# Taller 4
# Análisis de sistemas en espacio de estados

Jaime Patricio Chiqui Chiqui
Ingenieria en Telecomunicaciones
*Universidad de Cuenca*
Cuenca, Ecuador
jpatricio.chiquic@ucuenca.edu.ec

***Abstract*—Este trabajo presenta el análisis de un sistema eléctrico RLC modelado mediante la representación en espacio de estados. Se desarrolla el procedimiento matemático para obtener las matrices $A$, $B$, $C$ y $D$ a partir de las ecuaciones diferenciales del circuito, permitiendo estudiar su comportamiento dinámico ante diferentes tipos de entrada. Además, se implementa la simulación computacional en MATLAB y Python, utilizando métodos numéricos como *ode45* y la librería *SciPy*, respectivamente. Se obtienen y comparan las respuestas al escalón, al impulso y ante señales arbitrarias, validando la concordancia entre las herramientas utilizadas.**

***Index Terms*—Espacio de estados, sistema RLC, MATLAB, Python, modelado, simulación numérica, sistemas dinámicos.**

## I. Introducción

El análisis de sistemas dinámicos es fundamental en el estudio de la ingeniería eléctrica y de control, ya que permite describir el comportamiento temporal de variables como corriente, voltaje o desplazamiento. Una de las formas más potentes de representar un sistema lineal e invariante en el tiempo (LTI) es el **modelo en espacio de estados**, el cual describe el sistema mediante un conjunto de ecuaciones diferenciales de primer orden que relacionan las variables de estado, la entrada y la salida [1].

Este taller tiene como propósito construir y analizar el modelo en espacio de estados de un circuito RLC serie, comparando los resultados analíticos con simulaciones realizadas en **MATLAB** y **Python**. En cada caso, se determina la respuesta del sistema ante distintas señales de excitación, evaluando su estabilidad y comportamiento transitorio.

## II. Marco teórico

### A. Concepto general del espacio de estados

El análisis de sistemas en el dominio del **espacio de estados** constituye uno de los pilares del control moderno. A diferencia del enfoque clásico basado en funciones de transferencia, el método del espacio de estados describe el sistema mediante un conjunto de ecuaciones diferenciales de primer orden que representan su comportamiento interno [1].

El modelo general se expresa como:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{1}$$

donde:

- $x(t)$: vector de estado que contiene las variables internas del sistema.
- $u(t)$: entrada o señal de excitación.
- $y(t)$: salida del sistema.
- $A$: matriz de estado que representa la dinámica interna.
- $B$: matriz de entrada.
- $C$: matriz de salida.
- $D$: matriz de transmisión directa.

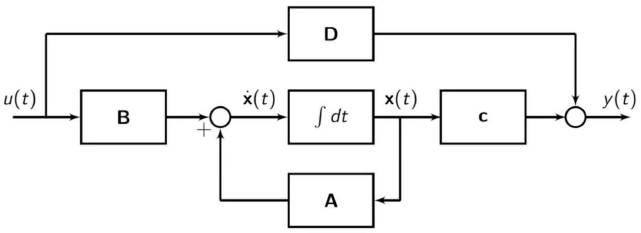


Fig. 1: Representación general de un sistema en espacio de estados [1].

Para analizar el sistema en el dominio de la frecuencia, es común aplicar la **transformada de Laplace** a las ecuaciones del espacio de estados, asumiendo condiciones iniciales nulas. De este modo, el modelo se transforma en:

$$\begin{cases} sX(s) = AX(s) + BU(s) \\ Y(s) = CX(s) + DU(s) \end{cases} \tag{2}$$

Despejando $X(s)$ de la primera ecuación se obtiene:

$$X(s) = (sI - A)^{-1}BU(s)$$

y sustituyendo en la ecuación de salida:

$$Y(s) = C(sI - A)^{-1}BU(s) + DU(s) \tag{3}$$

La ecuación 3 permite derivar directamente la **función de transferencia** del sistema, que relaciona la entrada y la salida en el dominio de Laplace:

$$G(s) = \frac{Y(s)}{U(s)} = C(sI - A)^{-1}B + D \tag{4}$$

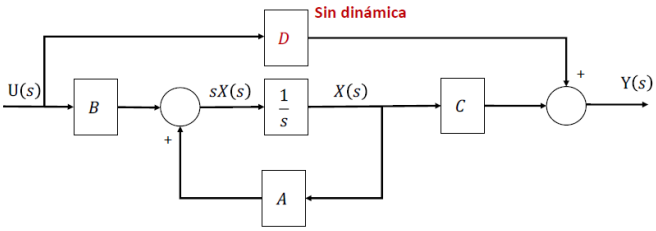


Fig. 2: Diagrama de bloques del modelo en espacio de estados en el dominio de Laplace [1].

