

# Interpolaci n para obtener y_t1 y y_t2 directamente
interp_wr = interp1d(tiempo, Velocidad_angular, kind='linear', fill_value="extrapolate")
interp_TL = interp1d(tiempo, torque, kind='linear', fill_value="extrapolate")

wr_t1 = -(interp_wr(t1_chen)-7.6245)
wr_t2 = -(interp_wr(t2_chen)-7.6245)
```

```
wr_t3 = -(interp_wr(t3_chen)-7.6245)
TL_t1 = interp_TL(t1_chen)
TL_t2 = interp_TL(t2_chen)
TL_t3 = interp_TL(t3_chen)

K2 = 0.12
K2C = (7.6245 - 3.6379)/K2 # Ganancia del circuito para wr/TL

# Par metros del m todo de Chen:
k1=(1/K2)*(wr_t1/K2C)-1
k2=(1/K2)*(wr_t2/K2C)-1
k3=(1/K2)*(wr_t3/K2C)-1

be =  4 * k1**3 * k3 - 3 * k1**2 * k2**2 - 4 * k2**3 + k3**2 + 6 * k1 * k2 * k3

if be > 0:
    alfa1 = (k1 * k2 + k3 - np.sqrt(be)) / (2 * (k1**2 + k2))
    alfa2 = (k1 * k2 + k3 + np.sqrt(be)) / (2 * (k1**2 + k2))
else:
    alfa1 = (k1 * k2 + k3 - np.sqrt(be)) / (2 * (k1**2 + k2))
    alfa2 = (k1 * k2 + k3 + np.sqrt(be)) / (2 * (k1**2 + k2))
beta = (2 * k1**3 + 3 * k1 * k2 + k3 - np.sqrt(be)) / np.sqrt(be)

t1_chen_n = t1_chen - 0.701                    # Normalizacion de la escala de tiempo.

T_1p=np.real(-t1_chen_n/np.log(alfa1))
T_2p=np.real(-t1_chen_n/np.log(alfa2))
T_3p =np.real( beta*(T_1p-T_2p)+T_1p)
```

Listing 8: Modelo Wr/TL ajustado con el método de Chen

Dando como resultado la siguiente función de transferencia:

Función de transferencia wr/TL:

$$\frac{0.640991940570602 \cdot s + 33.2216666666667}{(0.00468923961841731 \cdot s + 1) \cdot (0.0885344365467611 \cdot s + 1)}$$

Cabe destacar que del algoritmo de Chen en este caso solamente se extrajo la ganancia, manteniendo los parámetros T1 y T2 que dan origen a la ecuación caracterítica desde la respuesta $\frac{\omega(s)}{V_a(s)}$

Simulando la función de transferencia obtenida y comparándola con la respuesta original del motor frente al escalón de tensión se observa:

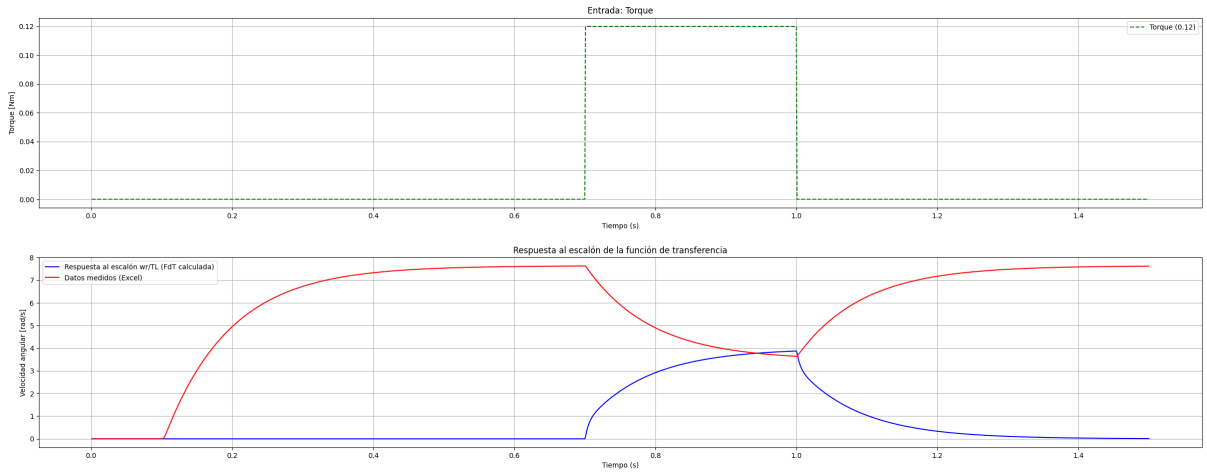

Figura 7: Respuesta al escalón de la respuesta encontrada comparada con la curva original de velocidad angular

La pendiente inversa con respecto a la curva original se debe al ajuste realizado en el algoritmo de Chen para captar solamente el efecto del torque.

Validación del modelo obtenido por el método de Chen

A continuación se simulan en conjunto las respuestas obtenidas permitiendo una comparación total con las curvas originales:

```
# Simulaci n de la respuesta de FT y FT_1 usando torque como entrada

# Simulaci n de FT con entrada de tensi n
t_FT, y_FT, _ = lsim(FT_lti, U=Tensi n, T=tiempo, X0=[0, 0])

# Simulaci n de FT_1 con entrada de torque
t_FT1, y_FT1, _ = lsim(FT_1_lti, U=torque, T=tiempo, X0=[0, 0])

# Modelo combinado: suma de ambas respuestas
y_modelo = y_FT - y_FT1

plt.figure(figsize=(12, 10))


# Subplot 1: FT wr/Va
plt.subplot(3, 1, 1)
plt.plot(tiempo, y_FT, '-', color='green', label='FT wr/Va')
plt.ylabel('Velocidad [rad/s]')
plt.title('FT wr/Va - Respuesta al escal n para escal n de 2V')
plt.grid()
plt.legend()

# Subplot 2: FT_1 wr/TL
plt.subplot(3, 1, 2)
plt.plot(tiempo, -y_FT1, '-', color='orange', label='FT wr/TL')
plt.xlabel('Tiempo [s]')
plt.ylabel('Velocidad [rad/s]')
plt.title('FT wr/TL - Respuesta al escal n para escal n de 0.12Nm')
plt.grid()
plt.legend()

# Subplot 3: Velocidad angular medida (Excel) y Modelo combinado superpuestos
plt.subplot(3, 1, 3)
plt.plot(tiempo, Velocidad_angular, color='red', label='Velocidad angular (Datos Excel)')
plt.plot(tiempo, y_modelo, color='blue', label='Modelo (FT wr/Va + FT wr/TL)')
plt.ylabel('Velocidad [rad/s]')
plt.title('Comparaci n: Curva dato vs Modelo')
plt.grid()
plt.legend()


plt.tight_layout()
plt.show()
```

Listing 9: Simulación de la respuesta obtenida con el método de Chen

Al simularlo resultan las siguientes curvas:

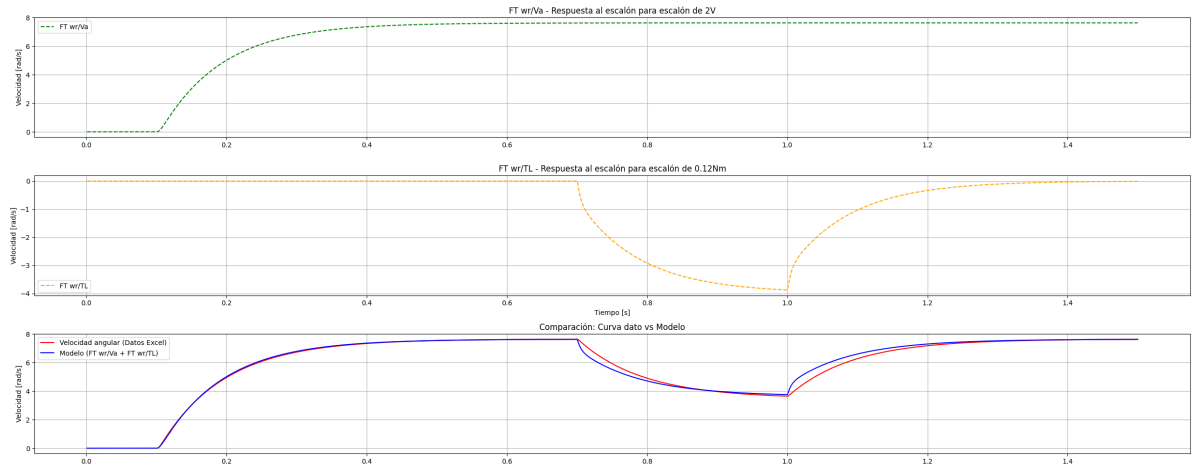

Figura 8: Comparación entre las curvas experimentales y los modelos ajustados con el método de Chen.

Se observa cierta discrepancia en la respuesta de la velocidad angular frente al torque, lo cual en la sección siguiente puede producir una estimación errónea de los parámetros del motor.

Estimación de parámetros físicos del motor

Comparando las funciones de transferencia teóricas con las obtenidas por Chen, se estimaron los parámetros físicos del sistema. Para reducir el número de incógnitas, primero se calculó el valor de la resistencia de armadura a partir de los valores de tensión y corriente encontrados en las curvas medidas.

