Universidad Nacional de Córdoba
Facultad de Ciencias Exactas, Físicas y Naturales
Escuela de Ingeniería Electrónica

Informe Actividad Práctica N°1

Trabajo Práctico Sistemas de Control II

Autor: Villar, Federico Ignacio
Profesor: Pucheta, Julián
Fecha de entrega: 2 de mayo 2023
Córdoba, Argentina

Resumen

En el siguiente documento se detalla la resolución de la primera actividad práctica propuesta por el profesor Julián Pucheta en la materia de Sistemas de Control II. Se desarrollarán los procedimientos utilizados, así como también se adjuntarán los diferentes códigos de Python y Matlab utilizados para la compleción de la actividad.

Al final del informe, se añaden algunos enlaces de utilidad para la investigación de fuentes y futuras lecturas de este informe como bibliografía. Se trata de que las conclusiones sean cortas, concisas y representativas del trabajo realizado.

Todas las gráficas de funciones aduntas en el documento son de elaboración propia haciendo uso de código en los lenguajes antes mencionados. Cabe destacar que las herramientas de software utilizadas fueron:

- Visual Studio Code
- Matlab
- GitHub Desktop

Todo este trabajo se encuentra publicado en un repositorio de GitHub y mantenido por el alumno autor del informe. Se adjunta junto con los links anteriormente mencionados, el enlace del repositorio online. Además, en la entrega se adjuntan los códigos utilizados en formatos: **.m**, **.ipynb** y **.mlx**, con el archivo de Live Script también exportado en formato **.pdf**.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Índice de Códigos

1. Consigna

1.1. Caso de estudio 1: sistema de dos variables de estado

Sea el circuito eléctrico de la figura, con las representaciones en variables de estado

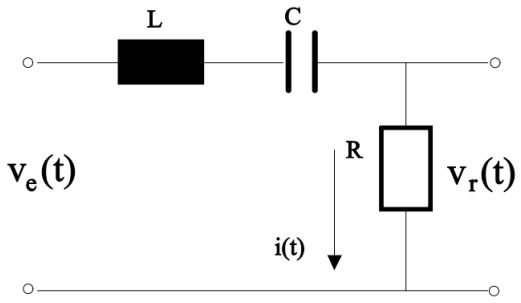

Figura 1: Circuito RLC.

$$\dot{x} = Ax(t) + bu(t)$$
$$y = c^T x(t)$$

donde las matrices contienen a los coeficientes del circuito,

$$A = \begin{bmatrix} -R/L & -1/L \\ 1/C & 0 \end{bmatrix}, \; b = \begin{bmatrix} 1/L \\ 0 \end{bmatrix}, \; c^T = \begin{bmatrix} R & 0 \end{bmatrix}$$

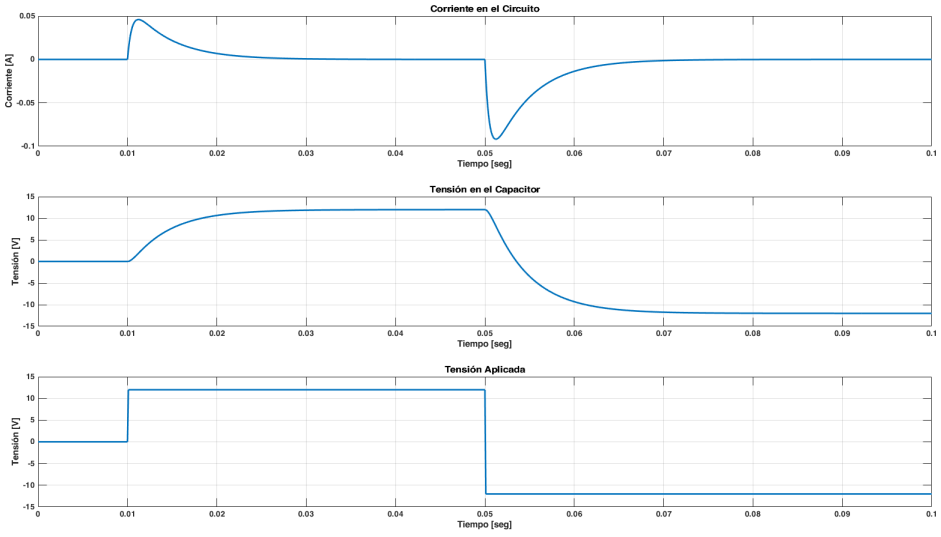

Figura 2: Curvas del circuito RLC para una entrada de 12V.

1. Asignar valores a $R = 4.7k\Omega$, $L = 10\mu Hy$, y $C = 100nF$. Obtener simulaciones que permitan estudiar la dinámica del sistema, con una entrada de tensión escalón de 12V, que cada $1ms$ cambia de signo.

2. Asignar valores a $R = 5.6k\Omega$, $L = 10\mu Hy$, y $C = 100nF$; repetir lo anterior para comparar el resultado y verificar la correcta simulación.

3. En el archivo $CurvasMedidasRLC.xls$ encontrarán las series de datos que deberían emplear 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.

4. Una vez determinados los parámetros R, L y C, emplear la serie de corrinte desde 0.05 seg en adelante para validar el resultado.

1.2. Caso de estudio 2: sistema de tres variables de estado

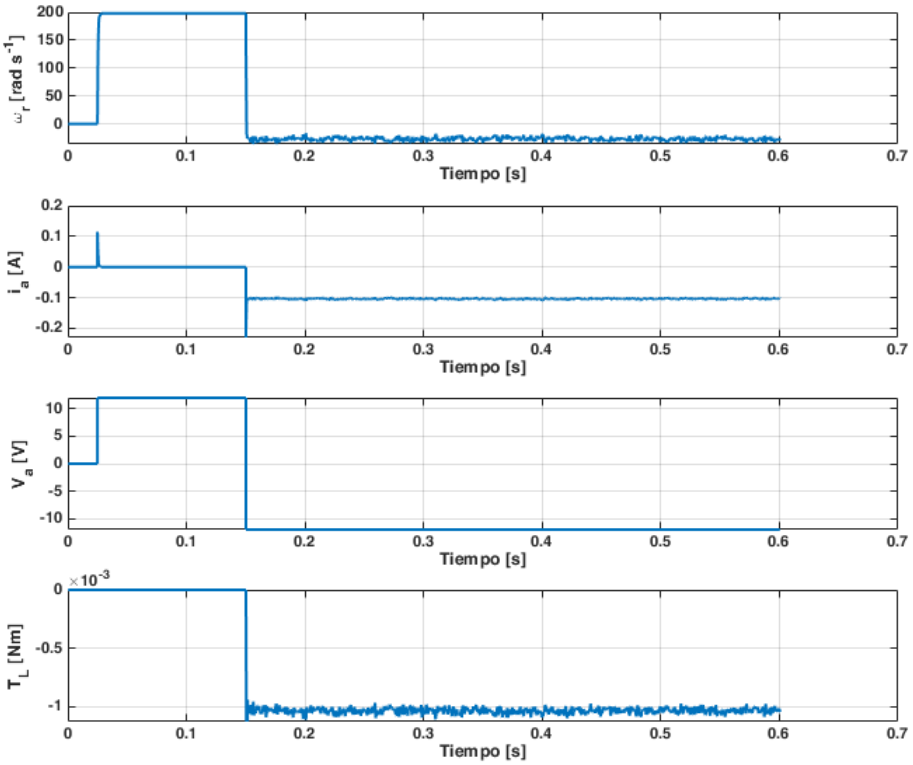

Figura 3: Curvas de un motor CC para una entrada de 12V.

Dadas las ecuaciones del motor de corriente continua con torque de carga T_L no nulo, con los parámetros: $L_{AA} = 366\ 10^{-6}$; $J = 5\ 10^{-9}$; $R_A = 55.6$; $B = 0$, $K_i = 6.49\ 10^{-3}$; $K_m = 6.53\ 10^{-3}$:

$$\frac{di_a}{dt} = -\frac{R_A}{L_{AA}} i_a - \frac{K_m}{L_{AA}} \omega_r + \frac{1}{L_A A} v_a$$

$$\frac{d\omega_r}{dt} = \frac{K_i}{J} i_a - \frac{B_m}{J}\omega_r - \frac{1}{J}T_L$$
$$\frac{d\theta_t}{dt} = \omega_r$$

Implementar un algoritmo de simulación para inferir el comportamiento de las variables de interés meidante integración Euler con $\Delta t = 10^{-7}$ segundos para:

1. Obtener el torque máximo que puede soportar el motor modelado mediante las ecuaciones diferenciales cuando se lo alimenta con $12V$, graficando para 5 segundos de tiempo la velocidad angular y corriente i_a.

2. Mostrar simulaciones de 5 segundos que permitan observar la correinte i_a en todo momento y establecer su valor máximo como para dimensionar dispositivos electrónicos.

3. A partir de las curvas de mediciones de las variables graficadas en la figura 3, se requiere obtener el modelo del sistema considerando como entrada un escalón de $12V$, como salida a la velocidad angular, y a partir de 0.1 segundos se aplica un T_L aproximado de 1.04 10^{-3} Nm. En el archivo $CurvasMedidasMotor.xls$ 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 de la corriente.

4. Implementar un PID en tiempo discreto para que el ángulo del motor permanezca en una referencia de 1 rad. (Tip: partir de $K_P = 0.1$; $K_i = 0.01$; $K_D = 5$).

2. Resolución del Caso 1

Este caso está resuelto enteramente haciendo uso de Python, es por eso que se adjuntan para la obtensión de las diferentes señales y/o respuestas los diferentes códigos usados. Las librerías a utilizar se importan de la siguiente manera:

Código 1: Librerías utilizadas.

```
from IPython.display import Image
from scipy import signal
from control.matlab import *
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sympy as sym
```

2.1. Simulaciones

Se comienza con la definición de los siguientes valores para los componentes del circuito:

- $R = 4.7k\Omega$
- $L = 10\mu Hy$
- $C = 100nF$

Luego, siguiendo los lineamientos de la consigna, se debe obtener la señal de tensión alternada solicitada, que es la siguiente:

Código 2: Tensión en el circuito.

```
t = np.linspace(0, 0.01, 10000)
u = np.piecewise(t, [t<=0.001, t>0.001], [lambda t: 0*t, lambda t: 12*signal.square(2 * np.pi * 500
  ↪ * (t - 0.001))])
plt.plot(t,u)
plt.title('Tensión aplicada al circuito')
plt.xlabel('Tiempo [seg]')
plt.ylabel('$V_e(t) [V]$')
plt.grid()
```

La señal obtenida con el código anterior se adjunta en la figura 4. Para obtenerla se utilizó el comando **lambda** de Python, que permite declarar una señal por partes. De esa forma, se planteó una señal retardada cuadrada. Ahora, para obtener la señal cuadrada se utilizó el comando **square** del paquete **Numpy**. Para lograr el efecto periódico deseado, simplemente se le pasa como argumento la velocidad angular requerida.