El enfoque en espacio de estados permite estudiar el comportamiento interno del sistema, analizar su estabilidad y diseñar controladores incluso para sistemas multivariable (MIMO).

### B. Variables de estado y vector de estados

Las **variables de estado** son aquellas magnitudes físicas que describen completamente el estado dinámico de un sistema en cualquier instante de tiempo. El conocimiento de sus valores en un instante inicial $t_0$, junto con la entrada $u(t)$ para $t \geq t_0$, permite determinar el comportamiento del sistema en todo momento [2].

El conjunto mínimo de variables necesarias para definir el estado se denomina **estado del sistema**. Estas variables se agrupan en un vector de estados:

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} \tag{5}$$

donde cada componente $x_i(t)$ representa una variable de estado independiente.

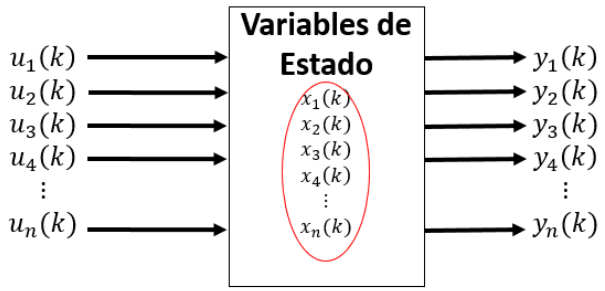


Fig. 3: Representación conceptual de un vector de estados.

### C. Espacio de estados y ecuaciones del sistema

El **espacio de estados** es el conjunto de todos los posibles valores que puede adoptar el vector $x(t)$. Cada punto en este espacio n-dimensional representa una condición particular del sistema.

Las ecuaciones que gobiernan la evolución temporal del estado se denominan **ecuaciones del espacio de estados**, y su forma matricial es la mostrada en la Ecuación 1. Estas ecuaciones permiten describir tanto sistemas eléctricos, mecánicos o térmicos, mediante relaciones lineales entre entradas, estados y salidas.

### D. Pasos para obtener el modelo en espacio de estados

Según la teoría presentada en clase, los pasos para obtener el modelo en espacio de estados de un sistema físico son:

1) **Obtener la ecuación diferencial:** Se parte del modelo físico aplicando leyes de Kirchhoff o Newton según el tipo de sistema.
2) **Despejar la derivada de mayor orden:** Se reorganiza la ecuación diferencial para expresar la derivada de mayor orden en función de las restantes variables.
3) **Definir las variables de estado:** Se seleccionan $n$ variables para convertir la ecuación diferencial de orden $n$ en un conjunto de $n$ ecuaciones de primer orden.
4) **Expresar el sistema en forma matricial:** Finalmente, se escriben las ecuaciones en la forma estándar del espacio de estados:

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

### E. Aplicación al circuito RLC serie

El circuito RLC serie constituye un ejemplo clásico de sistema lineal de segundo orden, conformado por una resistencia $R$, una inductancia $L$ y una capacitancia $C$.

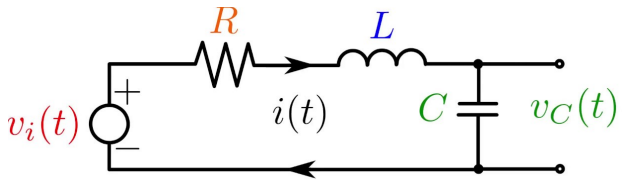


Fig. 4: Circuito RLC serie utilizado para el modelado en espacio de estados.

Aplicando la Ley de Kirchhoff de voltajes, se obtiene:

#### 1. Modelo Matemático del Sistema

$$V_i(t) - Ri(t) - L\frac{di(t)}{dt} - \frac{1}{C}\int i(t)\,dt = 0 \tag{6}$$

#### 2. Derivada de Mayor Orden

$$\frac{di(t)}{dt} = -\frac{1}{LC}\int i(t)\,dt - \frac{R}{L}i(t) - \frac{1}{L}V_i(t) \tag{7}$$

#### 3. Variables de Estado

$$x_1(t) = V_C(t) \tag{8}$$
$$x_2(t) = i(t) \tag{9}$$

**Variables de Estado en Sistemas Eléctricos**

La cantidad de variables de estado en un sistema eléctrico está determinada por el número de **componentes que almacenan energía**. Para circuitos RLC:

- **Capacitores (C):** Almacenan energía en campo eléctrico → 1 variable de estado: $V_C(t)$
- **Inductores (L):** Almacenan energía en campo magnético → 1 variable de estado: $i_L(t)$
- **Resistores (R): NO** almacenan energía → **NO** contribuyen a variables de estado

*4. Ecuaciones de Estado*

$$\dot{x}_1(t) = \frac{1}{C} i(t) = \frac{1}{C} x_2(t) \tag{10}$$

$$\dot{x}_2(t) = \frac{di(t)}{dt} = -\frac{1}{L} x_1(t) - \frac{R}{L} x_2(t) - \frac{1}{L} V_i(t) \tag{11}$$

*5. Representación Matricial*

Usando la ec. 1, se obtiene:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{C} \\ -\frac{1}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} V_i(t) \tag{12}$$

$$V_C(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + 0 \cdot V_i(t) \tag{13}$$

*6. Matrices del Sistema*

$$A = \begin{bmatrix} 0 & \frac{1}{C} \\ -\frac{1}{L} & -\frac{R}{L} \end{bmatrix}; B = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix}; C = \begin{bmatrix} 1 & 0 \end{bmatrix}; D = 0 \tag{14}$$

*7. Función de Transferencia desde Espacio de Estados*

$$G(s) = C(sI - A)^{-1} B + D = \frac{\frac{1}{LC}}{s^2 + \frac{R}{L} s + \frac{1}{LC}} \tag{15}$$

### *F. Modelado computacional*

Las herramientas **MATLAB** y **Python** permiten resolver y simular modelos en espacio de estados mediante integradores numéricos como Runge–Kutta. En MATLAB se emplea `ode45` o la función `ss()`, mientras que en Python se usa `solve_ivp()` de la librería *SciPy* [4], [5].

El modelado en espacio de estados no solo permite describir sistemas físicos mediante ecuaciones, sino también implementarlos en herramientas computacionales para validar su comportamiento teórico.

## III. Desarrollo

### *A. Espacio de estados*

Considere un circuito RLC en serie, como se muestra en la Fig. 4, el cual ya esta modelado en espacio de estados (ec. 13), donde la variable de salida será el voltaje del capacitor. El modelo en espacio de estados resultante de este circuito es el la ec. 14.

### *B. Implementación en MATLAB*

Se realizó la simulación del sistema RLC en MATLAB utilizando las matrices del modelo descrito en la Ecuación 1. El siguiente código implementa el modelo y calcula las respuestas al escalón e impulso haciendo las variables de estado de la carga **q** y se obtuvieron las siguientes matrices **A, B C, D**.

Listing 1: Simulación del sistema RLC en espacio de estados.

```
clc; clear; close all;

R = 100; L = 0.1; Cap = 1e-6;

A = [0 1; -1/(L*Cap) -R/L];
B = [0; 1/L];
C = [1/Cap 0];
D = 0;

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

figure();
subplot(2,1,1); step(sys);
title('Respuesta al Escal n'); grid on;
subplot(2,1,2); impulse(sys);
title('Respuesta al Impulso'); grid on;
```

En la Fig. 5 se muestra la respuesta del sistema al escalón unitario, mientras que la Fig. 6 presenta la respuesta al impulso. Ambas curvas demuestran un comportamiento oscilatorio amortiguado, característico de sistemas de segundo orden.

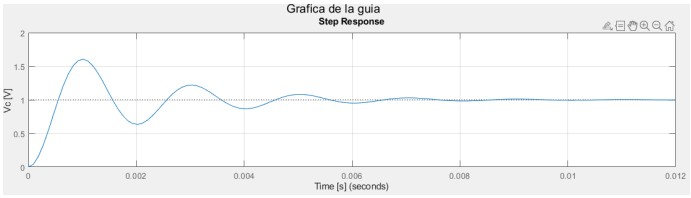


Fig. 5: Respuesta al escalón del sistema RLC.

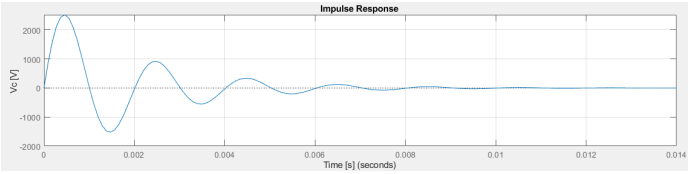


Fig. 6: Respuesta al impulso del sistema RLC.

