Práctica 2

22/03/2024
Fundamentos de Control

Juan López Puebla

Descripción externa e interna

Descripción externa de sistemas

Table of Contents

Tradicionalmente, el trabajo de los ingenieros de control comienza con el desarrollo de un **modelo matemático** de la planta/proceso/sistema dinámico a controlar: un conjunto de ecuaciones que representan la dinámica del sistema con precisión. Estas ecuaciones, independientemente de la naturaleza del sistema, se describen en términos de **ecuaciones diferenciales** obtenidas a partir de **leyes fisicas** que gobiernan dicho sistema (p.ej. leyes de Kirchhoff para sistemas eléctricos o de Newton para sistemas mecánicos), y nos permiten conocer y entender como se comporta nuestro sistema.

La **teoría de control clásica**, centrada en sistemas SISO (Single Input Single Output) invariantes en el tiempo, propone la representación del modelo matemático como **función de transferencia**. Esta es especialmente útil para analizar la respuesat transitoria o la respuesta en frecuencia de este tipo de sistemas, como veremos más adelante.

1. Funciones de transferencia

La función de transferencia $G(s)$ que caracteriza a un sistema se construye a partir de las ecuaciones diferencias que lo modelan, y está definida como la relación entre la **transformada de Laplace** de la salida $Y(s)$ y la transformada de la entrada $U(s)$, con condiciones iniciales (CI) nulas:

$$\text{Función de transferencia} = G(s) = \frac{L\{(y(t))\}}{L\{x(t)\}} = \frac{Y(s)}{X(S)}\Big|_{\text{CI}=0}$$

Y(s)=X(s)*G(s)

De este modo, para el sistema lineal e invariante en el tiempo (LTI)

$a_0 y^{n)} + a_1 y^{n-1} + \cdots + a_{n-1} y' + a_n y = b_0 x^{m)} + b_1 x^{m-1)} + \cdots + b_{m-1} x' + b_m x$ con $n \geq m$ (sistema causal), se tiene que:

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \cdots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n}$$

donde:

- n (potencia más alta del denominador, también llamado **ecuación característica**) es el **orden** del sistema.
- Las raíces del denominador son los **polos** del sistema.
- Las raíces del numerador son los **ceros** del sistema.

Dicho de otro modo, la función de transferencia permite representar nuestro sistema,originalmente expresado en el dominio del tiempo en forma de ecuaciones diferenciales cuya manipulación puede ser compleja, por medio de una nueva variable compleja s en forma de ecuaciones algebráicas, empleando para ello la transformada de Laplace.

A esta representación se la conoce como **descripción externa** del sistema, ya que no acarrea ninguna información sobre la estructura interna o estructura física del mismo.

Tarea 1: Dada la siguiente función de transferencia $G(s) = \frac{s^2 + 2s + 1}{s^3 + 4s + 1}$, obten sus ceros y sus polos. ¿Cuál es el orden del sistema?

*Pista: Recuerda el cuaderno sobre polinomios y el comando **`roots`**.*

```
% Tu código aquí
num = [1 2 1];
den = [1 0 4 1];
%obtenemos los polos
polos = roots(den)
```

```
polos = 3×1 complex
   0.1231 + 2.0113i
   0.1231 - 2.0113i
  -0.2463 + 0.0000i
```

```
%obtenemos los ceros
ceros = roots(num)
```

```
ceros = 2×1
    -1
    -1
```

```

```

Resultado esperado:

```
ceros = 2×1
     -1
     -1
```

```
polos = 3×1 complex
     0.1231 + 2.0113i
     0.1231 - 2.0113i
    -0.2463 + 0.0000i
```

2. Creación de modelos de sistemas con formato de funciones de transferencia.

MATLAB nos brinda la **Control System Toolbox**, un conjunto de herramientas para trabajar con modelos lineales de sistemas y su control. La construcción de una función de transferencia de un sistema se puede abordar por tres caminos diferentes, con la función tf, el comando zpk,o de forma directa. Veamos cada una de ellas.

2.1 tf

La primera manera de definir funciones de transferencia es mediante el comando **`tf`**. Su sintaxis es:

- *`Sys = tf( num, den )`*: donde *num* y *den* son, respectivamente, los vectores de coeficientes de los polinomios del numerador y denominador de la función de transferencia.

Tras la ejecución de esta función, la función de transferencia se muestra en pantalla como un cociente de polinomios en s no factorizados (a partir de ahora a este formato se le denominará formato tf).