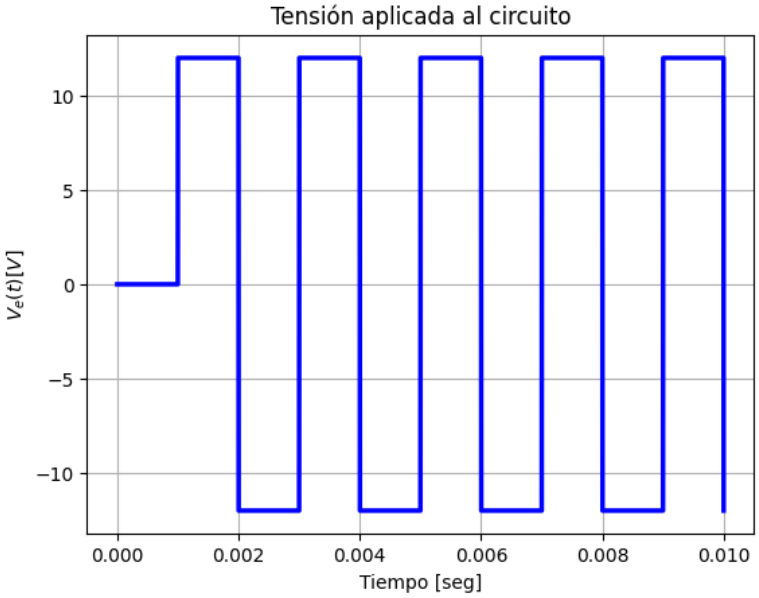

Figura 4: Señal de tensión aplicada al circuito.

Luego, para obtener la respuesta del circuito, se hace uso de los comandos que permiten trabajar con el modelado en espacio de estados. Se obtiene el siguiente resultado:

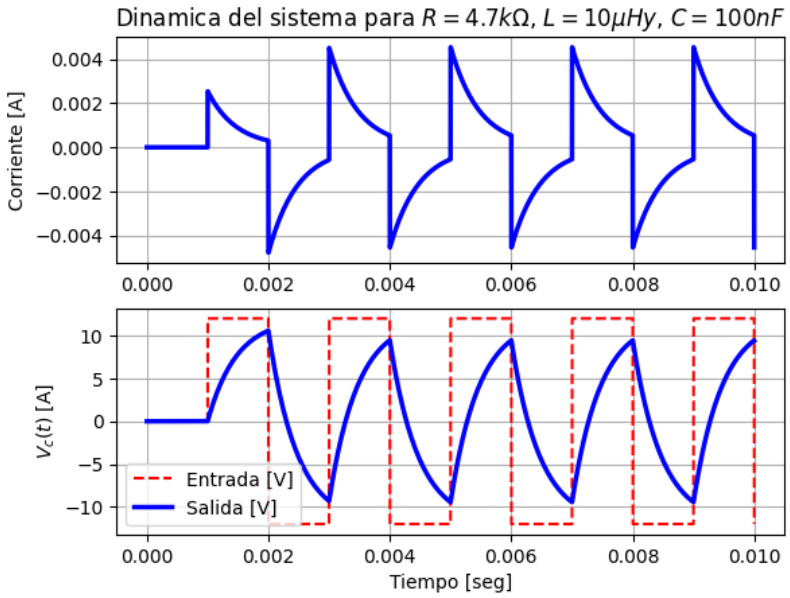

Figura 5: Resultados de la primera simulación.

Se puede apreciar en la figura 5, que en el gráfico superior está la dinámica del sistema para la salida de Corriente y en la gráfica inferior se tiene la salida de tensión en el capacitor. Se está trabajando con un sistema de una única entrada y dos salidas, para poder simular una u otra situación, se modifica la matriz de salidas C. Para lograr el comportamiento descrito, se utilizó el siguiente código:

Código 3: Primera simulación de la dinámica del sistema

```
R = 4700
L = 0.00001
C = 0.0000001

A = [[-R/L, -1/L], [1/C, 0]]
B = [[1/L], [0]]
C1= [1, 0]
C2 = [0, 1]
D = 0

sys1 = signal.StateSpace(A,B,C1,D)
sys2 = signal.StateSpace(A,B,C2,D)

t = np.linspace(0, 0.01, 10000)
u = np.piecewise(t, [t<=0.001, t>0.001], [lambda t: 0*t, lambda t: 12*signal.square(2 * np.pi * 500
  ↪ * (t - 0.001))])

t1,y1,x1 = signal.lsim(sys1,u,t)
t2,y2,x2 = signal.lsim(sys2,u,t)

plt.subplot(211)
#plt.plot(t, u, 'r--')
plt.plot(t1, y1, 'b-', linewidth=2.5)
plt.grid()
plt.ylabel('Corriente [A]')
plt.title('Dinamica del sistema para $R=4.7k\Omega$, $L=10\mu Hy$, $C=100nF$')

plt.subplot(212)
plt.plot(t, u, 'r--')
plt.plot(t2, y2, 'b-', linewidth=2.5)
plt.grid()
plt.xlabel('Tiempo [seg]')
plt.ylabel('$V_c(t)$ [A]')
plt.legend(['Entrada [V]', 'Salida [V]'], loc='best')
```

Siguiendo con la consigna, luego se pide repetir el procedimiento pero con diferentes valores para los componentes pasivos que conforman el circuito:

- $R = 5.6k\Omega$
- $L = 10\mu Hy$
- $C = 100nF$

Con lo que, a priori, se predicen salidas bastantes similares a la anterior en cuestión de órdenes de magnitud, ya que, con los nuevos valores impuestos, la constante RLC del circuito casi no se ve afectada. A continuación, el resultado:

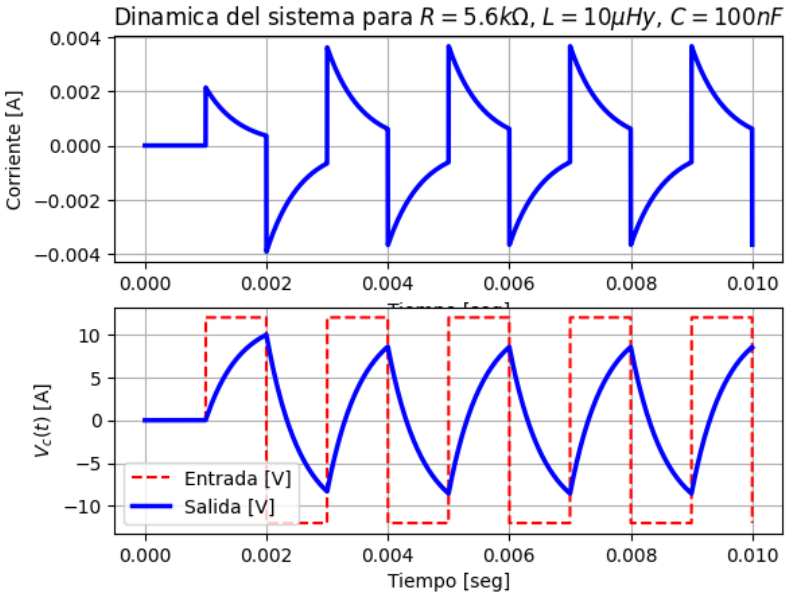

Figura 6: Resultados de la segunda simulación.

Puede verse en la figura anterior, cómo las salidas tienen una magnitud ligeramente menor con respecto a la primera simulación. Ahora, para lograr una salida bastante diferente, se modifican los valores de los componentes:

- $R = 10k\Omega$
- $L = 1\mu Hy$
- $C = 100\mu F$

Con este nuevo cambio, la constante de carga aumenta, y, la corriente tiene un mayor tiempo de establecimiento. Y, además, la carga del capacitor aumenta levemente en cada período de la señald de entrada. Aquí se nota una gran diferencia en las señales de salida. Para obtener las gráficas de las últimas simulaciones simplemente se trabajó con el mismo código adjunto anteriormente, con la salvedad de que las variables R, L y C se instanciaron con diferentes valores, siendo estos los mencionados al inicio de cada simulación.

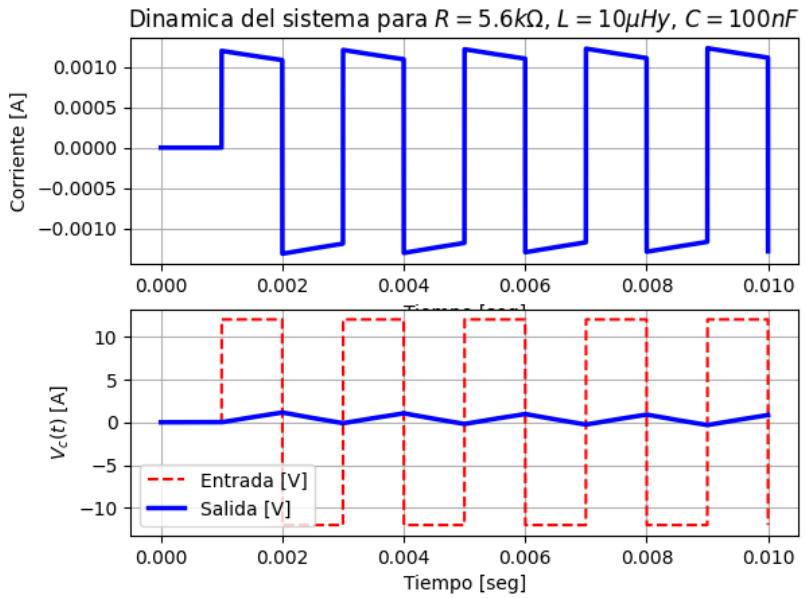

Figura 7: Resultados de la tercera simulación.

