Universidad Nacional de Córdoba

Facultad de Ciencias Exactas, Físicas y Naturales

Sistemas de control II

Trabajo Práctico N° 1 - Representación de sistemas y controladores

Nombre	DNI
Diaz Mateo	41.265.543

Docentes Pucheta Julian

Córdoba, República Argentina
23 de abril de 2024

Índice

Índice de figuras

1. Introduccion

En el siguiente informe se desarrolla la resolucion del primer trabajo práctico de la materia propesto por el profesor Julián Pucheta en la cátedra de Sistemas de Control II. Se detallan procedimientos y se adjuntan codigos utilizados para la resolución. Al final del informe se añaden enlaces de fuentes y bibliografia. Las herramientas utilizadas fueron:

- Visual Studio Code
- Sistema de versionado Git
- Python y librerias
- Octave
- Latex

La totalidad de este informe fue escrita usando Latex. El trabajo ademas se encuentra en un repositorio de Github personal donde se adjuntan todos los codigos en sus diversos formatos

2. Consignas

2.1. Caso de estudio 1. Sistema de dos variables de estado

Sea el sistema eléctrico de la figura, con la representaciones en variables de estado:

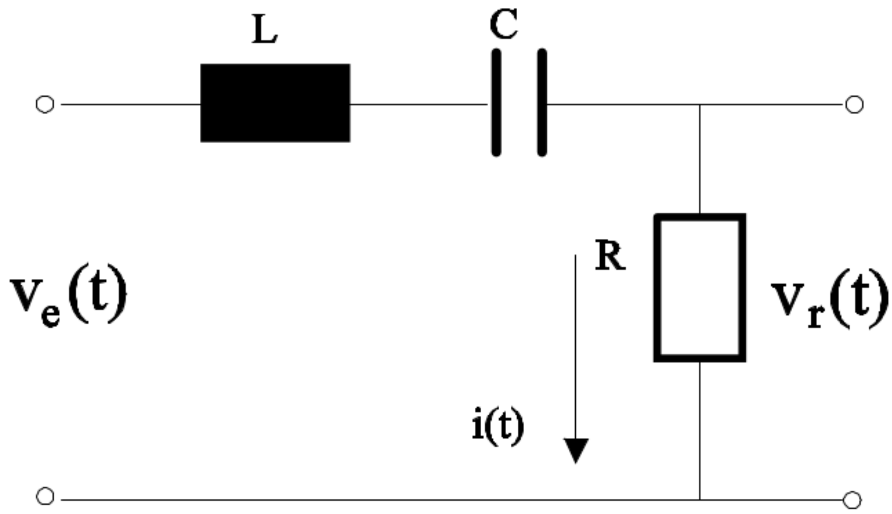

Figura 1: Circuito RLC

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

Donde las matrices que contienen a los coeficientes del circuito

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

$$B = \begin{bmatrix} -1/L \\ 0 \end{bmatrix}$$

$$c^T = \begin{bmatrix} R & 0 \end{bmatrix}$$

- Asignar valores a R=47ohm, L=1uHy, 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.
- En el archivo CurvasMedidasRLC.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.
- 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.

2.2. Caso de estudio 2. Sistema de tres variables de estado

Dada las ecuaciones del motor de corriente continua con torque de carga T_L con los parámetros : L_AA=366e10-6; J=5e10-9; R_A=55,6; B=0; K_i= 6,49e10-3 ; K_m= 6,53e10-3

$$\frac{di_a}{dt} = \frac{R_A}{L_{AA}} ia - \frac{K_m}{L_{AA}} \omega_r + \frac{1}{L_{AA}} Va$$

$$\frac{d\omega_r}{dt} = \frac{K_i}{J} ia - \frac{B_m}{J} \omega_r + \frac{1}{J} T_L$$

$$\frac{d\theta_t}{dt} = \omega_r$$

- Implementar un algoritmo de simulación para inferir el comportamiento de las variables interés mediante integración Euler con $\Delta t = 10-7$ segundos para calcular su operación con un controlador:
- Obtener el torque máximo que puede soportar el motor modelado mediante las Ecs. cuando se lo alimenta con 12V,graficando para 5 segundos de tiempo la velocidad angular y corriente ia para establecer su valor máximo como para dimensionar dispositivos electrónicos.
- A partir de las curvas de mediciones de las variables graficadas 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 TL aplicado una perturbación. 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 del modelo
- 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 (Tip: partir de KP=0,1; Ki=0,01; KD=5).
- Implementar un sistema en variables de estado que controle el ángulo del motor, paraconsignas de $\pi/2$ y $-\pi/2$ cambiando cada 2 segundos y que el TL de 1,15 10-3 aparece sólo para $\pi/2$, para $-\pi/2$ es nulo. Hallar el valor de integración Euler adecuado. El objetivo es lograr ladinámica del controlador adecuada.
- Considerando que no puede medirse la corriente y sólo pueda medirse el ángulo, por loque debe implementarse un observador. Obtener la simulación en las mismas condiciones que en el punto anterior, y superponer las gráficas para comparar.

3. Resolucion Caso 1

Para este caso de análisis utilizaremos Python . Se nos pide obtener simulaciones que estudian la dinámica del sistema. Se adjuntaran codigos y señales obtenidas.
Empezaremos planteando las librearias a utilizar:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import control as ct
from control.matlab import *
from IPython.display import Image
from scipy import signal
from math import log
```

3.1. Simulaciones

Definimos los valores de los componentes como:

- $R = 47\Omega$
- $L = 1uHy$
- $C = 100nF$

En primer lugar obtenemos la señal de 12V que cambia de signo cada 1ms. Para ello se hizo uso de la libreria *scipy* con su herramienta de señal cuadrada. Tras varios ajuste,respecto al tiempo de delay,obtenemos entonces:

```python
t_sim = 1000  # Duración de la simulación en ms
t = np.linspace(0, 0.01, t_sim)  # Arreglo de tiempo en ms
frecuencia = 500  # Frecuencia de la señal en Hz

delay = 0.001

# Generar señal cuadrada con fase ajustada para que empiece en 0 V en t=0
entrada = 12 * signal.square(2 * np.pi * frecuencia * t-np.pi, duty=0.5)

# Ajustar la señal para que esté en 0 V en t = 0 ms
entrada[0] = 0

entrada_delayed= np.where(t >= delay, entrada, 0)