Ejemplo:

Dado un sistema cuya función de transferencia es:

$$G(s) = \frac{2s + 2}{s^2 + 3s + 5}$$

Su modelo es utilizable por MATLAB mediante una variable `G` creada tal que así:

```
num = [2 2]; % Numerador: 2s + 2
den = [1 3 5]; % Denominador s^2 + 3s + 5
G = tf(num,den)
```

```
G =

     2 s + 2
  -------------
  s^2 + 3 s + 5

Continuous-time transfer function.
Model Properties
```

Tarea 2: Sea el sistema definido por $G(s) = \frac{14s + 140}{s^3 + 9s^2 + 33s + 65}$, obtener su función de transferencia en MATLAB usando el comando **`tf`**.

```
% Tu código aquí
num = [14 140];
```

```
den = [1 9 33 65];
G = tf(num,den)
```

```
G =

        14 s + 140
  ----------------------
  s^3 + 9 s^2 + 33 s + 65

Continuous-time transfer function.
Model Properties
```

```

```

Resultado esperado:

```
G =

        14 s + 140
  ----------------------
  s^3 + 9 s^2 + 33 s + 65

Continuous-time transfer function.
```

Tarea 3: Sea el sistema definido por $G(s) = \frac{14(s+10)}{(s+5)(s+2-3i)(s+2+3i)}$, obtener su función de transferencia en MATLAB usando el comando **`tf`**. Comprueba que el resultado coincide con la función de transferencia de la Tarea 2.

*Pista: Aquí el comand **conv** puede ser útil.*

```
% Tu código aquí
 num = 14*[1 10]
```

```
num = 1×2
    14   140
```

```
 u = [1 5];
 v = [1 2-3i];
 g = [1 2+3i];
 den = conv(conv(u,v),g)
```

```
den = 1×4
     1     9    33    65
```

```
G = tf(num,den)
```

```
G =

        14 s + 140
  ----------------------
  s^3 + 9 s^2 + 33 s + 65

Continuous-time transfer function.
Model Properties
```

2.2 zpk

El segundo método de definición de funciones de transferencia es mediante el comando **zpk**. Su sintaxis es:

- *Sys = zpk(ceros, polos, k):* donde *ceros* y *polos* son, respectivamente, vectores que contienen a todos los ceros y polos de la función de transferencia y *k* es el valor de la ganancia homónima, en el caso en el que la función de transferencia se exprese en la forma:

$$\mathrm{Sys}(s) = \frac{k\prod_{j=1}^{j=m}(s-\mathrm{cero}_j)}{\prod_{i=1}^{i=n}(s-\mathrm{polo}_i)}$$

Si la función de transferencia considerada no tiene polos o ceros, el correspondiente vector deberá especificarse como vacío: *[ ]*. Tras la ejecución de esta función, la función de transferencia se muestra en pantalla como un cociente de polinomios en s factorizados (a partir de ahora a este formato se le denominará formato zpk).

Ejemplo:

Dado un sistema cuya función de transferencia es:

$$G(s) = \frac{2(s+2)}{(s+3)(s+5)}$$

Su modelo es utilizable por MATLAB mediante una variable G creada así:

```
z = -2;
p = [-3 -5];
k = 2;
G = zpk(z,p,k)
```

```
G =

    2 (s+2)
  -----------
  (s+3) (s+5)

Continuous-time zero/pole/gain model.
Model Properties
```

```
% O directamente
G = zpk( -2, [ -3 -5 ], 2 )
```

```
G =

    2 (s+2)
  -----------
  (s+3) (s+5)

Continuous-time zero/pole/gain model.
Model Properties
```

Tarea 4: Sea el sistema definido por $G(s) = \frac{14(s+10)}{(s+5)(s+2-3i)(s+2+3i)}$, obtener su función de transferencia en MATLAB usando el comando **zpk**.

```
% Tu código aquí
z = -10;
p = [-5 -2+3i -2-3i];
k = 14;
G = zpk(z,p,k)
```

```
G =

        14 (s+10)
  ---------------------
  (s+5) (s^2 + 4s + 13)

Continuous-time zero/pole/gain model.
Model Properties
```

Resultado esperado:

```
G =

        14 (s+10)
  ---------------------
  (s+5) (s^2 + 4s + 13)

Continuous-time zero/pole/gain model.
```

Tarea 5: Sea el sistema definido por $G(s) = \frac{14s+140}{s^3+9s^2+33s+65}$, obtener su función de transferencia en MATLAB usando el comando **zpk**. El resultado ha de ser el mismo que en la Tarea 4.

*Pista: Aquí **roots** puede ser útil de nuevo.*

```
% Tu código aquí
num = [14 140];
den = [1 9 33 65];
z = roots(num)
```

```
z = -10
```

```
p = roots(den)
```

```
p = 3×1 complex
  -5.0000 + 0.0000i
  -2.0000 + 3.0000i
  -2.0000 - 3.0000i
```

```
G = zpk(z,p,14)
```

```
G =

        14 (s+10)
  ---------------------
```

```
  (s+5) (s^2 + 4s + 13)

Continuous-time zero/pole/gain model.
Model Properties
```

2.3 De forma directa

Si la función de transferencia considerada tiene polos y/o ceros, el valor de la función de transferencia podrá asignarse directamente, como un cociente de polinomios expresado en su formato algebraico ordinario y cuya variable independiente se denomina s, si con anterioridad se ejecuta una cualquiera de las dos siguientes instrucciones:

- *s = tf('s')*
- *s = zpk('s')*

La única diferencia que se observaría tras la ejecución de una de estas dos funciones está en el formato con que MATLAB mostraría las funciones de transferencia que fueran creadas a continuación. Si se opta por la primera opción, se mostrarían con formato `tf` y si se opta por la segunda, se mostrarían con formato `zpk`.