Finalmente, para poder comprobar si las simulaciones son correctas, se trabaja con la forma de las señales obtenidas. Estas son similares a las que se pueden ver en el archivo de Excel. Las señales reales, son las siguientes:

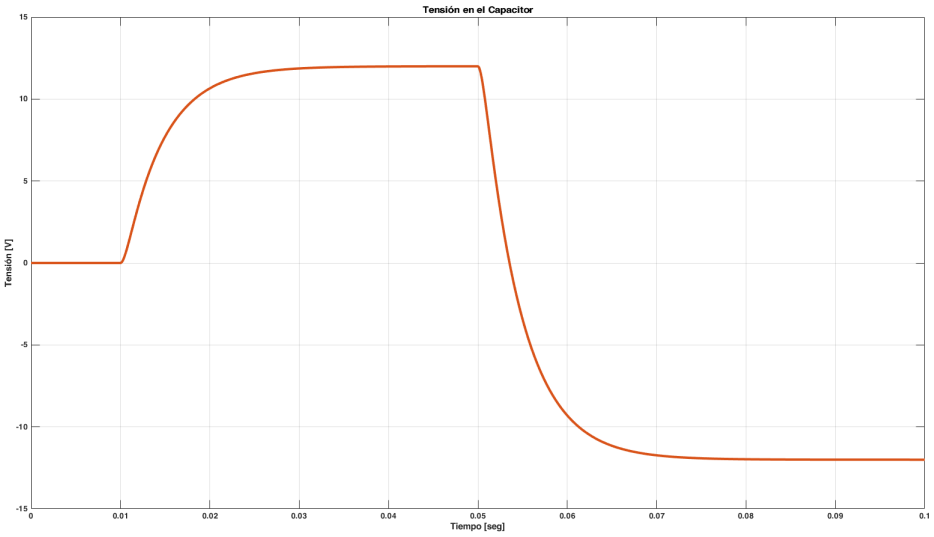

Figura 8: Tensión en el capacitor según el circuito real.

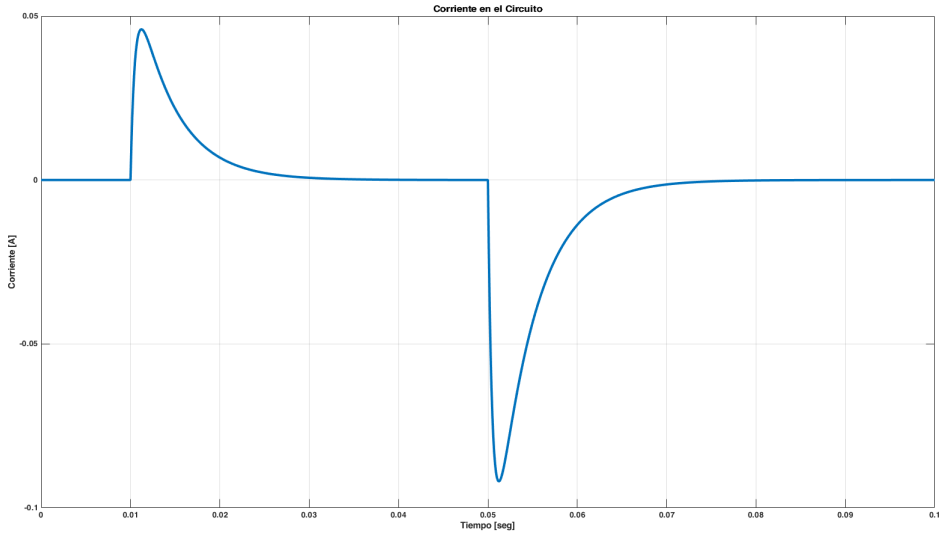

Figura 9: Corriente en el circuito según el circuito real.

2.2. Determinación de los valores de los componentes

Para el problema en cuestión, se tienen las siguientes definiciones:

- $u(t) = v_e(t)$
- $y(t) = v_c(t)$

En donde se puede expresar también que $x(t) = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$ con:

- $x_1 = i(t)$
- $x_2 = v_c(t)$

Se puede escribir entonces:

- $\dot{x} = \begin{bmatrix} -R/L & -1/L \\ 1/C & 0 \end{bmatrix} * \begin{bmatrix} i(t) \\ v_c(t) \end{bmatrix} + \begin{bmatrix} 1/L \\ 0 \end{bmatrix} * v_e(t)$
- $y = \begin{bmatrix} R \\ 0 \end{bmatrix} * \begin{bmatrix} i(t) & v_c(t) \end{bmatrix}$

Las anteriores expresiones brindan información sobre la dinámica del sistema a simular. Es por eso que, teniendo en cuenta que el vector se define como:

$$\dot{x} = \begin{bmatrix} \frac{di(t)}{dt} \\ \frac{dv_c(t)}{dt} \end{bmatrix}$$

Se pueden extraer las expresiones correspondientes a la corriente y a la tensión en el capacitor mediante una simple resolución de la suma de matrices, obteniéndose así:

- $\frac{di(t)}{dt} = \frac{v_e(t)-v_c(t)-Ri(t)}{L}$
- $\frac{dv_c(t)}{dt} = \frac{i(t)}{C}$

Se aplica la transformada de Laplace a las dos ecuacioens diferenciales anteriores, obteniéndose así:

- $sI(s) = \frac{1}{L}(V_e(s) - V_c(t) - RI(s))$
- $sV_c(s) = \frac{1}{C}I(s)$

Se puede despejar $I(s)$ en cada una de las ecuaciones, para después igualar y obtener la función de transferencia de la tensión en el capacitor. Para ello, nuevamente se utilizará Python, esta vez con el paquete simbólico.

Código 4: Despejes con el paquete simbólico de Python.

```
s,I,R,L,C,Ve,Vc = sym.symbols('s I R L C Ve Vc')
eq1 = sym.Eq(s*I,(1/L)*(Ve-Vc-R*I))
eq2 = sym.Eq(s*Vc,(1/C)*I)
I1 = sym.solve(eq1, I)
I2 = sym.solve(eq2, I)

print('Despeje de la corriente en la primera ecuación',I1)
print('Despeje de la corriente en la segunda ecuación',I2)
```

De donde se obtienen los siguientes despejes:

- $\frac{-V_c+V_e}{Ls+R}$
- CV_cs

Luego, igualando los resultados obtenidos, simplificando la función de transferencia obtenida, se logra:

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

Código 5: Despeje de la función de transferencia.

```
eq3 = sym.Eq((-Vc + Ve)/(L*s + R),C*Vc*s)
Vcap = sym.solve(eq3,Vc)
print(sym.simplify((Ve/(C*L*s**2 + C*R*s + 1))/Ve))
```

Se puede apreciar que la función de transferencia del sistema es de segundo órden, además, por la forma de la respuesta al impulso de la misma, es posible inferir que tiene dos polos reales y distintos. Para reconocer esta función de transferencia, se utiliza el método de Chen. Para ello se plantea un sistema de tres ecuaciones, pero además se deben fijar tres puntos en los que se analizará la respuesta del sistema ante una entrada del tipo escalón de tensión. El método nombrado propone dos formas diferentes de funciones de transferencia para dos tipos distintos de forma de la respuesta ante una entrada escalón unitario. En este caso, se propone la siguiente función de transferencia:

$$G(s) = \frac{K(T_3 s+1)}{(T_1 s+1)(T_2 s+1)}$$

En primera instancia, se muestran las diferentes gráficas analizadas, obtenidas del archivo CurvasMedidasRLC.xls.

Con ello, se define inicialmente un intervalo de tiempo t_1 que será usado como referencia para obtener los valores de la valuación de la función respuesta del sistema en puntos equidistantes, requisito del método de Chen. Para ello, se debe tener en cuenta la respuesta del sistema despreciando el retardo de la tensión de entrada. El intervalor t_1 es de $0.01[seg]$. Se obtienen así 3 puntos:

- $y(t_1) = y(0.02) = 10.652017[V]$
- $y(2t_1) = y(0.03) = 11.8663285[V]$
- $y(3t_1) = y(0.04) = 11.9867446[V]$

Con estos datos, y el de la ganancia estática, $K = 1$, se obtienen los valores de:

- $k_1 = \frac{y(t_1)}{K} - 1$
- $k_2 = \frac{y(2t_1)}{K} - 1$
- $k_3 = \frac{y(3t_1)}{K} - 1$

Se procede a aplicar la resolución por el método de Chen:

Código 6: Resolción por método de Chen.

```
yt_1 = 10.652017
y2t_1 = 11.8663285
y3t_1 = 11.9867446
K = 12

k_1 = (yt_1/K) - 1
k_2 = (y2t_1/K) - 1
k_3 = (y3t_1/K) - 1

series = 'k_1', 'k_2', 'k_3'
data = k_1, k_2, k_3
tableData = pd.Series(data, series)
print(tableData)

b=4*k_1**3*k_3-3*k_1**2*k_2**2-4*k_2**3+k_3**2+6*k_1*k_2*k_3;
alpha_1 = (k_1*k_2+k_3-np.sqrt(b))/(2*(k_1**2 +k_2))
alpha_2 = (k_1*k_2+k_3+np.sqrt(b))/(2*(k_1**2 +k_2))
beta = (2*k_1**3 + 3*k_1*k_2 + k_3 - np.sqrt(b))/(np.sqrt(b))
constantSeries = 'alpha_1', 'alpha_2', 'beta', 'b'
constantData = alpha_1, alpha_2, beta, b
constants = pd.Series(constantData, constantSeries)
print(constants)
t_1 = 0.01
T_1 = -t_1/np.log(alpha_1)
T_2 = -t_1/np.log(alpha_2)
```

```
T_3 = beta*(T_1-T_2)+T_1
seriesT = 'T_1', 'T_2', 'T_3'
dataT = T_1, T_2, T_3
timeConstants = pd.Series(dataT, seriesT)
print(timeConstants)
```

La resolución consiste en obtener las constantes siguientes:

- $b = 4k_1^3k_3 - 3k_1^2k_2^2 - 4k_2^3 + k_3^2 + 6k_1k_2k_3$
- $a_1 = \frac{k_1k_2+k_3-\sqrt{b}}{2(k_1^2+k_2)}$
- $a_2 = \frac{k_1k_2+k_3+\sqrt{b}}{2(k_1^2+k_2)}$
- $\beta = \frac{2k_1^3+3k_1k_2+k_3-\sqrt{b}}{\sqrt{b}}$

Las constantes de tiempo del sistema son:

- $\hat{T_1} = -\frac{t_1}{ln(\alpha_1)}$
- $\hat{T_2} = -\frac{t_1}{ln(\alpha_2)}$
- $\hat{T_3} = \beta(\hat{T_1} - \hat{T_2}) + \hat{T_1}$

Ahora, dado que la función de transferencia del circuito analizado no posee ceros, se desprecia el que aparece por el cálculo de la función de transferencia mediante el método de Chen. Con eso en mente, y desarrollando la expresión, se obtiene lo siguiente:

$$G(s) = \frac{1}{0.00000306s^2+0.005037s+1}$$

Y se procede a realizar la simulación del sistema, obteniéndose así la salida de tensión en el capacitor para la señal de entrada dada como una señal cuadrada alternada.