```
#Calculo de Ra:
pico_corriente = Corriente_en_armadura.max()
max_tension = Tension.max()
Ra_est = max_tension / pico_corriente
```

Listing 10: Cálculo de R_a

```
Ra_est = Ra_est
La_est = Ra_est*T_3p
Ki_est = (K1C)/(K2C*Ra_est)
Jm_est = (T1*T2)/(K2C*Ra_est**2*T_3p)
Km_est = ((T1-T_3p)*(T2-T_3p))/(K1C*T_3p**2)
Bm_est = ((-T1*T2+T1*T_3p+T2*T_3p))/(K2C*Ra_est**2*T_3p**2)
```

Listing 11: Cálculo de parámetros físicos estimados

```
===== COMPONENTES ESTIMADOS =====
Ra estimada = 2.41
La estimada = 0.0466
Ki estimada = 0.0475
Km estimada = -0.71
Jm estimada = 0.0001
Bm estimada = 0.0192
```

Validación con funciones de transferencia teóricas

Se construyeron las funciones de transferencia teóricas utilizando los parámetros estimados y se simuló la respuesta completa:

```
Wr_Va_est= (Ki_est/(Jm_est*La_est)) / (s**2 + ((Bm_est*La_est+Jm_est*Ra_est)/(Jm_est*
    La_est))*s + ((Bm_est*Ra_est+Ki_est*Km_est)/(Jm_est*La_est)))

Wr_TL_est=(((La_est/Ra_est)*s+1)/(Ra_est*Jm_est*La_est)) / (s**2 + ((Bm_est*La_est+
    Jm_est*Ra_est)/(Jm_est*La_est))*s + ((Bm_est*Ra_est + Ki_est*Km_est)/(Jm*La)))

# Definir los coeficientes de las funciones de transferencia Wr_Va_est y Wr_TL_est
# Wr_Va_est = (Ki_est / (Jm_est * La_est)) / (s^2 + ((Bm_est * La_est + Jm_est * Ra_est)
    / (Jm_est * La_est)) * s + ((Bm_est * Ra_est + Ki_est * Km_est) / (Jm_est * La_est)))
num_Wr_Va = [Ki_est / (Jm_est * La_est)]
den_Wr_Va = [1, (Bm_est * La_est + Jm_est * Ra_est) / (Jm_est * La_est), (Bm_est * Ra_est
    + Ki_est * Km_est) / (Jm_est * La_est)]

# Wr_TL_est = ((La_est / Ra_est) * s + 1) / (Ra_est * Jm_est * La_est) / (s^2 + ((Bm_est
    * La_est + Jm_est * Ra_est) / (Jm_est * La_est)) * s + ((Bm_est * Ra_est + Ki_est *
    Km_est) / (Jm_est * La_est)))
num_Wr_TL = [(La_est / Ra_est)/(Jm_est*La_est*Ra_est), 1/(Jm_est*La_est*Ra_est)]
den_Wr_TL = [1, (Bm_est * La_est + Jm_est * Ra_est) / (Jm_est * La_est), (Bm_est * Ra_est
    + Ki_est * Km_est) / (Jm_est * La_est)]

# Crear sistemas LTI para ambas funciones de transferencia
system_Wr_Va = lti(num_Wr_Va, den_Wr_Va)
system_Wr_TL = lti(num_Wr_TL, den_Wr_TL)
print("\nFuncion de transferencia te rica Wr_Va con parametros estimados:")
print(num_Wr_Va)
print(den_Wr_Va)
print("\nFuncion de transferencia te rica Wr_TL con parametros estimados:")
print(num_Wr_TL)
print(den_Wr_TL)

# Simular la respuesta de Wr_Va_est con la entrada Tension
_, y_Wr_Va, _ = lsim(system_Wr_Va, Tensi n, tiempo) # Usar directamente las se ales
    del archivo