*1)* ***Comparación:****:* Para esta tarea como comprobación se resolvió haciendo de las ecuaciones diferencias sean ahora las variale de estado, la corriente y el voltaje del capacitor, y no como carga q, por lo que se obtuvo ahora las nuevas matrices, las cuales son **A1, B1, C1, D1**.

Listing 2: Simulación del sistema RLC en espacio de estados.

```
A1 = [-R/L -1/L; 1/Cap 0];    % Matriz de Estado
B1 = [1/L; 0];                % Matriz de Entrada
C1 = [0 1; 1 0];      % Matriz de Salida: [v_c; i]
D1 = 0;              % Matriz de Transferencia directa

sys1 = ss(A1,B1,C1,D1);
```

En la Fig. 7 se muestra la respuesta del sistema al escalón unitario, mientras que la Fig. 8 presenta la respuesta al impulso. Ambas curvas demuestran un comportamiento oscilatorio amortiguado, característico de sistemas de segundo orden.

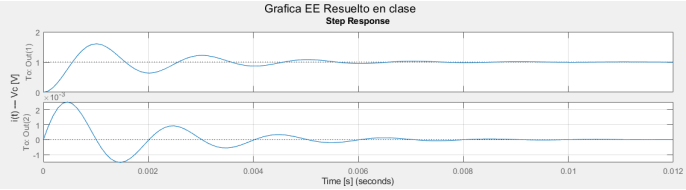


Fig. 7: Respuesta al escalón del sistema RLC.

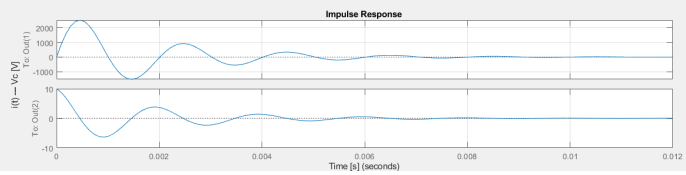


Fig. 8: Respuesta al impulso del sistema RLC.

Por lo tanto, la Fig. 5 y la Fig. 7, son las mismas al igual que la Fig. 6 y la Fig. 8, solo que en las ultimas tambien se mando a graficar la respuesta de la **i(t)** del circuito.

Esta respuesta fue obtenida por medio de su modelo en espacio de estados.

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

***2) Pasar de Espacio de Estados a función de transferencia y viceversa:***

```
%% Pasar de EE a ft y viceversa

% De Espacio de Estados a ft
[num1,den1] = ss2tf(A1,B1,C1(1,:),D1); % v_c(s)/U(s)
[num2,den2] = ss2tf(A1, B1, C1(2,:),D1); % i(s)/U(s)
G1 = tf(num1, den1)
G2 = tf(num2, den2)

% ft a Espacio de Estados
[Ap, Bp, Cp, Dp] = tf2ss(num1, den1)
```

*Salida del Código*

Las funciones de transferencia obtenidas son:

$$G_1(s) = \frac{V_c(s)}{U(s)} = \frac{1e07}{s^2 + 1000s + 1e07}$$

$$G_2(s) = \frac{I(s)}{U(s)} = \frac{10s}{s^2 + 1000s + 1 \times 10^7}$$

La conversión de vuelta a espacio de estados produce:

$$A_p = \begin{bmatrix} -10^3 & -10^7 \\ 0 & 0 \end{bmatrix}, B_p = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, C_p = \begin{bmatrix} 0 & 10^7 \end{bmatrix}, D_p = 0$$

*Comparación con las Matrices Originales*

**Matrices originales del espacio de estados:**

$$A_1 = \begin{bmatrix} -\frac{R}{L} & -\frac{1}{L} \\ \frac{1}{C} & 0 \end{bmatrix} = \begin{bmatrix} -1000 & -10 \\ 1 \times 10^6 & 0 \end{bmatrix}, \quad B_1 = \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} = \begin{bmatrix} 10 \\ 0 \end{bmatrix}$$

$$C_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad D_1 = 0$$

*Análisis de las Diferencias*

Las matrices obtenidas mediante `tf2ss` representan el mismo sistema dinámico pero en una **forma canónica controlable**, mientras que las matrices originales representan una **forma física** basada en las variables de estado naturales del circuito (corriente y voltaje).

Las principales diferencias son:

- **Escalado diferente**: Las matrices $A_p$ y $C_p$ tienen valores escalados por factores de $10^6$ y $10^7$ comparados con las originales
- **Forma canónica**: $A_p$ está en la forma controlable, mientras que $A_1$ refleja la estructura física del circuito RLC
- **Representación mínima**: La conversión produce una representación SISO para $V_c(s)/U(s)$, mientras que el sistema original es MIMO.