Código 7: Simulación del sistema obtenido.

```
t = np.linspace(0, 0.1, 1000)
u = np.piecewise(t, [(t<=0.01), (t>0.01) & (t<=0.05), (t>0.05)], [lambda t: 0*t, lambda t: 0*t+12,
  ↪ lambda t: 0*t-12])
G = signal.TransferFunction([1],[0.00000306, 0.005037, 1])
t1,y1,x1 = signal.lsim(G,u,t)

plt.plot(t, u, 'r--')
plt.plot(t1, y1, 'b-', linewidth=2.5)
plt.grid()
plt.xlabel('Tiempo [seg]')
plt.ylabel('Tensión [V]')
plt.title('Respuesta ante una entrada escalón de $12[V]$ alternada')
plt.legend(['$V_e(t)$', '$V_c(t)$'], loc='best')
```

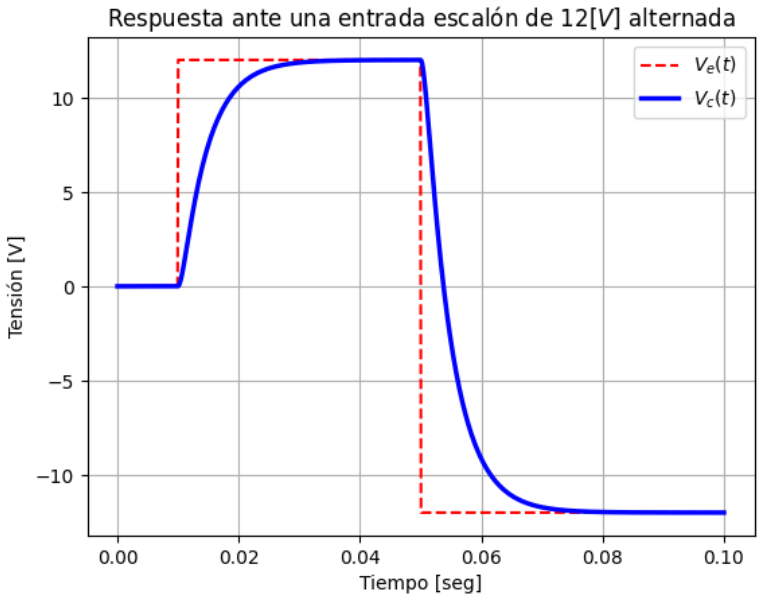

Figura 10: Respuesta temporal del sistema obtenido.

Se puede apreciar a simple vista una similitud con el gráfico que presenta la consigna de la actividad práctica. Pero para corroborar si realmente la respuesta para la tensión a bordes del capacitor, se creará un DataFrame que almacene los valores de la respuesta, con esta herramienta, es posible comparar los datos de la aproximación realizada mediante el método de Chen y los valores reales de la respuesta del circuito.

Código 8: Creación del DataFrame.

```
    df = pd.DataFrame(list(zip(t,y1)),columns=['Tiempo[seg]', 'Tensión en el capacitor [V]'])
    df.to_csv('tensionCapacitor.csv',index=False, sep=';', encoding='utf-8')
```

Ahora, se deben deducir los valores de R, L y C para el circuito en cuestión. Para ello, teniendo en cuenta que se cuenta con el dato de la ecuación característica de la función de transferencia de la tensión en el capacitor, se tienen 3 incógnitas y 2 ecuaciones. Para ello, se supone un valor de la resistencia en el circuito. En este caso, se supondrá $R = 220\Omega$, ya que es un valor comercial.

Código 9: Obtención de los valores de R, L y C.

```
R = 220
LC = 0.00000306
RC = 0.005037

C = RC/R
L = LC/C

componentSeries = 'R[ohm]', 'L[Hy]', 'C[F]'
componentValues = R, L, C
components = pd.Series(componentValues, componentSeries)
print(components)
```

Se obtiene la siguiente tabla:

Tabla 1: Componentes electrónicos del circuito.

Componente	Valor
R	220Ω
L	$0.133651Hy$
C	$0.000023F$

En el siguiente inciso de la consigna se pide emplear la serie de corriente desde $0.05seg$ en adelante para validar el resultado. Para ello, partiendo del modelo en espacio de estados que es dato del ejercicio, se lo usa para obtener la función de transferencia de la corriente:

Código 10: Comprobación de los valores obtenidos y simulación.

```
R = 220
L = 0.133651
C = 0.000023

A = [[-R/L, -1/L], [1/C, 0]]
B = [[1/L], [0]]
C= [1, 0]
D = 0

sys = signal.StateSpace(A,B,C,D)
print('Función de transferencia de la corriente: ', signal.ss2tf(A,B,C,D))

t = np.linspace(0, 0.1, 1000)
u = np.piecewise(t, [(t<=0.01), (t>0.01) & (t<=0.05), (t>0.05)], [lambda t: 0*t, lambda t: 0*t+12,
  ↪ lambda t: 0*t-12])

t1,y1,x1 = signal.lsim(sys,u,t)

print('Pico superior de corriente: ', max(y1), '[A], pico inferior de corriente: ', min(y1), ' [A]')
print('El pico superior se da en t=', t[np.where(y1 == max(y1))], ' y el pico inferior en ', t[np.
  ↪ where(y1 == min(y1))])

plt.plot(t1, y1, 'b-', linewidth=2.5)
plt.grid()
plt.xlabel('Tiempo [seg]')
plt.ylabel('Corriente [A]')
plt.title('Dinamica del sistema para $R=220\Omega$, $L=133.65 mHy$, $C=23\mu F$')
```

El código anterior lanza la siguiente gráfica, que permite corroborar la veracidad de los valores de los componentes calculados. Comparandola con la figura 9, se puede ver que son similares en magnitud y en respuesta temporal. Por lo que se asume que los resultados obtenidos son correctos.

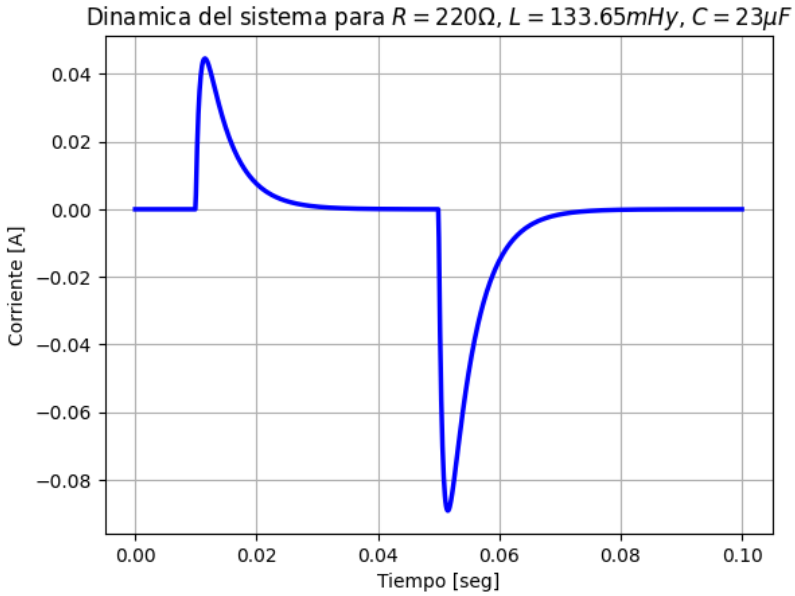

Figura 11: Comprobación de los valores mediante respuesta temporal.

Los **prints** utilizados sirvieron a modo de comprobación para comparar con los valores del Excel. Se imprime lo siguiente:

Código 11: Salida por consola de la función de transferencia de la corriente.

```
1  Función de transferencia de la corriente: (array([[0.        , 7.48217372, 0.        ]]), array
   ([1.00000000e+00, 1.64607822e+03, 3.25311901e+05]))
2  Pico superior de corriente: 0.044569008711275206 [A], pico inferior de corriente:
   -0.08913264394398462 [A]
3  El pico superior se da en t= [0.01151151] y el pico inferior en [0.05155155]
```

3. Resolución del Caso 2

La resolución de este caso está hecha enteramente en Matlab. Se irán adjuntando los códigos utilizados a medida que sea necesario.

3.1. Simulaciones

En primera instancia se integran las dos primeras ecuaciones diferenciales haciendo uso del método de Integración de Euler, con el paso fijado en la consigna del ejercicio.

Para lo anterior, primero se pasiva la entrada que corresponde al torque del motor. Se está aplicando entonces el teorema de superposición.

El código de Matlab para esta parte es el siguiente:

Código 12: Integración por Método de Euler.

```
T = 0.6;
deltaT = 1e-7;
Kmax = T/deltaT;
t = linspace(0,T,Kmax);

Laa = 366e-6;
J = 5e-9;
Ra = 55.6;
Bm = 0;
Ki = 6.49e-3;
Km = 6.53e-3;

Vin = 12;
iaP = 0;
wrP = 0;
Ia = zeros(1, Kmax);
Wr = zeros(1, Kmax);
u = linspace(0, 0, Kmax);
Ia(1) = 0;
Wr(1) = 0;
u(1) = Vin;

A = [-Ra/Laa -Km/Laa; Ki/J -Bm/J];
B = [1/Laa; 0];
C1 = [1 0];
C2 = [0 1];

Ial(1) = 0;
Wrl(1) = 0;
x = [Ia(1) Wr(1)]';
Xop = [0 0]';

ii = 0;
for i = 1:(Kmax-1)
  ii = ii+deltaT;
  if(ii>=0 && ii<0.025)
```

```
      Vin = 0;
    end
    if(ii>=0.025 && ii<0.15)
      Vin = 12;
    end
    if(ii>=0.15)
      Vin = -12;
    end
    u(i) = Vin;
    iaP = -(Ra/Laa)*Ia(i)-(Km/Laa)*Wr(i)+(1/Laa)*u(i);
    wrP = (Ki/J)*Ia(i)-(Bm/J)*Wr(i);
    xP = A*(x-Xop)+B*u(i);
    x = x+xP*deltaT;
    Y1 = C1*x;
    Y2 = C2*x;
    Ial(i+1) = x(1);
    Wrl(i+1) = x(2);
  end

```

Se puede ver cómo en las primeras líneas del código se definen las constantes globales para la resolución:

- Tiempo de simulación
- Paso de integración
- Límites de simulación
- Vector de Tiempo
- Constantes del motor definidas en la consigna
- Vectores inicializados en ceros para luego modificar
- Matrices del modelado en espacio de estados

Luego, se grafica la evolución de ambas salidas del sistema planteado para una señal de tensión que al cabo de $0.025seg$ comienza a valer $12V$ habiendo iniciado en $0V$. Más tarde, a los $0.125seg$ esta señal de entrada cambia de signo.