# Visualizar la señal de entrada
plt.figure(figsize=(10, 4))
plt.plot(t, entrada_delayed, drawstyle='steps-pre')
plt.title('Señal de entrada escalón')
plt.xlabel('Tiempo (S)')
plt.ylabel('Voltage (V)')
plt.grid(True)
plt.show()
```

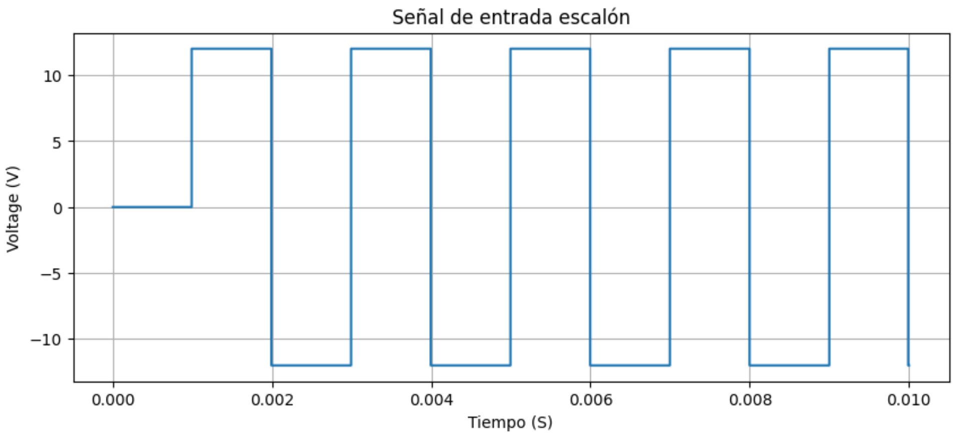

Figura 2: Dinamica Sistema R=47, L=1uHy, C=100nF

Haciendo uso nuevamente de la libreria Scipy, esta vez con el modelado en espacio de estados, planteamos las matrices y comandos correspondientes lo que nos genera el siguiente resultado

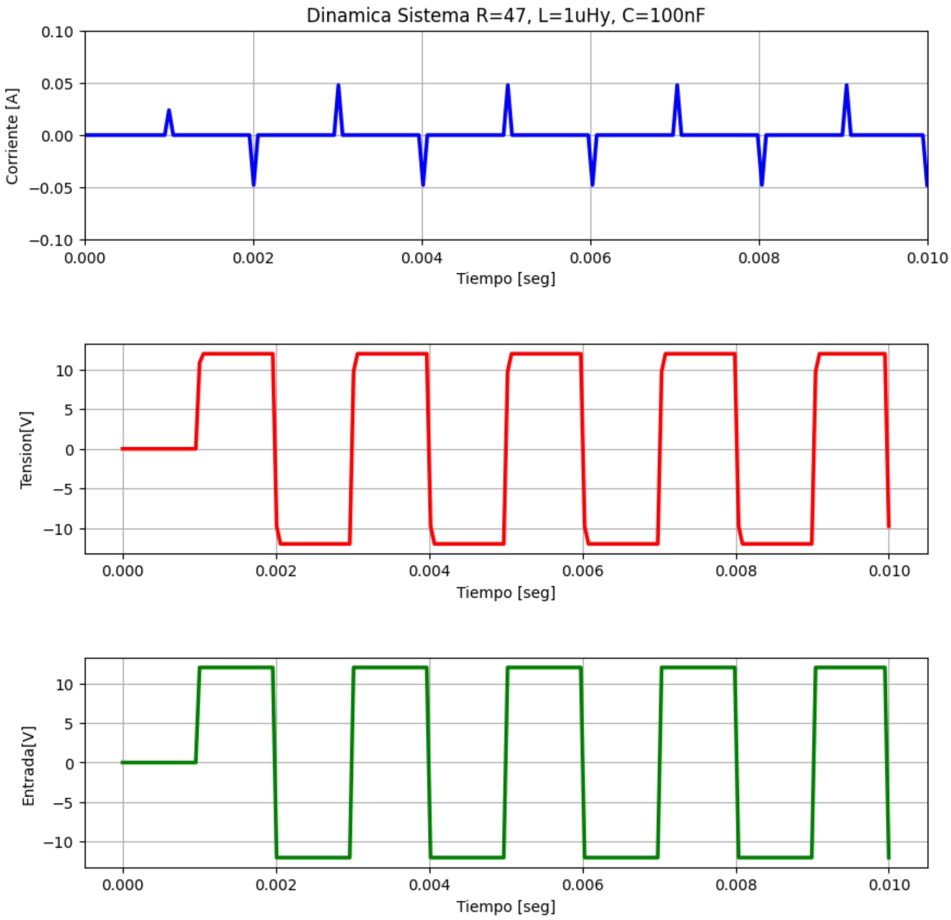

Figura 3: Señal aplicada al circuito

En este caso tenemos un sistema de una entrada y dos salidas ya que estamos interesados en medir dos variables del sistema. las dos variables de interés son la corriente y la tension en nuestro capacitor.Es por ello que planteando dos matrices de salida como C1 y C2 podemos simular el sistema. El código usado para ello fue:

```python
R=47
L=1e-6
C=100e-9

A=[[-R/L, -1/L], [1/C, 0]]
B=[[1/L], [0]]
C1=[[1, 0]] #Matriz para medir corriente
C2=[[0, 1]] #Matriz para medir voltage
D=[[0]]

sys1 = signal.StateSpace(A, B, C1, D) #voltaje capacitor
sys2 = signal.StateSpace(A, B, C2, D) #corrient

#-------------------------------------------------#
t_sim = 200 # Duración de la simulación en ms
t = np.linspace(0, 0.01, t_sim)  # Arreglo de tiempo en ms
frecuencia = 500  # Frecuencia de la señal en Hz

delay = 0.001

# Generar señal cuadrada con fase ajustada para que empiece en 0 V en t=0
entrada = 12 * signal.square(2 * np.pi * frecuencia * t-np.pi, duty=0.5)

# Ajustar la señal para que esté en 0 V en t = 0 ms
entrada[0] = 0

u= entrada_delayed= np.where(t >= delay, entrada, 0)
#-------------------------------------------------#
# Simular la respuesta del sistema
t1,y1,x1= signal.lsim(sys1,u, t) #simular sistemas LTI
t2,y2,x2= signal.lsim(sys2,u, t)

# Visualizar la salida del sistema
plt.figure(figsize=(10, 10))

plt.subplot(3, 1, 1)
plt.plot(t1, y1, 'b-', linewidth=2.5,label='Corriente')
plt.grid()
plt.title('Dinamica Sistema R=47, L=1uHy, C=100nF')
plt.xlabel('Tiempo [seg]')
plt.ylabel('Corriente [A]')
plt.ylim(-0.1, 0.1)
plt.xlim(0, 0.01)

plt.subplots_adjust(hspace = 0.5)  # Ajustar el espacio entre los subplots

plt.subplot(3, 1, 2)
plt.plot(t2, y2, 'r-', linewidth=2.5, label='Tension')
plt.xlabel('Tiempo [seg]')
plt.ylabel('Tension[V]')
plt.grid()