**Equivalencia de los sistemas:**

A pesar de las diferencias en las matrices, ambos sistemas son **equivalentes** ya que:

$$C_1(sI - A_1)^{-1}B_1 + D_1 = C_p(sI - A_p)^{-1}B_p + D_p = \frac{1 \times 10^7}{s^2 + 1000s + 1 \times 10^7}$$

La función de transferencia resultante es idéntica, lo que demuestra que representan el mismo comportamiento entrada-salida.

### C. Simulación mediante `ode45`

La ecuación diferencial del sistema puede resolverse numéricamente con el método de Runge–Kutta implementado en la función `ode45` de MATLAB:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{16}$$

El siguiente código muestra la resolución y graficación de la salida del capacitor $V_C(t)$:

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

ts = 0.015; tspan = [0 ts];
u = 1; x0 = [0; 0];

[t, X] = ode45(@(t,x) modelRLC(t, x, A, B, u), tspan
    , x0);
y = C * X.' + D * u;

figure();
plot(t, y);
xlabel('Tiempo [s]'); ylabel('Vc [V]');
title('Respuesta del sistema RLC usando ode45');
grid on;
```

La Fig. 9 muestra la respuesta temporal obtenida por integración numérica, la cual coincide con los resultados teóricos del modelo en espacio de estados (Fig. 5).

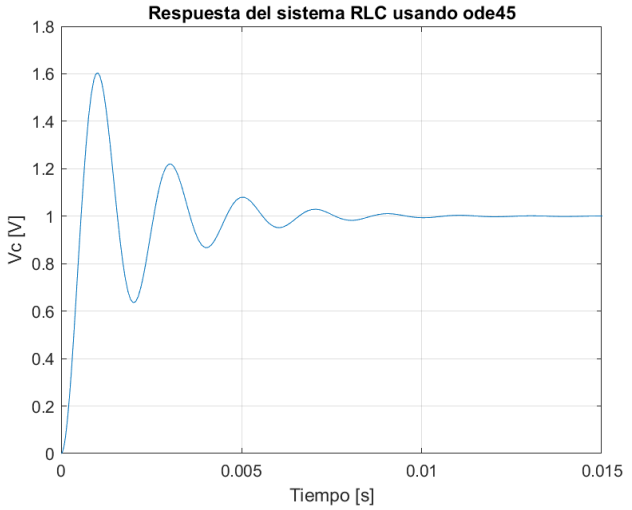


Fig. 9: Respuesta temporal del sistema obtenida con `ode45`.

*1) **Representación en diagrama de bloques mediante simulink**:* Como ya tenemos el diagrama de bloques para el modelo de espacio de estados en el dominio de Laplace, como se aprecia en la Fig. 2, podemos replicar esto en simulink:

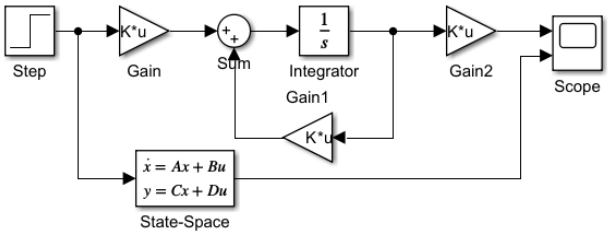


Fig. 10: Diagrama de bloques para obtener la respuesta del sistema en simulink.

- **Bloque Step**:
  - Step time: 0.01
  - Initial value: 0
  - Final value: 1
  - Sample time: 0
- **Bloque Gain (Matriz B)**:
  - Gain: `B1`
  - Multiplication: Matrix (K*u)
- **Bloque Gain1 (Matriz A)**:
  - Gain: `A1`
  - Multiplication: Matrix (K*u)
- **Bloque Gain2 (Matriz C)**:
  - Gain: `C1`
  - Multiplication: Matrix (K*u)
- **Bloque Integrador**:
  - No modificar nada.
- **Bloque State-Space**: Parametros
  - A: `A1`
  - B: `B1`
  - C: `C1`
  - D: `[0;0]`
  - Initial conditions: 0

Para los demás parámetros que no se han puesto es porque se deben dejar por defecto, además vamos a configurar lo siguiente presionando **ctrl + E**:

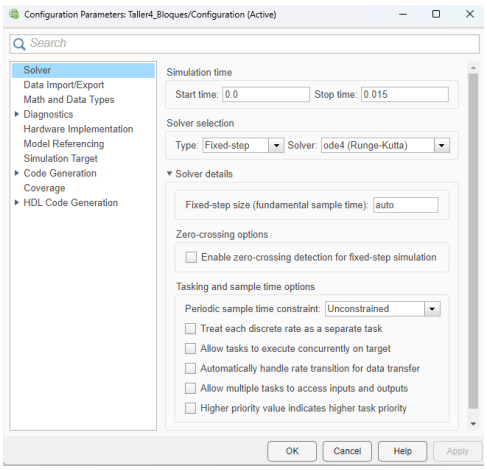


Fig. 11: Configuración para que simule con `ode45`.

Una vez configurado todos los parámetros necesarios, corremos el programa y obtendremos la respuesta del sistema como se aprecia en la Fig. 12 .

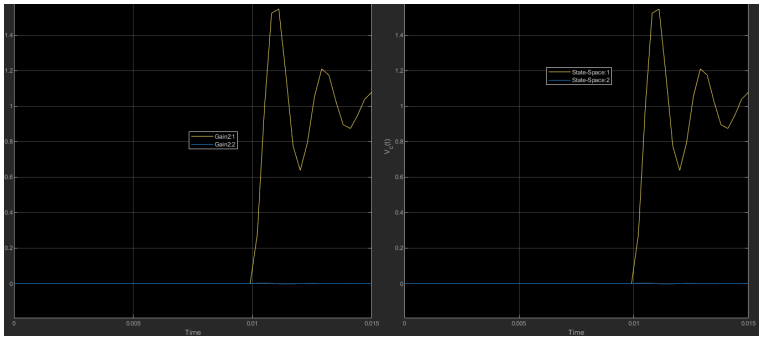

Fig. 12: Salida del sistema con `ode45` en simulink.

En la Fig. 12, ambas implementaciones producen resultados idénticos. Esto demuestra que en Simulink existen dos enfoques equivalentes para modelar sistemas en EE:

- **Diagrama de bloques con ganancias e integradores**: Representación manual que requiere configurar cada matriz por separado
- **Bloque State-Space**: Implementación directa que simplifica el modelo usando las matrices del sistema

Ambos métodos son iguales, pero el bloque State-Space ofrece una implementación más compacta y mantenible.

### D. Respuesta ante señales arbitrarias

Para evaluar la respuesta del sistema ante una señal de entrada no periódica y de comportamiento variado, se empleó una señal compuesta por distintos segmentos: una parte sinusoidal, escalones de amplitud constante y rampas ascendentes y descendentes. Este tipo de señal permite observar el comportamiento dinámico del sistema ante transiciones suaves y abruptas, evaluando su linealidad y capacidad de seguimiento de referencia.