Código 13: Gráfica de la integración con Euler.

```
subplot(3,1,1);
plot(t,u,'LineWidth',1.5);
title('Tension de entrada v_a(t)');
xlabel('Tiempo [s]');
ylabel('Tension [V]');
grid;
subplot(3,1,2);
plot(t,Ial,'LineWidth',1.5);
title('Corriente de armadura i_a(t)');
xlabel('Tiempo [s]');
```

```
ylabel('Corriente [A]')
grid;
subplot(3,1,3);
plot(t,Wrl,'LineWidth',1.5);
title('Velocidad angular \omega_r(t)');
xlabel('Tiempo [s]');
ylabel('\omega_r [rad s^{-1}]')
grid;

```

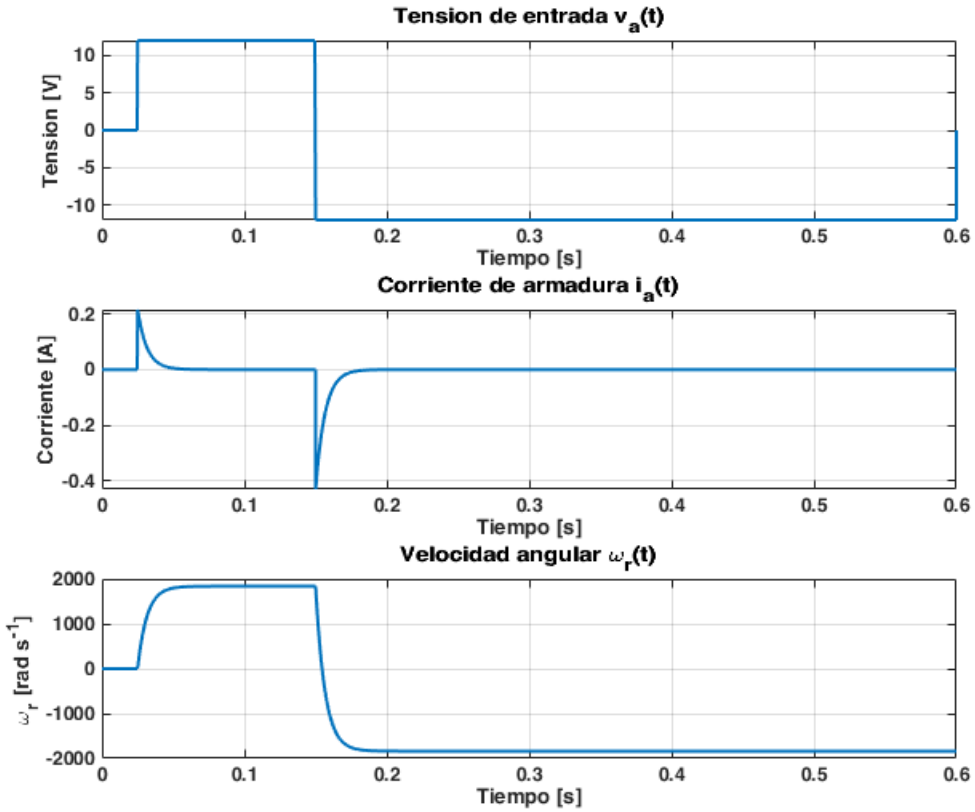

Figura 12: Resultados de la integración con Euler inicial.

Las gráficas anteriores eran simplemente como para comparar las formas de las señales de salida con las mostradas en la consigna. Ahora, se grafica según se pide en la consigna, se ingresa una entrada escalón de $12V$ y se simula el sistema durante $5seg$.

Código 14: Simulación de Euler por 5 segundos.

```
T = 5;
deltaT = 1e-7;
Kmax = 10000000;
t = linspace(0,T,Kmax);
```

```
5
6   Laa = 366e-6;
7   J = 5e-9;
8   Ra = 55.6;
9   Bm = 0;
10  Ki = 6.49e-3;
11  Km = 6.53e-3;
12
13  Vin = 12;
14  iaP = 0;
15  wrP = 0;
16  Ia = zeros(1, Kmax);
17  Wr = zeros(1, Kmax);
18  u = linspace(0, 0, Kmax);
19  Ia(1) = 0;
20  Wr(1) = 0;
21  u(1) = Vin;
22
23  A = [-Ra/Laa -Km/Laa; Ki/J -Bm/J];
24  B = [1/Laa; 0];
25  C1 = [1 0];
26  C2 = [0 1];
27
28  Ial(1) = 0;
29  Wrl(1) = 0;
30  x = [Ia(1) Wr(1)]';
31  Xop = [0 0]';
32
33  ii = 0;
34  for i = 1:(Kmax-1)
35    ii = ii+deltaT;
36    u(i) = Vin;
37    iaP = -(Ra/Laa)*Ia(i)-(Km/Laa)*Wr(i)+(1/Laa)*u(i);
38    wrP = (Ki/J)*Ia(i)-(Bm/J)*Wr(i);
39    xP = A*(x-Xop)+B*u(i);
40    x = x+xP*deltaT;
41    Y1 = C1*x;
42    Y2 = C2*x;
43    Ial(i+1) = x(1);
44    Wrl(i+1) = x(2);
45  end
46
47  subplot(3,1,1);
48  plot(t,u,'LineWidth',1.5);
49  title('Tension de entrada v_a(t)');
50  xlabel('Tiempo [s]');
51  ylabel('Tension [V]');
52  grid;
53  figure(1)
54  subplot(3,1,2);
55  plot(t,Ial,'LineWidth',1.5);
```

```
56  title('Corriente de armadura i_a(t)');
57  xlabel('Tiempo [s]');
58  ylabel('Corriente [A]')
59  grid;
60  subplot(3,1,3);
61  plot(t,Wrl,'LineWidth',1.5);
62  title('Velocidad angular \omega_r(t)');
63  xlabel('Tiempo [s]');
64  ylabel('\omega_r [rad s^{-1}]')
65  grid;
66
```

Obteniéndose la siguiente gráfica.

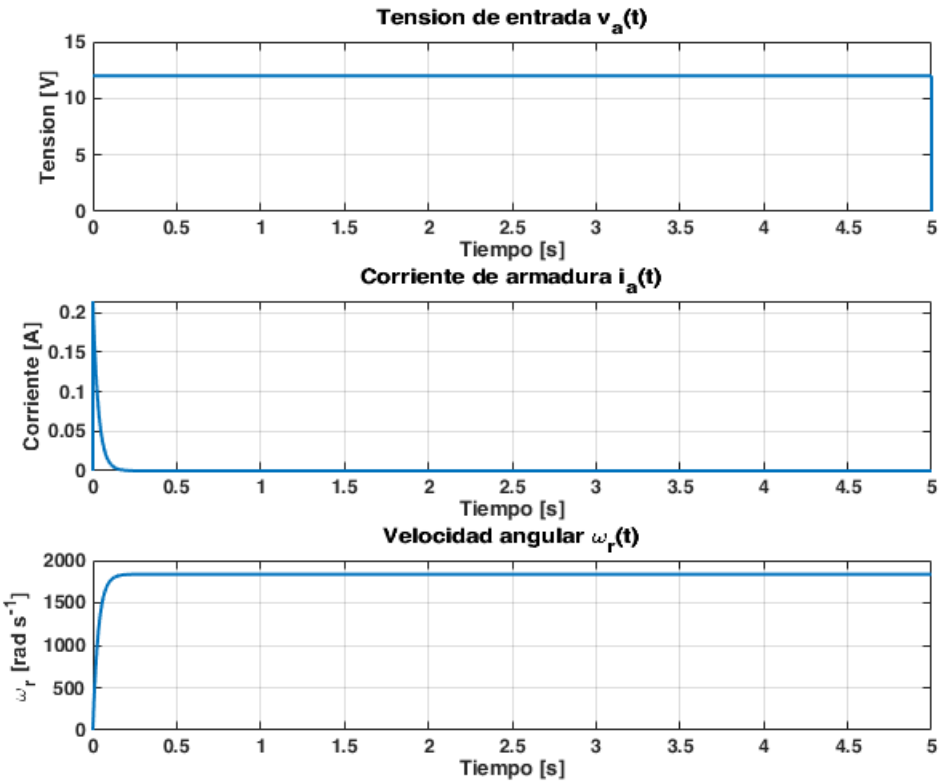

Figura 13: Simulación con Euler según consigna.

Ahora, la consigna pide obtener el máximo valor del torque que puede aceptar el sistema para que haga nula la velocidad angular. Esto puede obtenerse de dos maneras:

- Simulando mediante una entrada rampa de torque hasta encontrar el cruce por cero de la respuesta de la velocidad angular.
- Utilizar las ecuaciones diferenciales que modelan el motor de corriente continua.

Aquí, se calcula según la segunda forma propuesta. Para ello, se obtiene el pico de corriente, que en este caso es de $i_{a(max)} = 0.2146A$. Ahora, se reemplaza ese valor en la ecuación del torque del motor:

$$T = K_m i_a$$

Se obtiene entonces $T_{L(max)} = 0.0014Nm$.

Ahora, teniendo en cuenta que el sistema a trabajar es del tipo MIMO, es decir, múltiples entradas y múltiples salidas, para obtener las funciones de transferencia, se opta por trabajar de forma simbólica, es decir, haciendo uso de las expresiones no numéricas de las ecuaciones diferenciales que modelan el motor. A estas ecuaciones se las lleva al dominio de Laplace, y posteriormente, mediante ciertos despejes e igualaciones se obtienen las funciones de transferencia. Estas ecuaciones transformadas son las siguientes:

$$\begin{gathered} sI_a(s) = -\frac{R_A}{L_{AA}} I_a(s) - \frac{K_m}{L_{AA}} \Omega_r(s) + \frac{1}{L_{AA}} V_a(s) \\ s\Omega_r(s) = \frac{K_i}{J} I_a(s) - \frac{B_m}{J} \Omega_r(s) - \frac{1}{J} T_L(s) \\ s\theta_t(s) = \Omega_r(s) \end{gathered}$$

Este procedimiento se realiza en Matlab haciendo uso del paquete de matemática simbólica.

