UCUENCA
INGENIERÍA

Teoría de Control Moderno

Guias de Laboratorio

Autores

Galo Guzmán
galof.guzman@ucuenca.edu.ec

Pablo Andrés Barbecho Bautista
pablo.barbecho@ucuenca.edu.ec

Esta guía presenta una serie de prácticas diseñadas para llevar a cabo durante las horas de APE, con el propósito de reforzar los conocimientos adquiridos en la asignatura. Cada práctica incluye los pasos necesarios para abordar los retos planteados.

Cuenca, Marzo 2024

Tabla de Contenidos

Taller 4: Análisis de sistemas en espacio de estados

1.1 Objetivos

- Integrar conocimientos presentados durante las sesiones de ACD.

1.2 Antecedentes

La asignatura de control moderno contempla horas de APE que se corresponden con el desarrollo de los talleres propuestos durante el curso. Los talleres incluyen manejo de simuladores y desarrollo de los retos propuesto. El estudiante ha de subir el informe del taller en el apartado correspondiente del Evirtual con el desarrollo del reto propuesto.

El presente taller plantea la aplicación de los conceptos revisados en clase sobre los modelos en espacio de estados, en MATLAB. Se analiza la respuesta de los sistemas a diferentes estímulos.

1.3 Desarrollo

1.3.1 Espacio de estados

El espacio de estados es un concepto esencial en disciplinas como la teoría de control, la inteligencia artificial y el análisis de circuitos, entre otras áreas. Se refiere al conjunto completo de estados posibles que un sistema puede adoptar en un momento dado. En el ámbito de la teoría de control, estos estados pueden representar variables como la posición, la velocidad y otras características dinámicas relevantes. Por otro lado, en el análisis de circuitos eléctricos, este concepto permite modelar la evolución de los voltajes y las corrientes en un circuito determinado. A continuación, se presenta un ejemplo visual de un circuito eléctrico que ilustra cómo se aplican los principios del espacio de estados en este contexto.

Considere un circuito RLC en serie, como se muestra en la figura 1.1, el cual será modelado en espacio de estados, donde la variable de salida será el voltaje del capacitor.

Basándose en las ecuaciones de un sistema en espacio de estados, las cuales son:

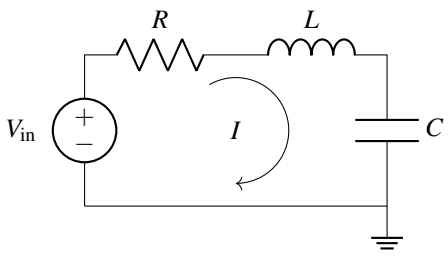

Figure 1.1: Circuito RLC en serie

$$\dot{x}(t) = A \cdot x(t) + B \cdot u(t)$$
$$y(t) = C \cdot x(t) + D \cdot u(t)$$

Es necesario expresar las ecuaciones diferenciales que describen el comportamiento del circuito RLC en función de variables de estado, las cuales, por conveniencia, se denominan como $x_1 = q$ (carga) y $x_2 = \frac{dq}{dt}$ (corriente), se despejan las ecuaciones en función de sus diferenciales.

El modelo en espacio de estados resultante de este circuito es el siguiente:

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix}$$
$$B = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix}$$
$$C = \begin{bmatrix} \frac{1}{C} & 0 \end{bmatrix}$$
$$D = \begin{bmatrix} 0 \end{bmatrix}$$

1.3.2 Espacio de estados en MATLAB

Ejecute el siguiente código en MATLAB:

Código 1 Definición en espacio de estados del circuito RLC

```
% Parámetros del circuito
R = 100    % Resistencia (ohmios)
L = 0.1    % Inductancia (henrios)
Cap = 1e-6   % Capacitancia (faradios)

A = [0  1; -1/(L*Cap) -R/L];       % Matriz de Estado
B = [0; 1/L];                      % Matriz de Entrada
C = [1/Cap 0];                     % Matriz de Salida
D = 0;                             % Matriz de Transferencia directa
```