# Simular la respuesta de Wr_TL_est con la entrada torque
_, y_Wr_TL, _ = lsim(system_Wr_TL, torque, tiempo) # Usar directamente las se ales del
    archivo
```

Listing 12: Simulación del modelo combinado

Resultando las siguientes funciones de transferencia (se plotea numerador y denominador) y sus correspondientes curvas:

```
Función de transferencia teórica Wr_Va con parámetros estimados:
[9182.742705086665]
[1, 224.54923071299572, 2408.7146093137135]

Función de transferencia teórica Wr_TL con parámetros estimados:
[1543.9666517047565, 80021.5138457504]
[1, 224.54923071299572, 2408.7146093137135]
```

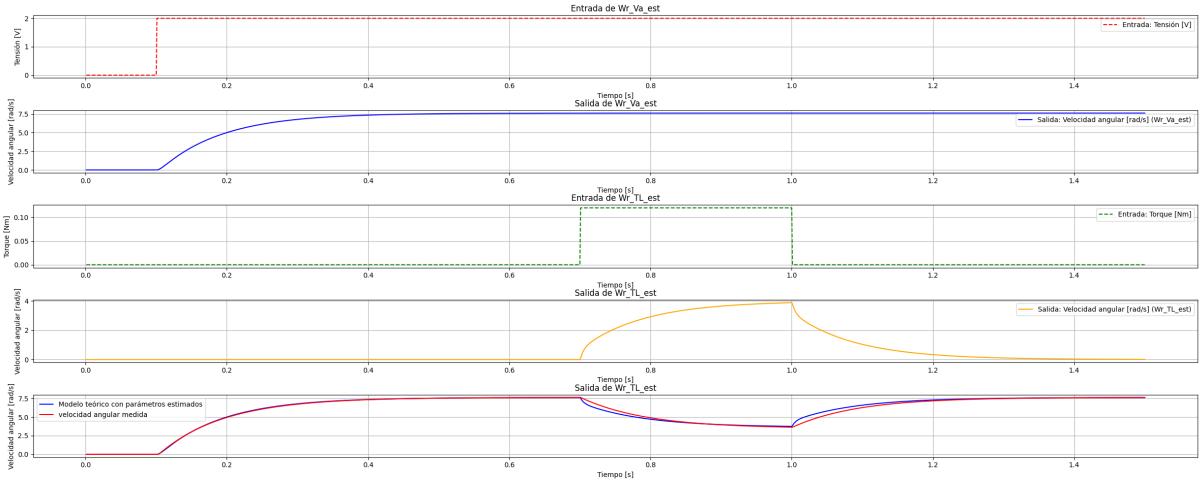

Figura 9: Comparación entre la respuesta teórica utilizando los parámetros físicos del motor estimados y la medición original

Al igual que lo observado en la figura 8 se observa cierta discrepancia entre el modelo encontrado y la curva original cuando se presenta el torque de carga.

Lecciones aprendidas y evaluación de logros

Durante el desarrollo de la Actividad Práctica Nº1 se consolidaron conocimientos clave en modelado, simulación y análisis de sistemas dinámicos. Se profundizó especialmente en el uso de representaciones en espacio de estados, identificación de parámetros mediante la respuesta al escalón y validación experimental de modelos a partir de datos medidos. A continuación, se resumen las principales lecciones aprendidas:

- **Comprensión del comportamiento dinámico de sistemas físicos:** a partir del modelado del circuito RLC y del motor de corriente continua, se reforzó la capacidad de interpretar físicamente las ecuaciones diferenciales asociadas a sistemas reales y de traducirlas a representaciones matemáticas y computacionales.
- **Aplicación del método de Chen para identificación de sistemas:** se adquirió experiencia práctica en la implementación de este método, validando sus resultados mediante comparación directa con datos experimentales tanto en sistemas eléctricos como electromecánicos.
- **Simulación numérica mediante el método de Euler:** se afianzaron habilidades en la implementación de integradores numéricos, adecuando el paso temporal según el comportamiento dinámico del sistema y comprendiendo sus limitaciones.
- **Evaluación crítica del modelo:** se identificaron discrepancias entre el modelo simulado y las mediciones experimentales, especialmente bajo perturbaciones externas como el torque de carga, lo cual permitió reflexionar sobre la robustez y los límites de los modelos obtenidos.

6. Enlaces y recursos

- Repositorio GitHub del estudiante: `https://github.com/BrunoUNC/Sistemas-De-Control-II-`
  `git`