Código 15: Obtención de las funciones de transferencia en base a un procedimiento simbólico.

```
syms s I_a R_A L_AA K_m Omega_r V_a K_i J B_m Theta_t real;
eq1 = s*I_a == -(R_A/L_AA)*I_a-(K_m/L_AA)*Omega_r+(1/L_AA)*V_a
eq2 = s*Omega_r == (K_i/J)*I_a-(B_m/J)*Omega_r
Se obtienen las funciones de transferencia que relacionan la velocidad angular y la corriente con la
  ↪ tensión de entrada, para ello se pasiva la entrada de .
Omega_r = solve(eq2,Omega_r)
eq1 = eval(eq1)
Ia_Va = collect(simplify(solve(eq1,I_a)/V_a))
syms I_a Omega_r real;
eq1 = s*I_a == -(R_A/L_AA)*I_a-(K_m/L_AA)*Omega_r+(1/L_AA)*V_a;
eq2 = s*Omega_r == (K_i/J)*I_a-(B_m/J)*Omega_r;
I_a = solve(eq2,I_a)
eq1 = collect(eval(eq1))
Omegar_Va = collect(simplify(solve(eq1,Omega_r)/V_a))
Thetar_Va = collect(simplify(Omegar_Va/s))
syms s I_a R_A L_AA K_m Omega_r K_i J B_m T_L Theta_t real;
eq1 = s*I_a == -(R_A/L_AA)*I_a-(K_m/L_AA)*Omega_r
eq2 = s*Omega_r == (K_i/J)*I_a-(B_m/J)*Omega_r-(1/J)*T_L
Omega_r = solve(eq1,Omega_r);
eq2 = eval(eq2);
I_a = solve(eq2,I_a)
Ia_TL = collect(simplify(I_a/T_L))
syms I_a Omega_r real;
eq1 = s*I_a == -(R_A/L_AA)*I_a-(K_m/L_AA)*Omega_r;
eq2 = s*Omega_r == (K_i/J)*I_a-(B_m/J)*Omega_r-(1/J)*T_L;
I_a = solve(eq1,I_a);
eq2 = eval(eq2);
Omega_r = solve(eq2,Omega_r);
```

```
Omegar_TL = collect(simplify(Omega_r/T_L))
Thetar_TL = collect(simplify(Omegar_TL/s))
[Ia_Va Ia_TL; Omegar_Va Omegar_TL; Thetar_Va Thetar_TL]
```

Que deja como resultado las siguientes ecuaciones:

$$\begin{aligned}
\frac{I_a(s)}{V_a(s)} &= \frac{J\,s+B_m}{(J\,L_{AA})\,s^2+(B_m\,L_{AA}+J\,R_A)\,s+B_m\,R_A+K_i\,K_m} \\
\frac{I_a(s)}{T_L(s)} &= \frac{K_m}{(J\,L_{AA})\,s^2+(B_m\,L_{AA}+J\,R_A)\,s+B_m\,R_A+K_i\,K_m} \\
\frac{\Omega_r(s)}{V_a(s)} &= \frac{K_i}{(J\,L_{AA})\,s^2+(B_m\,L_{AA}+J\,R_A)\,s+B_m\,R_A+K_i\,K_m} \\
\frac{\Omega_r(s)}{T_L(s)} &= \frac{K_i}{(J\,L_{AA})\,s^2+(B_m\,L_{AA}+J\,R_A)\,s+B_m\,R_A+K_i\,K_m} \\
\frac{Theta_r(s)}{V_a(s)} &= \frac{K_i}{(J\,L_{AA})\,s^3+(B_m\,L_{AA}+J\,R_A)\,s^2+(B_m\,R_A+K_i\,K_m)\,s} \\
\frac{Theta_r(s)}{T_L(s)} &= \frac{(-L_{AA})\,s-R_A}{(J\,L_{AA})\,s^3+(B_m\,L_{AA}+J\,R_A)\,s^2+(B_m\,R_A+K_i\,K_m)\,s}
\end{aligned}$$

Con la posible organización en forma de matriz como se ve a continuación:

$$\begin{pmatrix}
\frac{J\,s+B_m}{(J\,L_{AA})\,s^2+(B_m\,L_{AA}+J\,R_A)\,s+B_m\,R_A+K_i\,K_m} & \frac{K_m}{(J\,L_{AA})\,s^2+(B_m\,L_{AA}+J\,R_A)\,s+B_m\,R_A+K_i\,K_m} \\
\frac{K_i}{(J\,L_{AA})\,s^2+(B_m\,L_{AA}+J\,R_A)\,s+B_m\,R_A+K_i\,K_m} & \frac{(-L_{AA})\,s-R_A}{(J\,L_{AA})\,s^2+(B_m\,L_{AA}+J\,R_A)\,s+B_m\,R_A+K_i\,K_m} \\
\frac{K_i}{(J\,L_{AA})\,s^3+(B_m\,L_{AA}+J\,R_A)\,s^2+(B_m\,R_A+K_i\,K_m)\,s} & \frac{(-L_{AA})\,s-R_A}{(J\,L_{AA})\,s^3+(B_m\,L_{AA}+J\,R_A)\,s^2+(B_m\,R_A+K_i\,K_m)\,s}
\end{pmatrix}$$

En donde puede apreciarse que las primeras 4 funciones, en las dos primeras filas de la matriz poseen la misma ecuación característica, esto corrobora la veracidad de las mismas. Además, la última fila, al referirse a una variable que resulta de ser simplemente la integración de otra (velocidad angular, fila 2 de la matriz), se tienen las mismas funciones de transferencia, con la diferencia de que las ecuaciones características son de un grado mayor por el tema del integrador de Laplace.

3.2. Mediciones reales

Las mediciones reales del circuito son obtenidas del archivo Excel de la consigna de la actividad. A modo de ilustración, se graficaron las curvas para luego corroborarlas cuando sea necesario. El código usado es el siguiente:

Código 16: Gráfico de las curvas del motor real.

```
filename = 'curvasMotor.xlsx';
sheet = 1;
t = xlsread(filename,sheet,'A1:A31054');
w = xlsread(filename,sheet,'B1:B31054');
i = xlsread(filename,sheet,'C1:C31054');
v = xlsread(filename,sheet,'D1:D31054');
T = xlsread(filename,sheet,'E1:E31054');

subplot(4,1,1);
plot(t,w,'LineWidth',1.5);
ylabel('\omega_r [rad s^{-1}]');
xlabel('Tiempo [s]');
%title('Comportamiento del motor real')
grid;
subplot(4,1,2);
```

```
16  plot(t,i,'LineWidth',1.5);
17  ylabel('i_a [A]');
18  xlabel('Tiempo [s]');
19  grid;
20  subplot(4,1,3);
21  plot(t,v,'LineWidth',1.5);
22  ylabel('V_a [V]');
23  xlabel('Tiempo [s]');
24  grid;
25  subplot(4,1,4);
26  plot(t,T,'LineWidth',1.5);
27  ylabel('T_L [Nm]');
28  xlabel('Tiempo [s]');
29  grid;
```

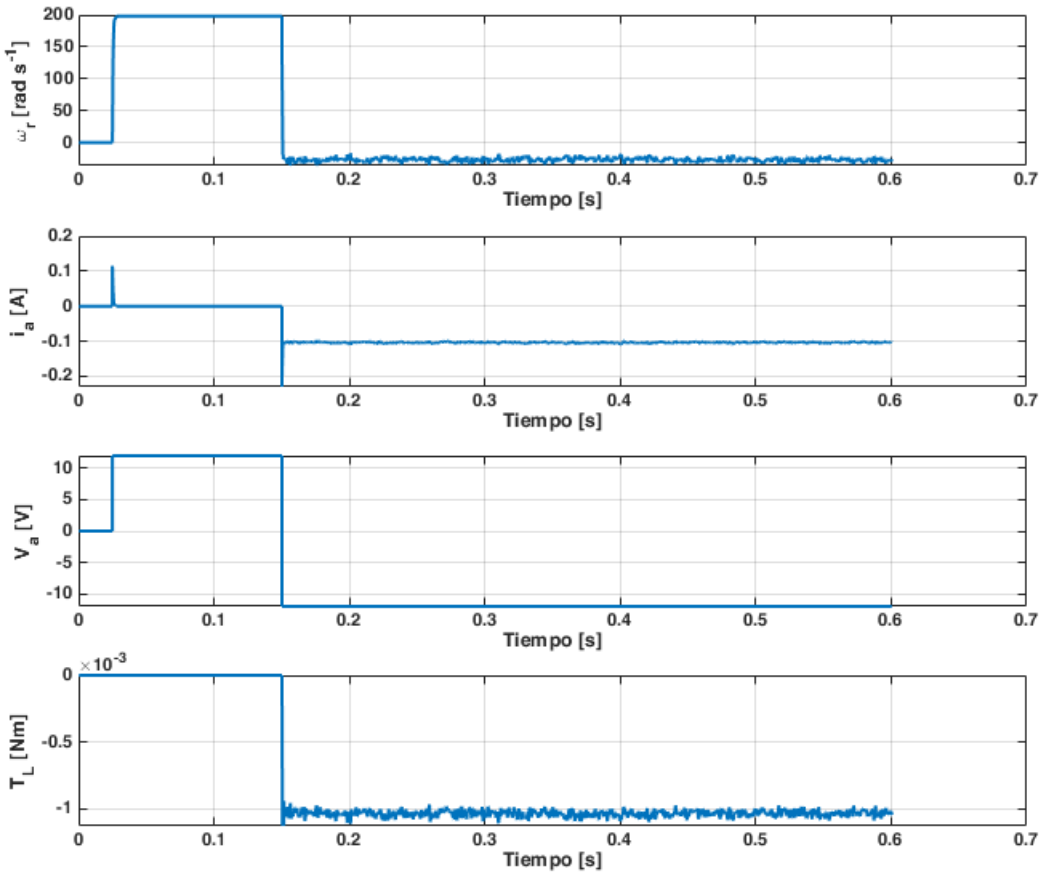

Figura 14: Curvas reales del motor.

3.3. Dinámica del sistema

El penúltimo inciso de la consigna requiere que se obtenga la dinámica del sistema para las dos entradas que tiene (tensión y torque).