Ahora, es posible declarar un sistema en espacio de estados usando la función *ss* y posteriormente analizar la respuesta transitoria de este modelo con las funciones *step* e *impulse*, de la siguiente forma:

```
sys = ss(A,B,C,D);

subplot(2,1,1);
step(sys);
xlabel('Time [s]'); ylabel('Vc [V]');
```

```
subplot(2,1,2);
impulse(sys);
xlabel('Time [s]'); ylabel('Vc [V]');
```

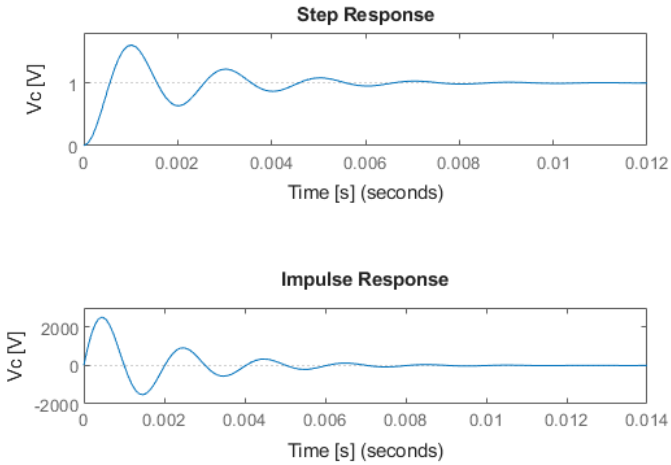

Figure 1.2: Respuesta al escalón e impulso del circuito RLC, usando *step* e *impulse*

La figura 1.2 muestra la respuesta al escalón y a un impulso del circuito RLC, obtenida por medio de su modelo en espacio de estados.

R Es posible también intercambiar los modelos en espacio de estados a función de transferencia y viceversa, usando las funciones *tf* y *ssdata* respectivamente.

1.3.3 Fundamentos de los modelos en espacio de estados

Dado que el espacio de estados es una representación de las ecuaciones diferenciales (o de diferencias) que describen el comportamiento de un sistema, resolver un modelo en espacio de estados equivale, esencialmente, a resolver estas ecuaciones. Este proceso de resolución puede llevarse a cabo mediante diversos métodos, que pueden ser analíticos, como la transformada de Laplace, o numéricos, mediante técnicas como la integración de Runge-Kutta.

En un nuevo script de MATLAB y tomando como base el código 1, declare una función llamada modelRLC, de la siguiente forma:

Código 2 Función del modelo en espacio de estados

```
% Aquí el código para definir el espacio de estados

function dx = modelRLC(t, x, A, B, u)
    dx = A * x + B * u;            % Ecuación de Estado
end
```

La función *modelRLC* realiza los cálculos para el sistema en espacio de estados, definido en las matrices A y B, con la entrada u, pasados como argumento hacia la función. El modelo retorna el valor de dx, que corresponde a los diferenciales de estado. Ya que las funciones de MATLAB deben estar insertadas al final del script, antes de la función inserte el siguiente código:

Código 3 Resolución del modelo usando *ode45*

```
ts = 0.015;                              % Tiempo de simulación
tspan = [0 ts];
u = 1;                                   % Voltaje de entrada
x0 = [0; 0];                             % Condiciones iniciales

[t, X] = ode45(@(t,x) modelRLC(t, x, A, B, u), tspan, x0);

y = C * X.' + D * u;
```

Básicamente este código define la resolución del modelo por medio del método Runge-Kutta con la función *ode45*, la simulación tendrá un tiempo de 15ms, aplicando una fuente de entrada de 1V. Ya que el modelo está en base a la variable de la carga q, para calcular la salida $y = V_c$ de voltaje del capacitor se aplica la ecuación $Vc = \frac{q}{C}$, como se muestra en el código 2. Los resultados se muestran en la figura 1.3.