```
%% 4. Respuesta a una se al arbitraria (PRBS)
ts = 0.1;       % Periodo de muestreo
t_total = 40;   % Tiempo total de simulaci n

t1 = 0:ts:10-ts; y1 = sin(2*pi*0.25*t1);
t2 = 10:ts:15-ts; y2 = 3 * ones(size(t2));
t3 = 15:ts:20-ts; y3 = linspace(3, 5, length(t3));
t4 = 20:ts:25-ts; y4 = 5 * ones(size(t4));
t5 = 25:ts:30-ts; y5 = linspace(5, -3, length(t5));
t6 = 30:ts:t_total; y6 = -3 * ones(size(t6));
% Combinar todas las secciones
valores = [y1, y2, y3, y4, y5, y6]';

ts = 0.05;    % Tiempo de simulaci n
x0 = [0; 0]; % Estado inicial
tspan = linspace(0, ts, length(valores));

% Inicializaci n
N = length(valores);
X = zeros(N, 2);
X(1, :) = x0';
t = zeros(N, 1);
t(1) = tspan(1);

for k = 2:N
    u = valores(k);
    [tk, Xk] = ode45(@(t, x) modelRLC(t, x, A1, B1,
        u), [tspan(k-1), tspan(k)], X(k-1, :)');
    t(k) = tk(end, :);
    X(k, :) = Xk(end, :);
end

% C lculo de la salida
y = C1 * X' + D1 * valores';

