Alumno: Brian Alex Fuentes Acuña.

Padrón: 101785.

Fecha: 28/9/2021, 2do. Cuatrimestre.

Clase 3: Modelos en variables de estado en tiempo discreto

Los modelos en variables de estado en tiempo discreto pueden provenir de sistemas que son inherentemente sampleados en momentos fijos en el tiempo, o porque al ser llevados al mundo de las computadoras digitales, es la forma más simple de representarlos.

Más adelante veremos cómo llevar modelos de sistemas físicos de tiempo continuo a modelos equivalentes en tiempo discreto, que nos permitirán analizar en el dominio discreto y controlarlos por computadora.

Ecuación de estado en tiempo discreto

Los sistemas de variables concentradas sampleados podemos llevarlos a una forma de ecuaciones en diferencias de primer orden de este estilo:

$$\begin{aligned} x_1(k+1) &= f_1(x_1(k), x_2(k), \ldots, x_n(k), k, u(k)) \\ &\vdots \\ x_n(k+1) &= f_n(x_1(k), x_2(k), \ldots, x_n(k), k, u(k)) \end{aligned}$$

Por simplicidad u la consideramos de dimensión 1, es decir única entrada, para simplificar la notación. Sin embargo, podrían ser múltiples entradas. Estas ecuaciones dependen de funciones arbitrarias $f_1 \ldots f_n$, e indirectamente dependen de las variables $x_1 \ldots x_n$, llamadas variables de estado.

Notamos cómo en este caso bien general el siguiente valor de la variable de estado, depende de los valores actuales de dichas variables, del instante actual de simulación k y de el valor de la entrada (o entradas) en el instante k. Ese conjunto o sistema de ecuaciones en diferencias definen la ecuación de estados, que en forma vectorial se escribe como:

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), u(k), k)$$

Usaremos indistintamente esta notación con paréntesis o directamente con subíndices:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, u_k, k)$$

Donde $\mathbf{x}_k$ es el vector de estado del sistema en el instante k:

$$\mathbf{x}_k = \begin{bmatrix} x_{1\,k} \\ x_{2\,k} \\ \vdots \\ x_{n\,k} \end{bmatrix}$$

Formado por las n variables de estado.

La nomenclatura es similar a los sistemas de tiempo continuo y en muchos casos notaremos que se repetirán los métodos de manera casi directa a través de la notación matricial.

La ecuación de salida tendrá la misma forma, pero estará definida para el instante k:

$$\mathbf{y}_k = \begin{bmatrix} y_{1\,k} \\ \vdots \\ y_{l\,k} \end{bmatrix} = \begin{bmatrix} g_1(x_{1\,k}, \ldots, x_{n\,k}, u_k, k) \\ \vdots \\ g_l(x_{1\,k}, \ldots, x_{n\,k}, u_k, k) \end{bmatrix}$$

Cabe aclarar, que el "instante de tiempo k" corresponder a un tiempo:

$$t_k = kT + t_0$$

Donde:

- T es el período de muestreo
- t_0 es el tiempo inicial, que para sistemas invariantes en el tiempo puede ser arbitrario y por ende en general conviene tomar 0.

Esto es particularmente útil para sistemas que provienen del muestreo periódico de sistemas de tiempo continuo, como se da en los sistemas de control digitales o por computadora.

Caso lineal

Para los sistemas lineales, las funciones f_i y g_j de las ecuaciones de estado, son combinaciones lineales de las variables de estado y de las entradas, y se pueden simplificar como se indica a continuación:

Ecuación de estado

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k$$

Ecuación de salida

$$\mathbf{y}_k = \mathbf{C}_k \mathbf{x}_k + \mathbf{D}_k \mathbf{u}_k$$

Probablemente en el futuro eliminemos la notación en negrita para los vectores y matrices para simplificar, pero deberemos estar atentos a las dimensiones de las variables y constantes en cada caso.

A continuación vamos a repetir el ejemplo de la clase pasada a modo de tarea, pero realizando primero una discretización intuitiva.