Se comienza reconociendo la función de transferencia que relaciona la velocidad angular con la tensión de entrada. Para ello se aplica el método de Chen en la primer sección de la respuesta, cuando se aplica una tensión positiva de 12V y el torque es nulo. Se definen las siguientes variables a computar:

- $t_1 = 1.9\ 10^{-6}\ seg$
- $y(t_1) = 0.13231206$
- $y(2t_1) = 0.51227978$
- $y(3t_1) = 1.0610847$
- $K = 198$

Estos valores son tomados del archivo de Excel que contiene la evolución temporal del sistema. Ahora, se computa la función de transferencia:

Código 17: Método de Chen para la velocidad angular en función de tensión de armadura.

```
t1 = 1.9e-6;
y1 = 0.13231206;
y2 = 0.51227978;
y3 = 1.0610847;
K = 198;
k1 = (y1/K)-1
k2 = (y2/K)-1
k3 = (y3/K)-1
b = 4*k1^3*k3-3*k1^2*k2^2-4*k2^3+k3^2+6*k1*k2*k3
alpha1 = (k1*k2+k3-sqrt(b))/(2*(k1^2+k2))
alpha2 = (k1*k2+k3+sqrt(b))/(2*(k1^2+k2))
beta = (2*k1^3+3*k1*k2+k3-sqrt(b))/sqrt(b)
T1 = -t1/(log(alpha1))
T2 = -t1/(log(alpha2))
T3 = beta*(T1-T2)+T1
s = tf('s');
OmegaVa = minreal((K)/((T1*s+1)*(T2*s+1)))/12
```

Ahora, para reconocer la función de transferencia de que relaciona la misma salida pero con la entrada de torque, se aplica nuevamente el método de Chen, pero con una salvedad. Se puede apreciar en las curvas reales, que, al aplicar el torque en el mismo momento que se aplica la señal de tensión negativa, el sistema se establece en un valor de -28.6, lo que implica que la entrada de torque tiene una ganancia del signo opuesto a la tensión. Ahora, teniendo en cuenta que la transformada de Laplace es una transformación lineal, se puede predecir que para la entrada de tensión invertida, el sistema se debería establecer en un valor de -198.2. Con esto en mente, puede predecirse que para una entrada de torque de la magnitud del Excel, es decir, $-1.04\ 10^{-3} Nm$, se tendrá una ganancia

real de 171.4. Con estos valores se computa entonces la solución haciendo uso nuevamente de Matlab. Se usan las siguientes variables:

- $t_1 = 1\mu\ seg$
- $K = 171.4$
- $y(t_1) = -0.11090132$
- $y(2t_1) = -0.18112165$
- $y(3t_1) = -0.2504948$

Código 18: Método de Chen para la velocidad angular en función del torque de carga.

```
t1 = 1e-6;
y1 = 171.4+0.11090132;
y2 = 171.4+0.18112165;
y3 = 171.4+0.2504948;
K = (171.4/1.03e-3);
k1 = (y1/K)-1
k2 = (y2/K)-1
k3 = (y3/K)-1
b = 4*k1^3*k3-3*k1^2*k2^2-4*k2^3+k3^2+6*k1*k2*k3
alpha1 = (k1*k2+k3-sqrt(b))/(2*(k1^2+k2))
alpha2 = (k1*k2+k3+sqrt(b))/(2*(k1^2+k2))
beta = (2*k1^3+3*k1*k2+k3-sqrt(b))/(sqrt(b))
T1 = -t1/(log(alpha1))
T2 = -t1/(log(alpha2))
T3 = beta*(T1-T2)+T1
s = tf('s');
OmegaT = minreal((K)*(T3*s+1)/((T1*s+1)*(T2*s+1)*(-1)))
```

Con las funciones de transferencia obtenidas se grafica la dinámica del sistema para las entradas de torque y tensión planteadas, para ello, se hace uso de una respuesta discretizada con el uso de un bucle. Se calculan ambas respuestas para la misma salida y dos entradas con el teorema de superposición. Finalmente, estas salidas se suman. Ahora, esta suma, debería arrojar una respuesta del sistema similar a la que se puede extraer del archivo de Excel. Se adjuntan a continuación el código y la figura obtenida.

Código 19: Dinámica del sistema para la velocidad angular.

```
    h = 1e-7;
    simTime = 0.6004;
    t = 0:h:(simTime-h);
    va = zeros(1,round(simTime/h));
    tL = zeros(1,round(simTime/h));
    timeInstants = round(simTime/h)
    for i = round(0.0250001/h):1:timeInstants
      if(i <= round(0.1501/h))
        va(1,i) = 12;
```

```
      else
        va(1,i) = -12;
      end
    end
    [wv, twv] = lsim(OmegaVa,va(1,:),t);
    for i = round(0.1503883/h):1:timeInstants
      tL(1,i) = -1.04e-3;
    end
    [wt, twt] = lsim(OmegaT,tL(1,:),t);
    omega = wv+wt;
    plot(t,omega,'LineWidth',1.5)
    title('Dinámica del sistema')
    ylabel('\omega_r [rad s^{-1}]');
    xlabel('Tiempo [s]');
    xlim([0 0.6])
    grid
```

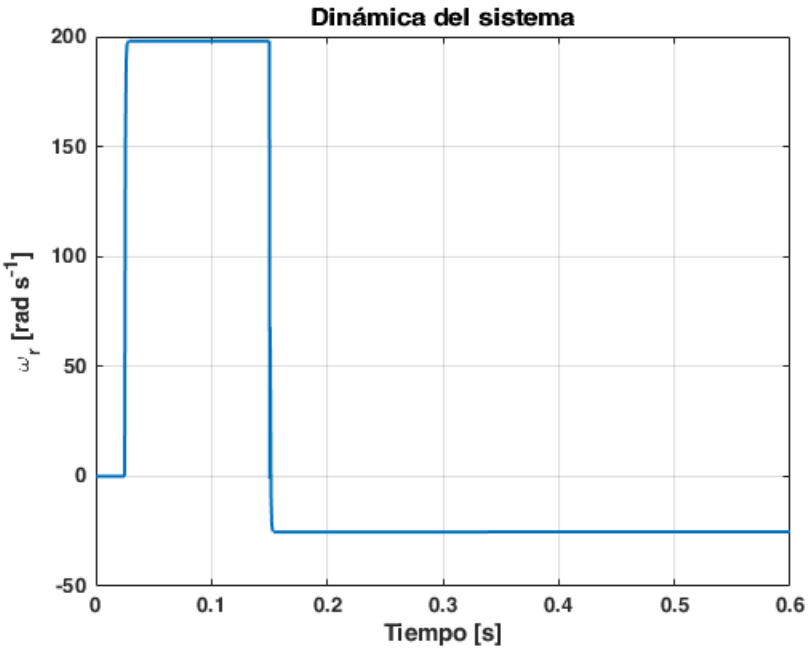

Figura 15: Respuesta temporal de la velocidad angular.

Puede verse que al curva aproximada mediante Chen es prácticamente la misma que al real, corroborándose así la exactitud del método y su aplicación en este documento.

Ahora, las funciones de transferencia obtenidas deberían tener las mismas ecuaciones características, pero, como Matlab no las imprime de esa forma, se obtuvieron de forma separada, y luego, mediante un tratamiento que se les hizo, factorizando y reescribiéndolas, se logró obtener dos funciones de transferencia con un denominador muy similar, a partir de las cuales se obtuvieron los valores de las constantes del motor de corriente continua.

Las funciones de transferencia obtenidas son:

- $\frac{\Omega_r(s)}{V_a(s)} = \frac{0.01896}{2.359\ 10^{-12}s^2+4.716\ 10^{-7}s+0.001149}$
- $\frac{\Omega_r(s)}{T_L(s)} = \frac{-4.786s-2}{2.359\ 10^{-12}s^2+2.882\ 10^{-5}s+1.202\ 10^{-5}}$

Con estas funciones de transferencia, y, suponiendo una resistencia de armadura de 2Ω, se obtuvieron los siguientes valores de las constantes:

- $R_A = 2$
- $J = 4.9285\ 10^{-13}$
- $L = 4.7857$
- $B_m = 9.8544\ 10^{-8}$
- $K_i = K_m = 0.01896$

Luego, con esos valores se construyen las funciones de transferencia que tienen como salida a la corriente de armadura, se obtuvo entonces:

- $\frac{I_a(s)}{V_a(s)} = \frac{4.584\ 10^{-9}s+1.896\ 10^{-9}}{2.359\ 10^{-12}s^2+4.716\ 10^{-7}s+0.001149}$
- $\frac{I_a(s)}{T_L(s)} = \frac{-0.01896}{2.359\ 10^{-12}s^2+4.716\ 10^{-7}s+0.001149}$

Se grafican ambas dinámicas del motor aproximado:

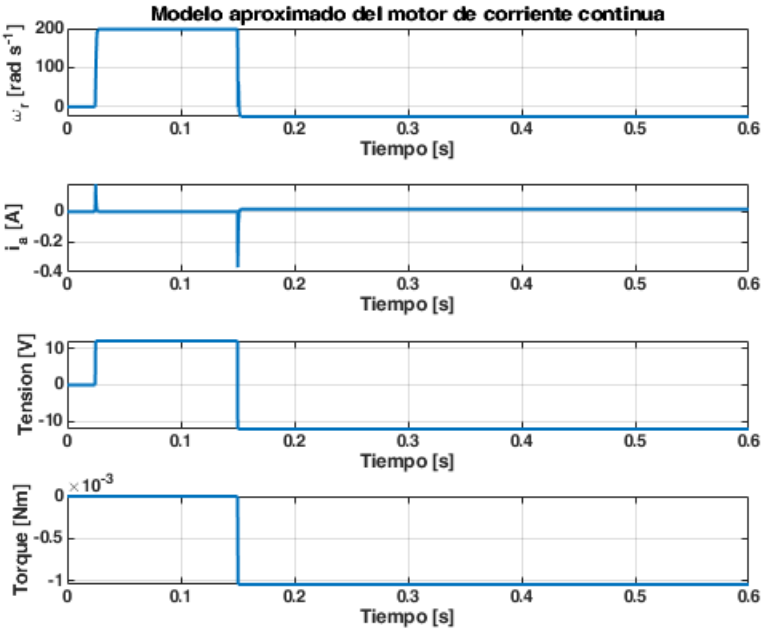

Figura 16: Dinámica del motor aproximado.

3.4. Control PID discreto

En la consigna se pide diseñar un controlador PID discreto para mantener una cierta referencia en el ángulo del eje del motor. Para ello, se planteó un PID haciendo uso de un bucle for, por el cual, a través de una función de Matlab definida como **motorModel.m** que posee las constantes del motor obtenidas, se puede simular un control.