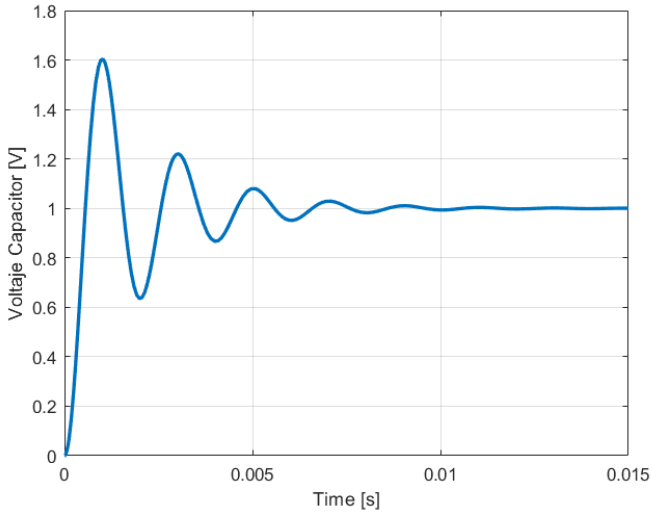

Figure 1.3: Respuesta al escalón del circuito RLC, usando *ode45*

1.3.4 Respuesta a una señal arbitraria

Retomando los conceptos presentados en el taller #2 sobre señales arbitrarias, ahora se plantea la generación de una señal arbitraria y su aplicación al sistema para analizar su respuesta. En este contexto, tomando como base el código del ejercicio anterior, es necesario realizar cambios en la sección del código 2, de forma que permita simular la respuesta del proceso con una señal arbitraria, como se muestra a continuación:

Código 4 Modificación del código 2 para simular con vector de entrada.

```
% Código para la definición del sistema en espacio de estados

% Código para generar la señal arbitraria

...

for k = 2 : N   % N es la dimensión del vector u
    [tk, Xk] = ode45(@(t,x) modelRLC(t, x, A, B, u(k-1)),
    ↪  [tspan(k-1), tspan(k)], X(k-1,:));
    t(k) = tk(end,:);
    X(k,:) = Xk(end,:);
end

...

% Continúa con el cálculo de la salida
```

En el código 4, se observa que ahora u corresponde a un vector con N valores, formado por dos escalones. Debido a esta modificación, es necesario enviar estos valores uno a uno a la función *ode45*, lo que resulta en una integración por tramos. Por lo tanto, se utilizan los valores de `tspan(k-1)` y `tspan(k)` como límites de integración. La repuesta del sistema a esta señal arbitraria, se muestra en la figura 1.4

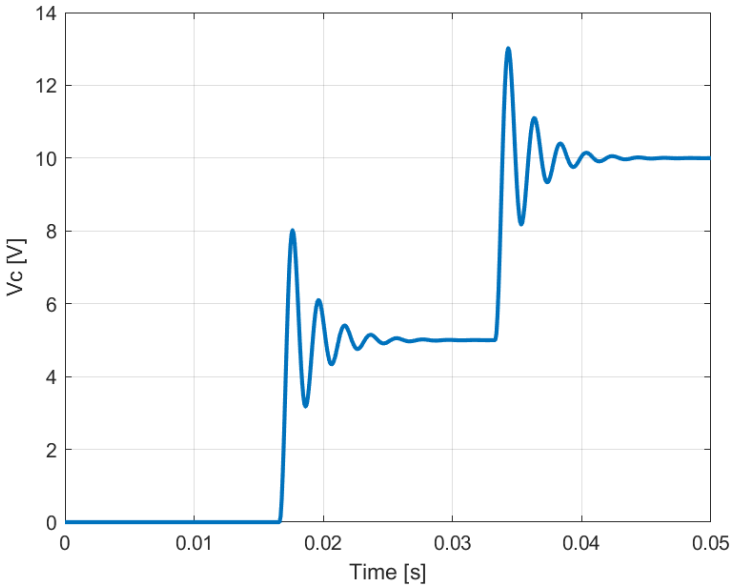

Figure 1.4: Respuesta del sistema a una señal arbitraria

1.4 Actividad Reto

El reto consiste en replicar los ejemplos mostrados en este documento con Python, haciendo uso de la librería *control* y *scipy*. Adicional, utilice un archivo de *csv* el cual contenga los valores de una señal arbitraria generada e inserte la señal al modelo en espacio de estados, de forma similar a como fue planteado en esta guía, reemplazando la entrada u. Documente sus resultados obtenidos y coméntelos en el reporte del taller.