Ejemplo:

Con este ejemplo buscamos:

- Mostrar que se puede usar un método similar de asignación de variables de estado de fase a partir de una ecuación en diferencias
- Hallar ecuaciones de estado a partir de ecuaciones en diferencias con $u_{k+1}, u_{k+2}, \ldots$

Ejercicio 4: Para el sistema eléctrico de la Figura 4, con entrada $v(t)$ y salida $v_L(t)$, escribir una única ecuación diferencial que describa al sistema. Luego, encontrar las siguientes descripciones en variables de estado:

a) Variables de estado de fase.

b) $\mathbf{x} = \begin{bmatrix} v_C \\ i_L \end{bmatrix}$

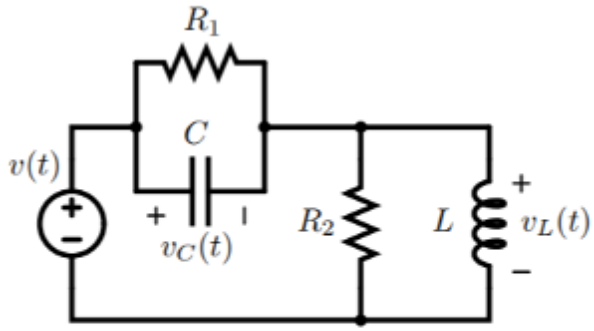


Figura 4: Circuito RLC modificado.

Ejercicio 4

Ejercicio

Primero, volvé al cálculo de la clase pasada para obtener la ecuación diferencial siguiente:

$$\frac{d^2v}{dt^2} + \frac{1}{RC}\frac{dv}{dt} = \frac{d^2v_L}{dt^2} + \frac{2}{RC}\frac{dv_L}{dt} + \frac{v_L}{LC}$$

En donde simplificamos el problema al hacer $R_1 = R_2 = R$.

Verficar que se puede usar una aproximación de primera diferencia para las derivadas:

$$\frac{df}{dt} = \dot{f} \simeq \frac{f(t+\Delta t) - f(t)}{\Delta t}$$
$$\frac{d^2f}{dt^2} \simeq \frac{\dot{f}\,(t+\Delta t) - \dot{f}\,(t)}{\Delta t}$$

para obtener la siguiente ecuación en diferencias:

$$u_{k+2} + b_1 u_{k+1} + b_0 u_k = y_{k+2} + a_1 y_{k+1} + a_0 y_k$$

Donde:

$$a_1 = \frac{2T_S}{RC} - 2$$

$$a_0 = 1 - \frac{2T_S}{RC} + \frac{T_S^2}{LC}$$
$$b_1 = \frac{T_S}{RC} - 2$$
$$b_0 = 1 - \frac{T_S}{RC}$$

Tarea:

Dado el circuito discretizado set toman los siguientes valores:

$$R_1 = R_2 = 20\ \Omega$$
$$C = 5\ \mu\text{F}$$
$$L = 10\ \mu\text{H}$$
$$T_S = 200\ \text{ns}$$

para resolver los siguientes puntos:

1. Hallá una descripción en variables de estado de fase del sistema discretizado. Usar un método similar al de la clase anterior pero con transformada Z. El método también está explicado en el video de la clase teórica.

RTA:

De la ecuacion

$$y_{k+2} + a_1 y_{k+1} + a_0 y_k = u_{k+2} + b_1 u_{k+1} + b_0 u_k$$

se hace un cambio de variables para eliminar las diferencias de la entrada.

$$y(k) = z_{k+2} + b_1 z_{k+1} + b_0 z_k$$

se tienen entonces las dos ecuaciones quedan:

$$z_{k+2} + a_1 z_{k+1} + a_0 z_k = u(k)$$
$$y(k) = z_{k+2} + b_1 z_{k+1} + b_0 z_k$$

luego tomando el termino $z_{k+2} = u(k) - a_1 z_{k+1} - a_0 z_k$ y reemplazando en la expresion de $y(k)$ quedan las ecuaciones de estado:

$$z_{k+2} + a_1 z_{k+1} + a_0 z_k = u(k)$$
$$y(k) = (b_1 - a_1) z_{k+1} + (b_0 - a_0) z_k + u(k)$$