En primera instancia, el control del motor se hizo con los valores de referencia de las constantes dadas en la consigna. A continuación el código de la función **motorModel.m**:

Código 20: Función motorModel.m

```
function [X] = motorModel(t, prevX, u)
L = 4.7857;
J = 4.9285e-13;
R = 2;
B = 9.8544e-8;
K = 0.01896;
Va = u;
h  = 1e-7;
omega = prevX(1);
wp = prevX(2);
theta = prevX(3);
for ii = 1:t/h
  wpp =(-wp*(R*J+L*B)-omega*(R*B+K*K)+Va*K)/(J*L);
  wp = wp+h*wpp;
  omega = omega + h*wp;
  thetap = omega;
  theta = theta + h*thetap;
end
X = [omega,wp,theta];
```

Esta función retorna en el arreglo X los valores de la velocidad angular, la derivada de la misma y el ángulo en radianes del eje del motor. A esa función de la llama desde el bucle for antes mencionado. La primera simulación se realiza de la siguiente forma:

Código 21: Control PID con constantes predefinidas.

```
X = -[0; 0; 0];
index = 0;
h = 1e-7;
ref = 1;
simTime = 1e-3;
Kp = 0.1;
Ki = 0.01;
Kd = 5;
samplingPeriod = h;
A = ((2*Kp*samplingPeriod)+(Ki*(samplingPeriod^2))+(2*Kd))/(2*samplingPeriod);
B = (-2*Kp*samplingPeriod+Ki*(samplingPeriod^2)-4*Kd)/(2*samplingPeriod);
C = Kd/samplingPeriod;
e = zeros(simTime/h,1);
u = 0;
for t = 0:h:simTime
```

```
    index = index+1;
    k = index+2;
    X = motorModel(h,X,u);
    e(k) = ref-X(3);
    u = u+A*e(k)+B*e(k-1)+C*e(k-2);
    theta(index) = X(3);
end
t = 0:h:simTime;
plot(t,theta,'LineWidth',1.5)
title('Salida Controlada por PID: K_p=0.1, K_i=0.01, K_D=5')
xlabel('Tiempo [s]')
ylabel('\theta [rad]')
grid
```

Esta simulación arroja la siguiente gráfica:

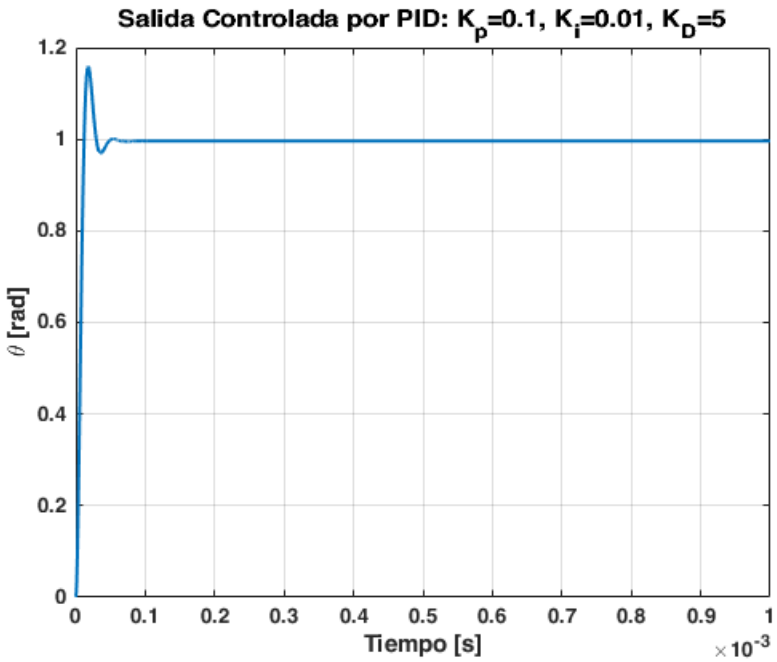

Figura 17: Salida del motor con PID predefinido.

Se puede ver en la gráfica que aparece un sobrepaso en la respuesta, para evitarla, se debería disminuir el peso de la acción derivativa del controlador PID. Para lograr eso, se disminuye ese valor. Pero, con simplemente disminuir ese valor aparece un error en estado estable. Entonces, para lograr un control que disminuye el error en estado estable y elimina el sobrepaso no deseado, se trabajó con sliders en Live Script de Matlab. Se llegó a las siguientes constantes:

- $K_P = 0.9$
- $K_i = 1.24$
- $K_D = 1.2$

- $e_{ss} = 1.5537\,\%$

Con esos valores, la respuesta obtenida es la siguiente:

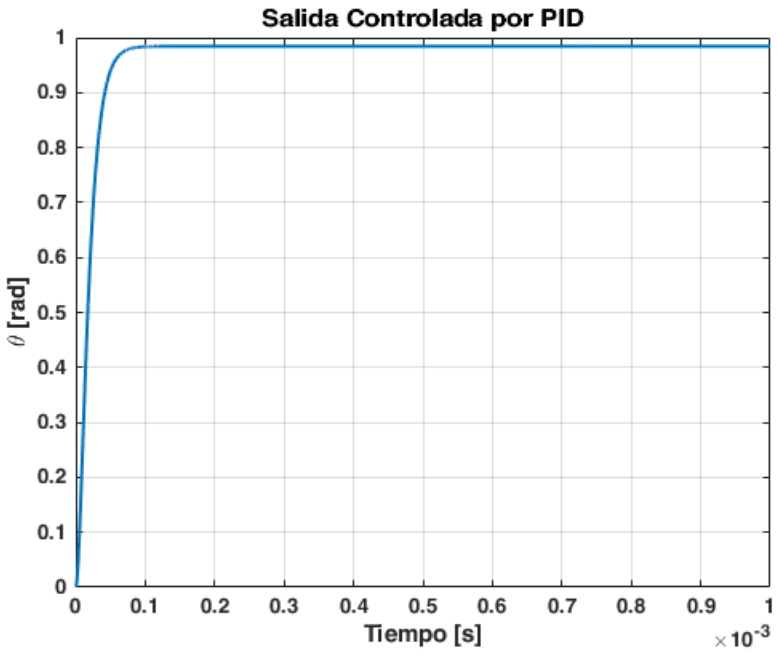

Figura 18: Salida del motor con PID sintonizado manualmente.

4. Observaciones

4.1. Generalidades

Para esta sección, simplemente se nombrarán un par de cuestiones que dificultaron en un inicio el desarrollo del trabajo práctico:

- El método de Chen funciona mejor cuando se tienen muestras t_1 más espaciadas. Si se pretende obtener a mano la función de transferencia, esto dificulta un poco la tarea. Sin embargo, si se la realiza en computadora, esto simplemente representa aumentar un paso de muestreo en una hoja de cálculo.
- Para el correcto uso del método de Chen, es necesario conocer a priori la función de transferencia a aproximar, ya que, no siempre se encuentra un cero en la función de transferencia a aproximar realmente, por lo que el cero que se calcula con el método en estos casos debe despreciarse, de modo de no obtener una respuesta simulada diferente de la esperada.
- El uso de herramientas de cálculo como Matlab o Python no asegura resultados correctos. Si bien esto es intuitivo, hay que tener en cuenta que al intentar automatizar diferentes tareas y escibir muchas líneas de código, es posible que se pasen por alto ciertos detalles o existan fórmulas mal escritas. Esto nuevamente hace que sea requisito excluyente tener una idea de lo que se pretende calcular y/o aproximar.
- El uso de Python para el cálculo de sistemas de control y simulaciones es más cómodo que trabajar con Matlab. Esto pasa sobretodo porque o se debe conseguir una licencia para Matlab, además de que Python es un lenguaje open-source que permite su ejecución en muchos entornos, por lo que en computadores con bajos recursos, o incluso online (herramientas como Google Colaboratory), facilitando así el uso para usuarios que no están muy familiarizados con algunas herramientas de programación.

Indicadores de logros

En la consigna del aula virtual se planteó la necesidad de indicar las lecciones aprendidas relacionadas con los indicadores de logro. Se enumeran a continuación los indicadores que se consideran aprendidos:

- Diseñar, proyectar y calcular sistemas de automatización y control para brindar soluciones óptimas de acuerdo a las condiciones definidas por el usuario.
- Analizar la factibilidad de controlar un proceso real conociendo su modelo.
- Calcular el modelo lineal de un proceso estable multivariable a partir de su respuesta al escalón.
- Inferir la evolución temporal de procesos reales representados en variables de estado.
- Analizar la factibilidad de controlar un proceso real conociendo su modelo.

4.2. Enlaces de utilidad

- https://github.com/all5one-sudo/TPsControl2
- https://python-control.readthedocs.io/en/0.9.3.post2/
- https://numpy.org/
- https://matplotlib.org/
- https://docs.scipy.org/doc/scipy/index.html

5. Conclusiones

Luego de la realización de este trabajo práctico, se puede decir que se terminó de comprender ciertos aspectos del modelado en el espacio de estados para procesos o sistemas reales, así como también pudo inferir el comportamiento para sistemas que cuentan con muchas variables de entrada y salida. Hasta el momento, estos sistemas eran completamente desconocidos por el alumno, ya que en la materia de Sistemas de Control I no fueron mencionados ni profundizados.

Además de lo anteriormente mencionado, gracias al material de clases de los tres profesores de la cátedra, se pudo comprender de una forma complementaria cada uno de los aspectos o variables a analizar en un sistema multivariable (MIMO).

El uso de software de simulación facilita en muchos casos los cálculos a realizar, pero no siempre son una buena opción, sobretodo cuando se trata de procedimientos en donde se requiere una gran intervención del ingeniero. Es por eso que para esta actividad se trabajó en forma de notebooks (con Jupyter Notebook y Live Script).

Finalmente, se entiende que el tema fue comprendido en su totalidad por el alumno, lográndose así el objetivo principal, que era entender de forma integral los temas de:

- Modelado en espacio de estados
- Tratamiento de sistemas de múltiples entradas y múltiples salidas
- Diseño de controladores en dominio discreto
- Familiarización con conceptos de control multivariable

Y, aplicando esos temas a un caso real, se notó que los temas son ligeramente más complejos que lo que se cree a priori.

Referencias

[1] Artículo de Chen *Paper Científico*
https://linkinghub.elsevier.com/retrieve/pii/S0895717710005613.

[2] Modelado en variables de estado de un sistema RLC *Material LIMAC*
http://www.inv.limac.efn.uncor.edu/wp-content/uploads/2014/03/2014_03_28_Material-Clases-Pucheta-SCII.pdf

[3] Modelación de un sistema monovariable de orden n *Material de clase profesor Pucheta*
https://fcefyn.aulavirtual.unc.edu.ar/pluginfile.php/308034/mod_resource/content/6/2023_03_27_Modelaci%C3%B3n%20de%20un%20sistema%20monovariable%20de%20orden%20n.pdf