plt.subplot(3, 1, 3)
plt.plot(t, u, 'g-', linewidth=2.5, label='Entrada')
plt.xlabel('Tiempo [seg]')
plt.ylabel('Entrada[V]')
plt.grid()
```

Como para comparar, simularemos una respuesta al sistema cambiando los valores de los componentes. En este caso definiremos:

- $R = 4{,}7k\Omega$
- $L = 10uHy$
- $C = 100nF$

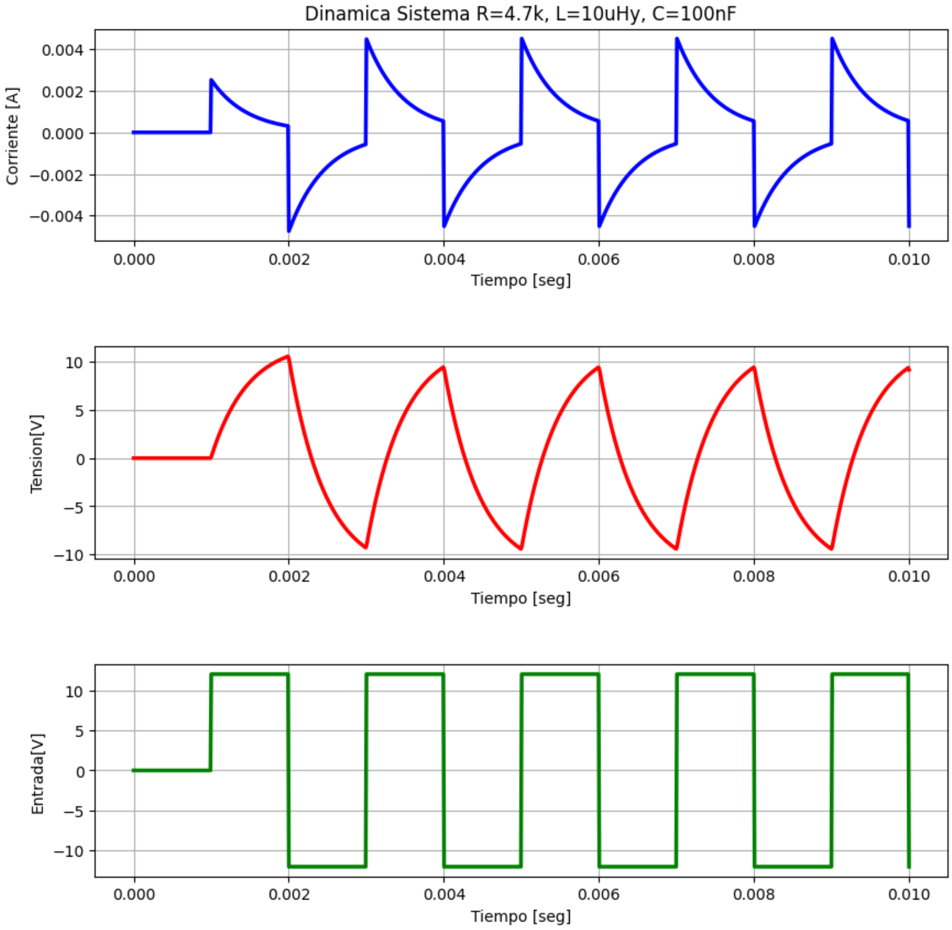

Figura 4: Dinamica Sistema R=4.7K, L=10uHy, C=100nF

Como se observar los parámetros variados fueron la resistencia y capacitor , que en conjunto podemos son la constante de carga de nuestro capacitor. Recordando las ecuaciones

$$V_c = V_0(1 - e^{\frac{-t}{RC}})$$

$$i = \frac{dq}{dt} = \frac{V_0}{R} * e^{\frac{-t}{RC}}$$

Vemos que al alterar la constante de carga, los tiempos de de establecimientos donde podemos notar la diferencia entre estos dos sismteas. El codigo que se uso fue el mismo simplemente variando los valores iniciales.

3.2. Determinacion de los valores de los componetes

Partiendo de los datos extraidos , graficaremos las curvas de tension de entrada, corriente y tension en el capacitor.Para eso, utilizando herramienta de la libreria pandas ploteamos lo siguiente

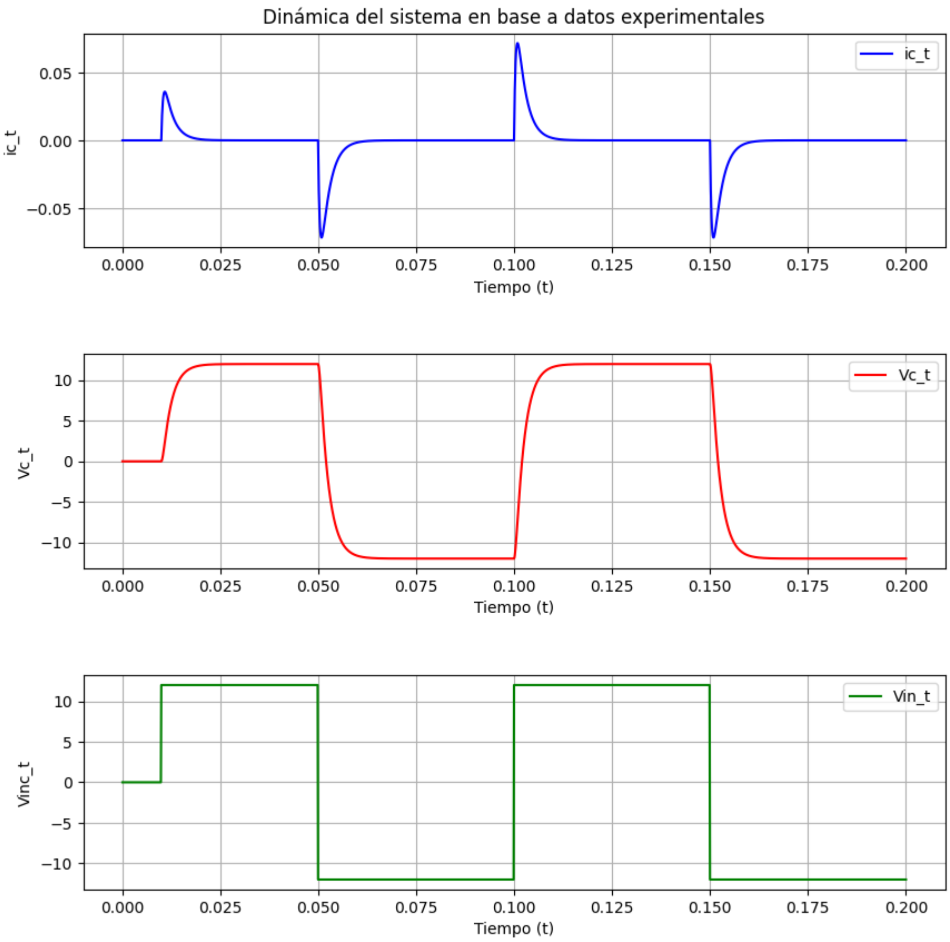

Figura 5: Dinamica sistema a estudiar

El objetivo ahora, es a partir de dichos gráficos es obtener la función de transferencia del sistema, la que después nos permite obtener los valores de R,L,C que son incógnitas del circuito

Basándonos en las variables de entrada y salida

$$\frac{di}{dt} = -\frac{R}{L}i - \frac{1}{L}vc + \frac{1}{L}ve$$

$$\frac{dv_c}{dt} = -\frac{1}{C}i$$

Podemos expresar las mismas en una ecuacion matricial-vectorial

$$\begin{bmatrix} \frac{di}{dt} \\ \frac{dv_c}{dt} \end{bmatrix} = \begin{bmatrix} -R/L & -1/L \\ 1/C & 0 \end{bmatrix} \begin{bmatrix} i \\ vc \end{bmatrix} + \begin{bmatrix} 1/L \\ 0 \end{bmatrix} [V_e]$$

Definiendo a I y Vc como variables de estado y a X como vector de estado podemos expresarlo:

$$\dot{x} = Ax(t) + b(u)$$

Transformando al dominio de laplace el conjunto de ecuaciones

$$sI(s) = \frac{1}{L}$$

$$sV_c(s) = \frac{1}{C}I(s)$$

Despejamos I de ambas ecuaciones para igualar y obtener la FDT

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

Se obvserva que es una FdT de segundo grado y tiene dos polos reales y distintos. Aplicaremos método Chen para reconocerla.

El método de **Chen** es una técnica utilizada para identificar la función de transferencia de un sistema a partir de datos experimentales. Es particularmente útil cuando se dispone de mediciones de la entrada y la salida del sistema, pero no se tiene información detallada sobre la estructura interna del sistema.Es una herramienta poderosa para la identificación de sistemas y se puede aplicar a una amplia variedad de sistemas dinámicos, desde sistemas mecánicos y eléctricos hasta sistemas biológicos y económicos. Sin embargo, es importante tener en cuenta que la precisión de los resultados obtenidos depende en gran medida de la calidad y la cantidad de los datos experimentales disponibles, así como de la elección de los parámetros.
Basandonos en los recursos de Identificacion.ipynb brindados en clase

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

Como nos indica el método debemos definir un intervalo de tiempo t_1 que será usado de referencia, que es lo que se nos pide en el método de de Chen. En primer lugar debemos detectar nuestro t_1 de referencia, para ello con ayuda de gráficas interactivas podremos expandir nuestros gráficos y obtener aproximadamente nuestro t_1 que luego integrara el algoritmo de Chen.
Con el objetivo de tomar bien la dinamica del sistema tomamos un t_1=0.01 (despreciando retardo) Basándonos en el recurso dado en clase:

$$y(t1) = y(0{,}01) = 10{,}6$$

$$y(2t1) = y(0{,}02) = 11{,}88$$

$$y(3t1) = y(0{,}03) = 11{,}99$$

Planteamos entonces el algoritmo:

```python
yt1 = 10.6
yt2 = 11.88
yt3 = 11.99
Kp= 12

yt_1=yt1/12
yt_2=yt2/12
yt_3=yt3/12
K=Kp/12

k_1 = (yt_1 / K) - 1
k_2 = (yt_2 / K) - 1
k_3 = (yt_3 / K) - 1

print('k_1: {:.2e}'.format(k_1))
print('k_2: {:.2e}'.format(k_2))
print('k_3: {:.2e}'.format(k_3))

#-----------------------------------------------#
be = 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(be)) / (2 * (k_1**2 + k_2))
alpha_2 = (k_1 * k_2 + k_3 + np.sqrt(be)) / (2 * (k_1**2 + k_2))
beta=(k_1+alpha_2)/(alpha_1-alpha_2)

print('be: {:.2e}'.format(be))
print('alpha_1: {:.2e}'.format(alpha_1))
```

```python
print('alpha_2: {:.2e}'.format(alpha_2))
print('beta: {:.2e}'.format(beta))

#-------------------------------------------------#
t_1 = 0.0054
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

print('T_1: {:.2e}'.format(T_1))
print('T_2: {:.2e}'.format(T_2))
print('T_3: {:.2e}'.format(T_3))
```

Una vez determinado T1,T2,T3 que son las constantes, puedo ya determinar la funcion de transferencia con la ayuda de las herramientas de la librería de control

```python

t_s= np.linspace(0, 0.03, 1000)

# Crear la función de transferencia utilizando control.tf()#
sys_G = K*ct.tf([0, 1],np.convolve([T_1, 1],[T_2, 1]))

# Calcular la respuesta al escalón del sistema identificado#
y_id , t_id = step(1*sys_G, t_s)

# Agregar un retraso a la respuesta al escalón
delay = 0.01  # Retraso en segundos
t_id_delayed = t_id + delay

print(sys_G)

plt.plot(t_id_delayed, y_id)

plt.xlabel('Time')
plt.ylabel('Step Response')
plt.title('Respuesta al escalon G identificada')
plt.grid(True)
plt.show()
```

Obteniendo asi la respuesta al escalon de G de nuestro sistema a identificar

$$\boxed{G(s) = \frac{1}{2{,}503 \times 10^{-6} \cdot S^2 - 3{,}32 \times 10^{-3} \cdot S + 1}}$$

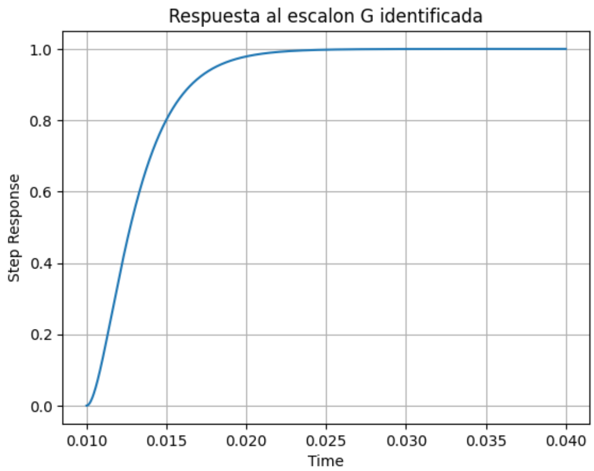

Figura 6: Respuesta a escalón sistema identificado

Superponiendo con la grafica anterior podemos ver una similitud con el sistema extraido de los datos experimentales. Para corroborar esto podemos superponer las graficas

Figura 7: Superposicion gráficos tension

Teniendo en cuenta los valores obtenidos de las graficas de datos, vemos que tenemos una tension de 12V y aproximadamente 40mA de pico de corriente, lo que nos permite estimar un valor de R . Tomaremos un valor de R=270 lo que nos permitirá calcular los valores restantes:

```
R=270
C=3.324e-3/R
L=2503e-6/C
print(L)
print(C)
```

Obteniendo asi los siguientes valores:

- $R = 270\Omega$
- $L = 203uHy$
- $C = 12{,}3uF$

Una vez determinado los parámtros R,L,C, emplearemos la serie de corriente desde 0.05s en adelante para validar el resultado:

```
# Definir los valores de R, L y C
R = 270
L = 203e-3
C = 12.3e-6
```

```

df= pd.read_excel('Curvas_Medidas_RLC_2024.xls') # extraigo datos de xls

t = df.iloc[:, 0]
Vin_t= df.iloc[:, 3]


# Interpolar los datos de entrada
Vin_t_func = interp1d(t, Vin_t, fill_value="extrapolate")

# Crear un arreglo de tiempo para la simulación
t_sim = np.linspace(0, max(t), len(t))

# Crear la señal de entrada para la simulación
u = Vin_t_func(t_sim)


A=[[-R/L, -1/L], [1/C, 0]]
B=[[1/L], [0]]
C=[[1, 0]] #Matriz para medir corriente
D=[[0]]

sys = signal.StateSpace(A, B, C, D)