Tomando las variables de estado de fase z_k y z_{k+1}, como entrada u_k y como salida y_k. De esta manera, se obtiene la siguiente representación matricial en variables de estado de fase:

$$\begin{bmatrix} z_{k+1} \\ z_{k+2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix} \begin{bmatrix} z_k \\ z_{k+1} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} (b_0 - a_0) & (b_1 - a_1) \end{bmatrix} \begin{bmatrix} z_k \\ z_{k+1} \end{bmatrix} + u(k)$$

Luego reemplazando los coeficientes se obtiene:

$$\begin{bmatrix} z_{k+1} \\ z_{k+2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -(1 - \frac{2T_S}{RC} + \frac{T_S^2}{LC}) & -(\frac{2T_S}{RC} - 2) \end{bmatrix} \begin{bmatrix} z_k \\ z_{k+1} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$

$$y_k = \begin{bmatrix} \frac{T_s}{RC} - \frac{T_s^2}{LC} & \frac{-T_s}{RC} \end{bmatrix} \begin{bmatrix} z_k \\ z_{k+1} \end{bmatrix} + u(k)$$

En donde **y(k)** es la salida de tension del inductor $v_L(k)$ y entrada **u(k)** es la tesnion de entrada que alimenta al circuito $v(k)$.

(EL cambio de variables de pudo haber hecho transformando primero con laplace, de esa forma queda mas a la vista el por que del cambio de variables)