%% Gr ficas
figure;
subplot(2,1,1);
plot(tspan, valores, 'b');
title('Se al de entrada');
xlabel('Tiempo [s]');
ylabel('Amplitud'); grid on;

subplot(2,1,2);
plot(t, y);
title('Salida del sistema');
xlabel('Tiempo [s]');
ylabel('Voltaje V_c [V]'); grid on;
```

En la Fig. 13 se muestran tanto la señal de entrada compuesta como la salida del sistema. Se observa que el modelo RLC responde con una dinámica amortiguada ante los cambios bruscos de entrada, reproduciendo fielmente la evolución temporal esperada de un sistema de segundo orden.

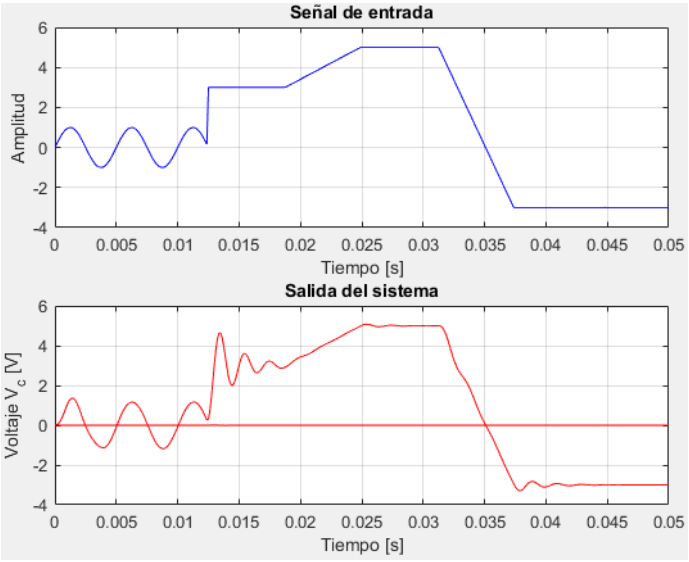


Fig. 13: Respuesta del sistema RLC ante señal arbitraria tipo PRBS.

## IV. Reto

### A. Simulación en Python

**El reto consiste en replicar los ejemplos mostrados en este documento con Python, haciendo uso de la librería control y scipy. Documente sus resultados obtenidos y coméntelos**

Para cumplir con el reto propuesto, se replicó el análisis del sistema RLC en **Python**, utilizando las librerías `control` y `scipy.integrate`.

El código implementa las ecuaciones del espacio de estados del circuito RLC, calcula las respuestas al escalón, impulso y resuelve la ecuación diferencial mediante el método de Runge–Kutta (`solve_ivp`), equivalente al `ode45`.

### B. Resultados obtenidos

En la Fig. 14 se observa la respuesta al escalón unitario del sistema RLC implementado en Python. La curva reproduce el comportamiento oscilatorio amortiguado del circuito, verificando la concordancia con la simulación en MATLAB.

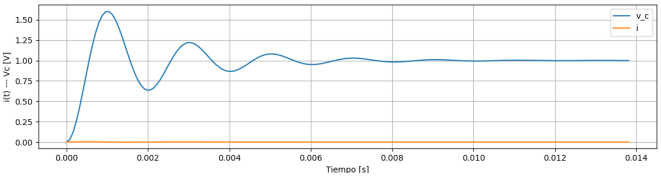

Fig. 14: Respuesta al escalón del sistema RLC implementado en Python.

Del mismo modo, la Fig. 15 muestra la respuesta al impulso, en la cual se aprecia una oscilación transitoria que decae progresivamente hasta el estado estacionario.

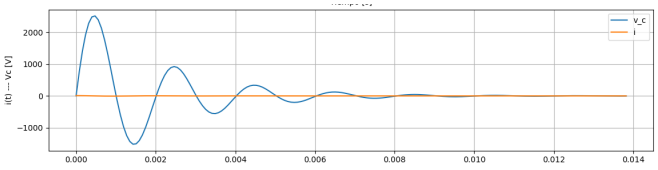


Fig. 15: Respuesta al impulso del sistema RLC en Python.

Posteriormente, se realizó la integración numérica del modelo mediante el método de Runge–Kutta, como se muestra en la Fig. 16. La respuesta temporal del voltaje en el capacitor coincide con la obtenida por la función `step_response`, confirmando la validez del modelo dinámico.

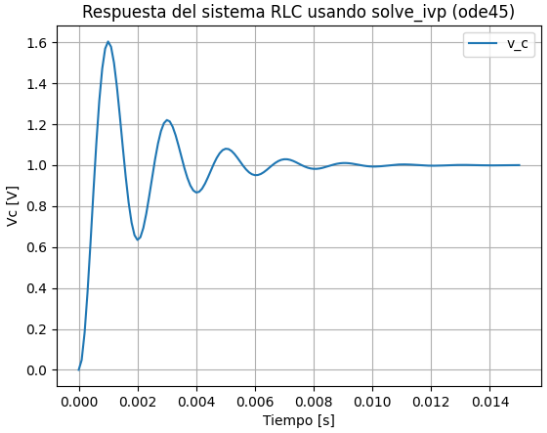


Fig. 16: Respuesta del sistema RLC mediante integración numérica (`solve_ivp`).

### C. Respuesta ante señal arbitraria (PRBS)

Finalmente, se evaluó el comportamiento del sistema ante una señal compuesta por tramos sinusoidales, escalones y rampas, generada para analizar su respuesta ante entradas no periódicas. El código implementado conserva la estructura del modelo MATLAB, empleando un bucle de integración `solve_ivp()` para cada intervalo.

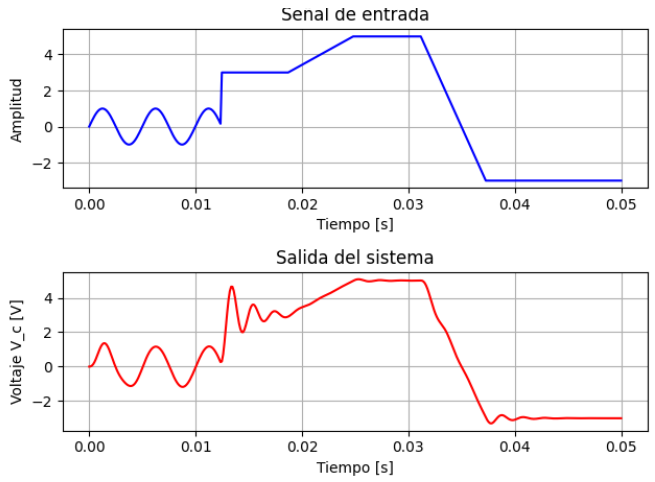


Fig. 17: Respuesta del sistema RLC ante una señal arbitraria (PRBS).