# Simular la respuesta del sistema
t1,y1,x1= signal.lsim(sys,u, t) # simular sistemas (LTI)

# Visualizar la salida del sistema
plt.figure(figsize=(10, 10))

# Plot both the simulated current and the measured current on the same subplot
plt.plot(t1, y1, 'b-', linewidth=2.5, label='Corriente simulada')
plt.plot(t, ic_t, 'r-', linewidth=2.5, label='Corriente medida')

plt.grid()
plt.title('Dinamica Sistema')
plt.xlabel('Tiempo [seg]')
plt.ylabel('Corriente [A]')
plt.legend()

plt.show()
```

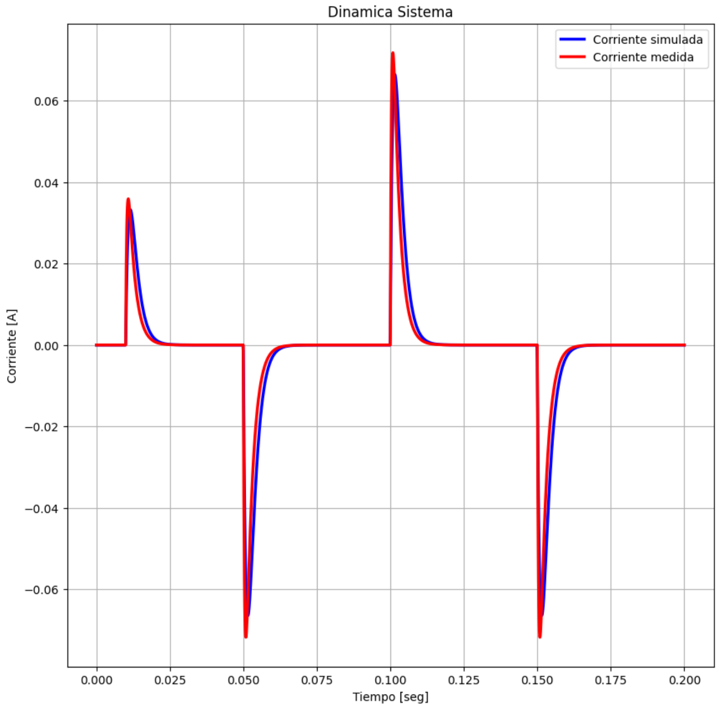

Figura 8: Superposicion gráficos corriente

Podemos hacer una comprobacion en base a codigo :

```
# Calcular el valor máximo de la corriente simulada
max_current_simulado = max(y1)

# Calcular el valor máximo de la corriente medida
max_current_medido = max(ic_t)

print("El valor máximo de la corriente simulada es: ", max_current_simulado)
print("El valor máximo de la corriente medida es: ", max_current_medido)
```

Obteniendo en consola

```
El valor máximo de la corriente simulada es:  0.06639982210909336
El valor máximo de la corriente medida es:  0.07176714792834286
```

4. Resolucion Caso 2 - Motor Corriente Continua

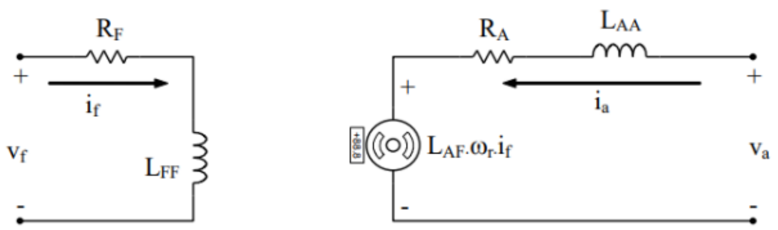

Figura 9: Simulación integración por Euler

4.1. Simulacion Euler

La resolucion de este caso esta hecha parte en python y parte en octave. Se adjuntaran los codigos correspondientes en cada caso

Empezaremos planteando las librearias a utilizar:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import control as ct
from control.matlab import *
import cmath as cm
from math import log
```

Para la simulacion utilizaremos el método de **Inegracion de Euler** con el paso fijado en la consigna.

```
# Definir los parámetros del sistema
R_A = 55.6
L_AA = 366e-6
K_m = 6.53e-3
J = 5e-9
B_m = 0
K_i = 6.49e-3

# Definir las matrices A, B, C y D del sistema de espacio de estados
A = np.array([[-R_A/L_AA, -K_m/L_AA],
              [K_i/J, -B_m/J]])
B = np.array([[1/L_AA], [0]])
C = np.array([[1, 0],
              [0, 1]])
D = np.array([[0], [0]])

# Definir la función que calcula la derivada del estado
def derivada_estado(x, u):
    return np.dot(A, x) + np.dot(B, u)

# Definir la función que calcula la salida
def salida(x, u):
    return np.dot(C, x) + np.dot(D, u)

# Definir el tiempo de simulación y el paso de tiempo
tiempo_simulacion = 1.0
delta_t = 10e-7

# Inicializar las variables del sistema
```

```python
x = np.array([[0], [0]])  # Condiciones iniciales nulas
u = np.array([[12]])      # Voltaje aplicado

# Almacenar resultados de la simulación
tiempos = [0]
corriente_armadura = [x[0, 0]]
velocidad_angular = [x[1, 0]]
voltaje = [u[0, 0]]

# Realizar la simulación mediante el método de Euler
t = 0.0
while t < tiempo_simulacion:
    # Calcular el siguiente estado utilizando el método de Euler
    x = x + derivada_estado(x, u) * delta_t

    # Calcular la salida
    y = salida(x, u)

    # Almacenar los resultados
    t += delta_t
    tiempos.append(t)
    corriente_armadura.append(y[0, 0])
    velocidad_angular.append(y[1, 0])
    voltaje.append(u[0, 0])

# Graficar las variables de interés
plt.figure(figsize=(10, 6))
plt.subplot(3, 1, 1)
plt.plot(tiempos, corriente_armadura, label='Corriente del armadura')
plt.xlabel('Tiempo (s)')
plt.ylabel('Corriente (A)')
plt.title('Corriente de armadura Ia')
plt.legend()
plt.grid()


plt.subplot(3, 1, 2)
plt.plot(tiempos, velocidad_angular, label='Velocidad angular del rotor')
plt.xlabel('Tiempo (s)')
plt.ylabel('Velocidad angular (rad/s)')
plt.title('Velocidad angular Wr')
y_min = min(velocidad_angular)
plt.ylim(y_min, 2000)
plt.legend()
plt.grid()


plt.subplot(3, 1, 3)
plt.plot(tiempos, voltaje, label='Voltaje aplicado')
plt.xlabel('Tiempo (s)')
plt.ylabel('Voltaje (V)')
plt.title('Voltaje aplicado Va')
plt.legend()
plt.grid()

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

Obtenemos entonces la siguiente gráfica:

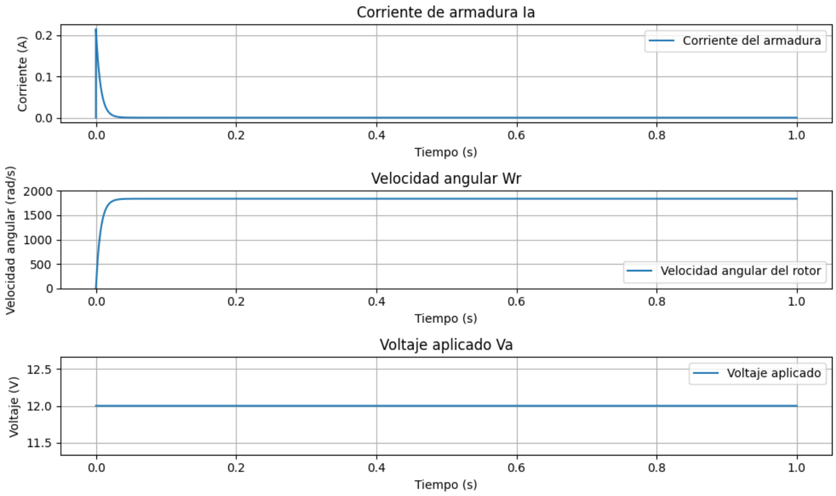

Figura 10: Simulación integración por Euler

4.2. Torque Máximo

Lo siguiente es obtener el valor máximo de torque que puede aceptar el sistema.Para ello podemos determinarlo a partir de el pico de corriente obtenido en las gráficas. Reemplazando ese valor en la ecuacion

$$T = K_m * i_a$$

Planteando en codigo:

```
max_Ia = np.max(corriente_armadura)
max_Ia = round(max_Ia, 4)  # Redondear a 4 decimales
print('El valor maximo de la corriente de armadura es de:')
print(max_Ia)

T= K_m*max_Ia
T = round(T, 4)  # Redondear a 4 decimales
print('El valor maximo del torque es de:')
print(T)
```

Obteniendo como resultado en consola:

```
El valor maximo de la corriente de armadura es de:
0.2147
El valor maximo del torque es de:
0.0014 Nm
```

4.3. Identificacion a partir de mediciones

Empezamos reconociendo la función de transferencia que relaciona la velocidad angular y al torque de carga TL. Para ello graficaremos las dinamicas obtenidas

```
import plotly.graph_objects as go


df= pd.read_excel('Curvas_Medidas_Motor_2024.xls') # extraigo datos de xls
t = df.iloc[:, 0] #selecciono primera columna y todas sus filas, guardo como variable t
W_r = df.iloc[:, 1]
```

```
i_a = df.iloc[:, 2]
v=df.iloc[:, 3]
T= df.iloc[:, 4]

# Crear una figura y un conjunto de subtramas
fig, axs = plt.subplots(4, figsize=(10, 10))

# Graficar voltaje en función del tiempo
axs[0].plot(t, v, label='Voltaje')
axs[0].set_xlabel('Tiempo (s)')
axs[0].set_ylabel('Voltaje (V)')
axs[0].legend()
axs[0].grid()

# Graficar corriente de armadura en función del tiempo
axs[1].plot(t, i_a, label='Corriente de armadura')
axs[1].set_xlabel('Tiempo (s)')
axs[1].set_ylabel('Corriente (A)')
axs[1].legend()
axs[1].grid()

# Graficar velocidad angular en función del tiempo
axs[2].plot(t, W_r, label='Velocidad angular')
axs[2].set_xlabel('Tiempo (s)')
axs[2].set_ylabel('Velocidad angular (rad/s)')
axs[2].legend()
axs[2].grid()

# Graficar torque en función del tiempo
axs[3].plot(t, T, label='Torque')
axs[3].set_xlabel('Tiempo (s)')
axs[3].set_ylabel('Torque (Nm)')
axs[3].legend()
axs[3].grid()

# Ajustar el layout
plt.tight_layout()
plt.show()

```

El codigo anterior nos arroja las siguientes graficas:

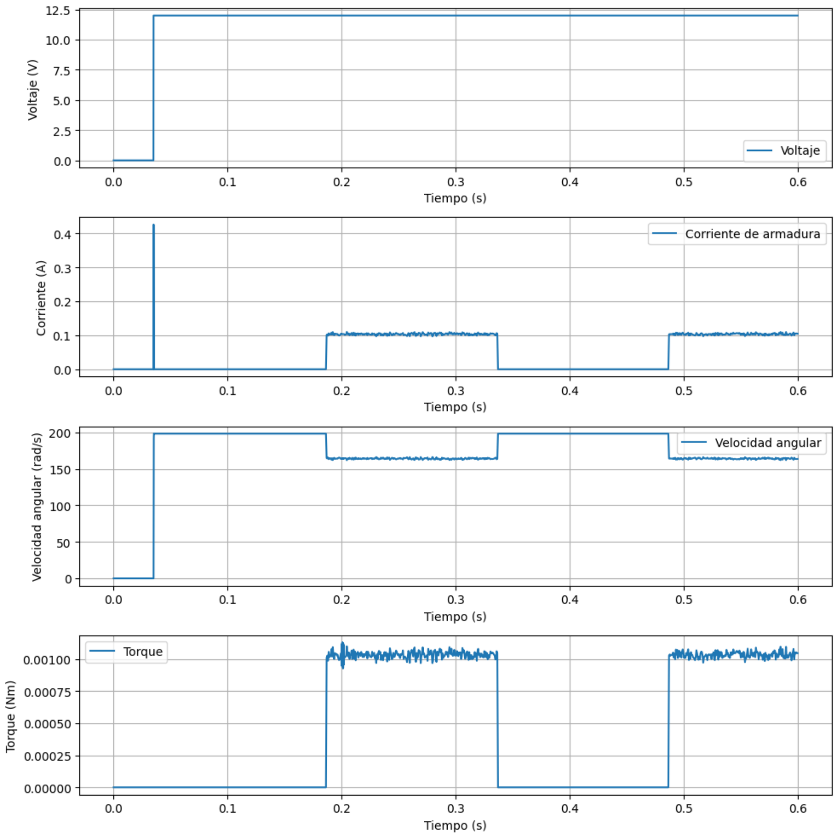

Figura 11: Simulación integración por Euler

Nuevamente para obtener una funcion de transferencia que relacione la velocidad angular como salida y la tension de Vin como la entrada junto con sus constantes escalares , utilizaremos el metodo de Chen previamente usado,considerando como entrada un escalón de 12V, como salida a la velocidad angular, y al torque de carga TL aplicado una perturbación.

Con ayuda de los valores del excel y curvas graficadas definimos las variables

```
yt1 =135.583968726574
yt2 =191.604441793493
yt3 =198.304639094699
yt_1 =yt1/12
yt_2 =yt2/12
yt_3 =yt3/12

Kp=198
K = Kp/12


k_1 = (yt_1 / K) - 1
k_2 = (yt_2 / K) - 1
k_3 = (yt_3 / K) - 1

print('k_1: {:.2e}'.format(k_1))
print('k_2: {:.2e}'.format(k_2))
print('k_3: {:.2e}'.format(k_3))

#-----------------------------------------------#

import cmath
be =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
```

```
if be > 0: #Cambiar el cálculo para usar números complejos - Extraido Collab
  alpha_1=(k_1*k_2+k_3-np.sqrt(be))/(2*(k_1**2+k_2))
  alpha_2=(k_1*k_2+k_3+np.sqrt(be))/(2*(k_1**2+k_2))
else :
  alpha_1=(k_1*k_2+k_3-cm.sqrt(be))/(2*(k_1**2+k_2))
  alpha_2=(k_1*k_2+k_3+cm.sqrt(be))/(2*(k_1**2+k_2))