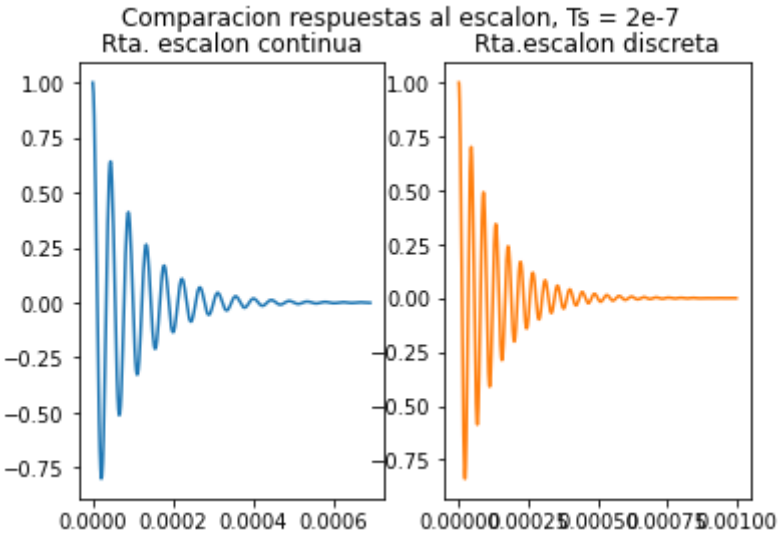
---

3. Simulá el sistema original (circuito) y el discretizado aproximado. Compará respuestas al escalón en el mismo gráfico superpuesto. ¿Qué tan buena es la aproximación? ¿Cómo creés que se puede mejorar la aproximación?

```
!pip install control
import control as ctrl
import numpy as np
import matplotlib.pyplot as plt

R = 20  # Ohm
C = 5e-6  # F
L = 10e-6  # H
Ts = 2e-7 # s. Si te interesa, podrías cambiar el valor y ver lo que sucede con la aproxim

# Definir el sistema en variables de estado
A_d= np.array([ [0, 1], [-(1 - 2*Ts/(R*C) + Ts*Ts/(L*C)), -(2*Ts/(R*C) - 2)] ])
B_d= np.array([ [0], [1] ])
C_d= np.array([Ts/(R*C) - Ts*Ts/(L*C), -Ts/(R*C)])
D_d= np.array([[1]])

#Finalmente el sistema en espacio de estados sera:
SS_d=  ctrl.ss(A_d, B_d, C_d, D_d, Ts)

# Simulación de respuesta al escalón del sistema original

A = np.array([ [0, 1], [-1/(L*C), -2/(R*C)] ])
B = np.array([ [0], [1] ])
C = np.array([ -1/(L*C), -1/(R*C)])
D = np.array([ [1] ])

SS_c = ctrl.StateSpace(A, B ,C, D)
t_c, z_c = ctrl.step_response(SS_c)

# Simulación de respuesta al escalón del sistema discretizado equivalente
t_d, z_d = ctrl.step_response(SS_d)

# Gráfico de ambas respuestas superpuestas en tiempos equivalentes
fig, ((ax1, ax2)) = plt.subplots(1, 2)
fig.suptitle('Comparacion respuestas al escalon, Ts = 2e-7 ')
```

```
ax1.plot(t_c, z_c)
ax1.set_title("Rta. escalon continua")
ax2.plot(t_d, z_d, 'tab:orange')
ax2.set_title("Rta.escalon discreta")
```

```
Collecting control
  Downloading control-0.9.0.tar.gz (339 kB)
     |████████████████████████████████| 339 kB 5.2 MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cy
Building wheels for collected packages: control
  Building wheel for control (setup.py) ... done
  Created wheel for control: filename=control-0.9.0-py2.py3-none-any.whl size=344928
  Stored in directory: /root/.cache/pip/wheels/5b/64/54/70faad181d7baff1184541ca00c9a
Successfully built control
Installing collected packages: control
Successfully installed control-0.9.0
Text(0.5, 1.0, 'Rta.escalon discreta')
```



Se observa que la respuesta es similar en ambos tiempo por lo que el sistema en tiempo discreto es equivalente al continuo y este depende de su tiempo de muestreo

2. Dibujá el diagrama de simulación.

RTA:

El diagrama de simulacion se realizo en el programa Simulink de matlab. por lo que los bloques son distintos a los mostrados en la teorica.

```
from google.colab import files
from IPython.display import Image
```

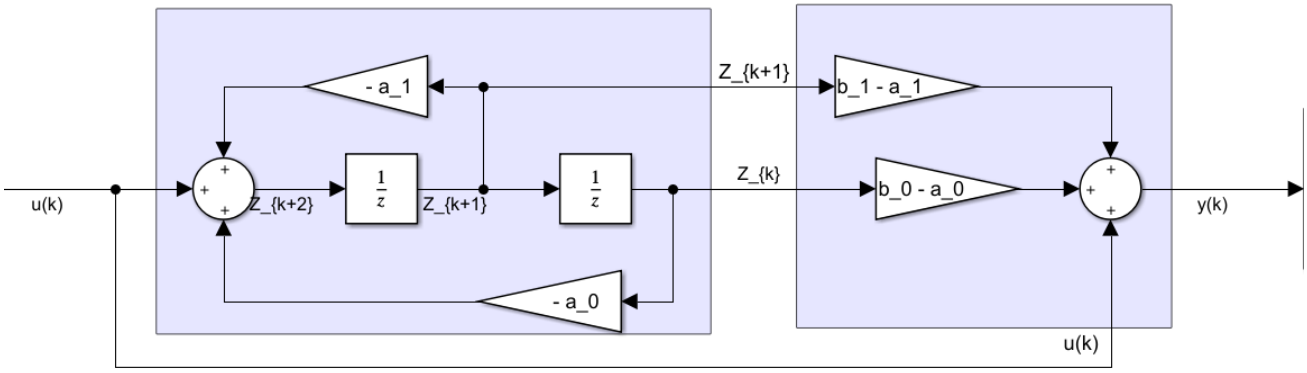
```
uploaded = files.upload()
```

Elegir archivos SIMULINK.PNG

- **SIMULINK.PNG**(image/png) - 31072 bytes, last modified: 28/9/2021 - 100% done

```
Saving SIMULINK.PNG to SIMULINK.PNG
```

```
Image("SIMULINK.PNG", width=800)
```



✓ 7 s se ejecutó 12:53