En la Fig. 17 se aprecia la correspondencia entre la señal de entrada y la salida del sistema. El circuito reproduce de forma estable los cambios de entrada, demostrando la linealidad del modelo y su correcta implementación computacional.

Los resultados obtenidos en Python son equivalentes a los simulados en MATLAB, validando tanto la formulación matemática como la implementación numérica del modelo en espacio de estados.

## V. Código de MATLAB

El código que se usó para desarrollar este taller está cargado en **GitHub** como se pidió en diapositiva, en el siguiente enlace se puede obtener el código: Obtener código de Matlab y Obtener Diagrama de bloques - simulink.

Y el código para Python se encuentra en el siguiente enlace: Obtener código de Python

## VI. Conclusiones

- El enfoque de **espacio de estados** permite una descripción integral de sistemas dinámicos complejos, capturando tanto la dinámica transitoria como la de régimen permanente de manera precisa y generalizable.
- Los análisis realizados evidencian que los **circuitos RLC** presentan comportamientos altamente dependientes de sus parámetros ($R$, $L$, $C$) y del tipo de excitación. Esto facilita la comprensión de fenómenos como amortiguamiento, resonancia y estabilidad, esenciales para el diseño de sistemas eléctricos y de control.
- Las **simulaciones computacionales** ofrecen una ventaja significativa sobre los métodos manuales, permitiendo visualizar de forma inmediata la evolución de las variables del sistema ante diferentes entradas y configuraciones. MATLAB se destaca por su rapidez y facilidad gráfica, mientras que Python proporciona flexibilidad y alternativas de código abierto.
- El empleo de **métodos numéricos** como `ode45` y `odeint` demuestra que la resolución de ecuaciones diferenciales puede ser eficiente y confiable, incluso cuando las soluciones analíticas son difíciles o imposibles de obtener.
- El taller refuerza la importancia de integrar la teoría y la práctica mediante herramientas computacionales, consolidando habilidades que son fundamentales para la **ingeniería aplicada**, el diseño de sistemas de control y la simulación de procesos dinámicos complejos.

## References

[1] K. Ogata, *Modern Control Engineering*, 5th ed., Prentice Hall, 2010.
[2] R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 14th ed., Pearson, 2022.
[3] T. Glad and L. Ljung, *Control Theory: Multivariable and Nonlinear Methods*, 2nd ed., Taylor & Francis, 2015.
[4] MathWorks, "State-Space Representation and ode45 Solver," *MATLAB Documentation*, 2024. [Online]. Available: https://www.mathworks.com/help/control/ref/ss.html
[5] SciPy, "Integration and ODE Solvers," *SciPy v1.13.0 Reference Guide*, 2024. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html