#alpha_1 = (k_1 * k_2 + k_3 - np.sqrt(be)) / (2 * (k_1**2 + k_2))
#alpha_2 = (k_1 * k_2 + k_3 + np.sqrt(be)) / (2 * (k_1**2 + k_2))
#beta =(2 * k_1**3 + 3 * k_1 * k_2 + k_3 - np.sqrt(be))/(np.sqrt(be))

beta=(k_1+alpha_2)/(alpha_1-alpha_2)

print('be: {:.2e}'.format(be))
print('alpha_1: {:.2e}'.format(alpha_1))
print('alpha_2: {:.2e}'.format(alpha_2))
print('beta: {:.2e}'.format(beta))
#-------------------------------------------------#

t_1 = 100e-6
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

print('T_1: {:.2e}'.format(T_1))
print('T_2: {:.2e}'.format(T_2))
print('T_3: {:.2e}'.format(T_3))
```

Una vez determinado T1,T2,T3 que son las constantes, puedo ya determinar la funcion de transferencia con la ayuda de las herramientas de la librería de control

```
t_s= np.linspace(0, 0.005, 1000)

# Crear la función de transferencia utilizando control.tf()#
sys_G = K*ct.tf([0, 1],np.convolve([T_1, 1],[T_2, 1]))


# Calcular la respuesta al escalón del sistema identificado#
y_id , t_id = step(1*sys_G, t_s)

# Agregar un retraso a la respuesta al escalón
delay = 0.0351  # Retraso en segundos
t_id_delayed = t_id + delay

print(sys_G)
plt.plot(t_id_delayed, y_id)

plt.xlabel('Time')
plt.ylabel('Step Response')
plt.title('Respuesta al escalon G identificada')
plt.grid(True)
plt.show()
```

Obteniendo asi la respuesta al escalon de G de nuestro sistema a identificar

$$\boxed{G(s) = \frac{16{,}5}{2{,}23 \times 10^{-9} \cdot S^2 - 8{,}433 \times 10^{-5} \cdot S + 1}}$$

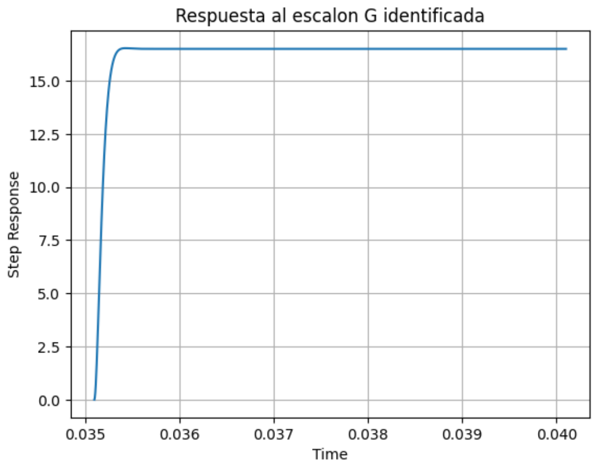

Figura 12: Respuesta a escalón sistema identificado Motor

Al comparar esta respuesta identificada con la que obtuvimos partiendo de los valores del excel. Tener en cuenta que debo multiplicar por K ya que la respuesta que obtuvimos es en base al escalon unitario.

```
# Graficar Vc_t
plt.figure(figsize=(10, 10))
plt.subplot(3, 1, 2)
plt.plot(t, W_r, 'r-', label='Wr_medido')

# Graficar la respuesta al escalón del sistema identificado en el mismo gráfico
Ksys=12

plt.plot(t_id_delayed, Ksys*y_id, label='Respuesta al escalón G identificada')

# Configurar los títulos de los ejes y la leyenda
plt.xlabel('Tiempo (t)')
plt.ylabel('Respuesta')
plt.title('Respuesta al escalon G identificada')
plt.legend()
plt.grid(True)

# Mostrar el gráfico
plt.show()
```

Obteniendo la grafica:

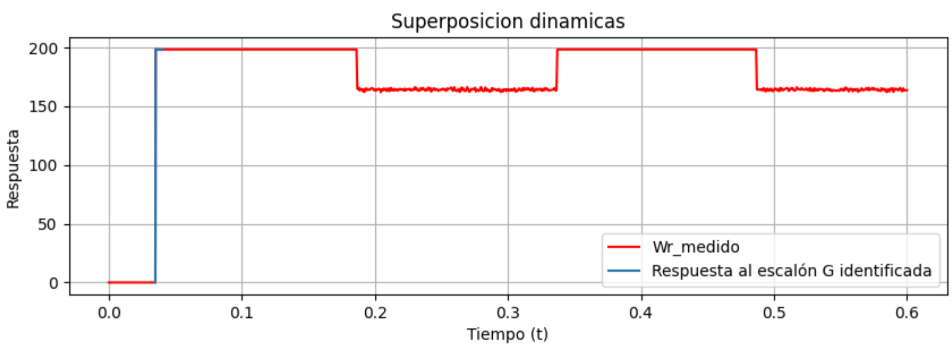

Figura 13: Superposicion dinamicas

El siguiente paso es determinar las constantes del modelo. Partiendo de la función de transferencia:

$$\frac{\omega_r(S)}{V_a(S)} = \frac{K_i}{L_{AA}Js^2 + s(R_aJ + L_{AA}B) + (R_AB + KiKm)}$$

Proponiendo B=0

$$\frac{\omega_r(S)}{V_a(S)} = \frac{K_i}{L_{AA}Js^2 + s(R_AJ) + (KiKm)}$$

La función de transferencia que obtenemos es:

$$\frac{\omega_r(S)}{V_a(S)} = \frac{16{,}5}{2{,}23e - 09s^2 + 8{,}433e - 05s + 1}$$

Ahora teniendo en cuenta el pico de corriente cuando se aplica la tensión de 12v, podemos determinar una corriente de armadura de : $R_A = 27\Omega$

```
Ra=27
K_i=16.5
J=8.433e-05/Ra
L_AA=2.23e-9/J
K_m=1/198

print('Ra: {:.2e}'.format(Ra))
print('K_i: {:.2e}'.format(K_i))
print('J: {:.2e}'.format(J))
print('L_AA: {:.2e}'.format(L_AA))
print('K_m: {:.2e}'.format(K_m))
```

Obteniendo en consola los resultados:

```
Ra: 2.70e+01
K_i: 1.65e+01
J: 3.12e-06
L_AA: 7.14e-04
K_m: 5.05e-03
```

4.4. Controlador PID discreto

Finalmente Debemos implementar un PID en tiempo discreto. Para este caso utilizaremos Octave. Basándonos en los recursos de clase simularemos el controlador y lo ajustaremos para que nuestro proceso actue de una manera deseada. Se plantea una funcion **modmotor.m** que posee las constantes del motor obtenidas.

Su codigo es:

```
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];
end
```

A esta función se la llama desde un bucle en el cual se simula un PID discreto teniendo en cuenta sus variables Kp, Ki, Kd, y el tiempo sampling. Con los valores propuestos en la consigna obtenemos lo siguiente

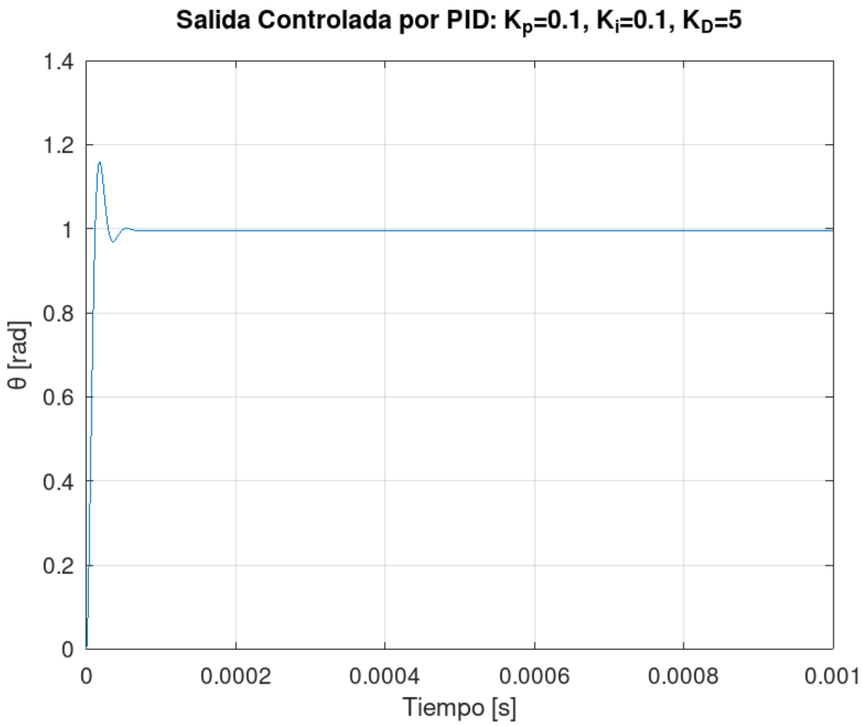

Figura 14: PID con valores de consigna

Podemos observar un sobrepico en la respuesta de aproximadamente un 15 %, lo que puede dar un mal funcionamiento al motor.En lo que respecta al error de estado estacionario en estas condiciones esta por debajo del 1 % como se calcula en el script. Para lograr quitar ese pico, disminuiremos nuestra acción derivativa, esto llevaría a aumentar el ess , es por ello que deberemos aumentar nuestra acción integral para lograr un comportamiento deseado. Utilizando los sliders programados podemos llegar a los siguientes valores:

- $K_p = 1$
- $K_i = 1{,}1$
- $K_d = 1{,}9$
- $ess = 0{,}987\,\%$

```
X = -[0; 0; 0];
index = 0;
h = 1e-7;
ref = 1;
simTime = 1e-3;
Kp=1
Ki=1.1
Kd=1.9
```

```
%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;
theta = zeros(simTime/h,1); % Agrega esta línea para inicializar theta
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); % Almacena el valor de theta en cada paso de tiempo
end
t = 0:h:simTime;
plot(t,theta)
title('Salida Controlada por PID: K_p=0.1, K_i=0.01, K_D=5')
ylabel('\theta [rad]')
grid
```

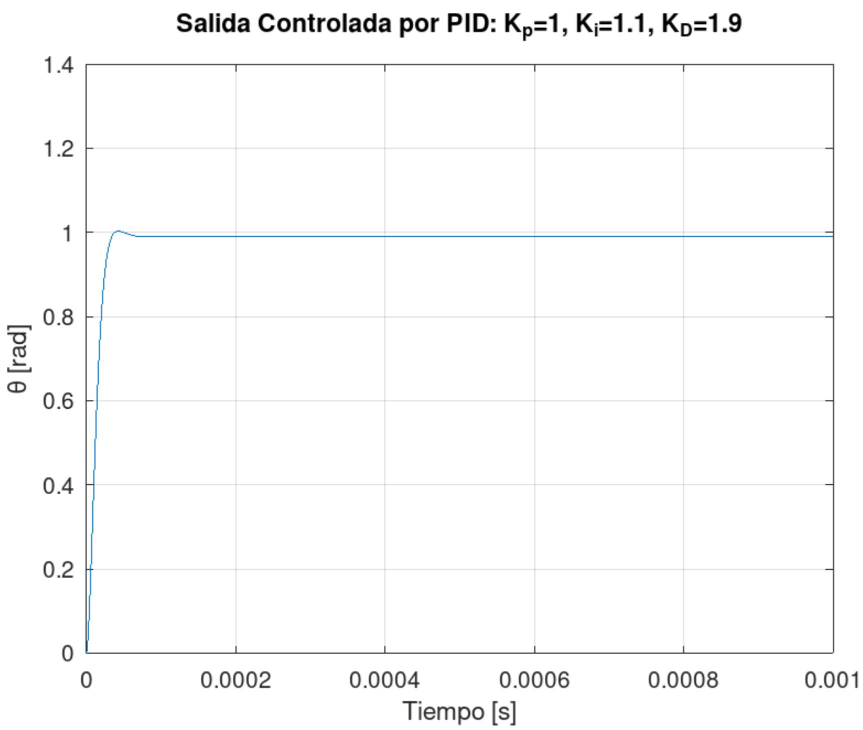

Figura 15: PID Ajustado

5. Observaciones y Logros

5.1. Observaciones

Se detallan cuestiones que fueron de dificultad a lo largo de desarrollo de la actividad:

- El paso t_1 que se debe determinar en el método de Chen es sumamente importante.En la aplicación en ambos casos, en primera instancia los valores que fueron tomados no fueron los ideales , logrando resultados poco esperados.
- Para el uso de Chen, lo ideal es tener conocida la FdT que esperamos del sistema, ya que por ejemplo en el caso del circuito RLC nos da un cero computacionalmente, pero sabemos que en el sistema no existe y debe depreciarse.
- El uso de python y sus diversas librerias que existen en internet facilitó cuestiones de planteo de problemas. Al ser la primera vez que se trabaja seriamente con el lenguaje existieron dificultades al momento de el desarrollo.
- En lo que respecto al PID, se intento plantear la resolucion en base a librerias de python, que no fueron logrados, por eso finalmente se resolvio con octave y el codigo brindado en clases.

5.2. Indicadores de Logros

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

5.3. Enlaces

- `https://github.com/mateooD/SCII_TPs/tree/main/Actividad_1`
- `https://python-control.readthedocs.io/en/0.9.4/conventions.html`
- `https://pythonbasics.org/read-excel/`
- `https://ctan.dcc.uchile.cl/macros/latex/contrib/minted/minted.pdf`
- `https://numpy.org/`
- `https://matplotlib.org/`
- `https://plotly.com/`
- `https://halvorsen.blog/documents/programming/python/resources/Python%20for%20Control%20Engineering.pdf`

6. Conclusiones

Al finalizar este trabajo práctico se entendieron aspectos del modelado en espacio de estados para procesos reales basados en el area de electrónica. El uso del software facilitó la operatoria. Las herramientas open source recomendadas por el docente fueron de mucha utilidad y permitieron el avance progresivo de el trabajo, adquiriendo de forma precisa los conceptos. Se logro poner en práctica y entender los siguientes conceptos:

- Modelado espacio de estados
- Sistemas MIMO
- Controladores tiempo discreto