Ejemplo 1:

Dado un sistema cuya función de transferencia es:

$$G(s) = \frac{2s+2}{s^2+3s+5}$$

Su modelo es utilizable por MATLAB mediante una variable G creada así:

```
s = tf('s');
G = (2*s+2)/(s^2+3*s+5)
```

```
G =

    2 s + 2
  -------------
  s^2 + 3 s + 5

Continuous-time transfer function.
Model Properties
```

Ejemplo 2:

Dado un sistema cuya función de transferencia es:

$$G(s) = \frac{2(s+2)}{(s+3)(s+5)}$$

Su modelo es utilizable por MATLAB mediante una variable G creada así:

```
s = zpk('s');
G = 2*(s+2)/((s+3)*(s+5))
```

```
G =
```

```
    2 (s+2)
  -----------
  (s+3) (s+5)

Continuous-time zero/pole/gain model.
Model Properties
```

```
% Incluso aunque lo introduzcamos con "formato" tf
G = (2*s+4)/(s^2+8*s+15)
```

```
G =

    2 (s+2)
  -----------
  (s+3) (s+5)

Continuous-time zero/pole/gain model.
Model Properties
```

¡OJO! Nótese que en función de la forma en la que venga expresada la función de transferencia, puede ser más interesante definirla usando el comando `tf`, `zpk`, o de forma directa.

Tarea 6: Sea el sistema definido por $G(s) = \frac{14s + 140}{s^3 + 9s^2 + 33s + 65}$, obtener su función de transferencia en MATLAB de forma directa, esto es, sin emplear los comandos **`tf`** ni **`zpk`**, de tal manera que el sistema resultante esté expresado en formato `zpk`. Comprobar que el resultado es el mismo que en la Tarea 5.

```
% Tu código aquí
s = zpk('s');
num = roots([14 140])
```

```
num = -10
```

```
den = roots([1 9 33 65])
```

```
den = 3×1 complex
  -5.0000 + 0.0000i
  -2.0000 + 3.0000i
  -2.0000 - 3.0000i
```

```
G = 14*(s+10)/((s+5)*(s+2-3i)*(s+2+3i))
```

```
G =

       14 (s+10)
  ---------------------
  (s+5) (s^2 + 4s + 13)

Continuous-time zero/pole/gain model.
Model Properties
```

```

```

3. Modificación del formato de presentación de funciones de transferencia

Una vez que una función de transferencia esté definida en el espacio de trabajo de MATLAB, se puede modificar el formato con el que ésta se muestra en su ventana de comandos:

- Si se desea pasar de formato zpk a formato tf, se ha de hacer:

```
Sys = tf( Sys )
```

- Si se desea pasar de formato tf a formato zpk, se ha de hacer:

```
Sys = zpk( Sys )
```

Tarea 7: Convierte la función de transferencia en formato `zpk` de la Tarea 5 a formato `tf` y almacenalo en una nueva variable `G2`. Tras esto, convierte `G2` de nuevo a formato `zpk`. El resultado ha de ser el mismo que el obtenido en la Tarea 6.

```
% Tu código aquí

G2 = tf (G)
```

```
G2 =

        14 s + 140
  -----------------------
  s^3 + 9 s^2 + 33 s + 65

Continuous-time transfer function.
Model Properties
```

```
G2 = zpk(G2)
```

```
G2 =

        14 (s+10)
  --------------------
  (s+5) (s^2 + 4s + 13)

Continuous-time zero/pole/gain model.
Model Properties
```

4. Representación del diagrama de polos y ceros de un sistema (pzmap).

Una vez que una variable contiene el modelo de un sistema, es posible generar una representación gráfica en la que se muestre la ubicación en el plano complejo de los polos y ceros del referido modelo. Dicha representación gráfica se denomina diagrama de polos y ceros del sistema. La función de MATLAB que permite obtener el diagrama de polos y ceros de un sistema a partir de su modelo es **`pzmap`**. Esta función es utilizable de forma independiente del formato en que esté expresado el modelo del sistema en cuestión (*`tf`* o *`zpk`*).

Para obtener el diagrama de polos y ceros de un sistema cuyo modelo está almacenado en una variable denominada *`Sys`*, se ha de hacer:

```
pzmap( Sys )
```

El diagrama obtenido se muestra en una ventana independiente de la ventana principal del paquete MATLAB. Los polos son representados en dicho diagrama empleando un aspa (‘x’) y los ceros son representados mediante una circunferencia (‘o’).

Ejemplo:

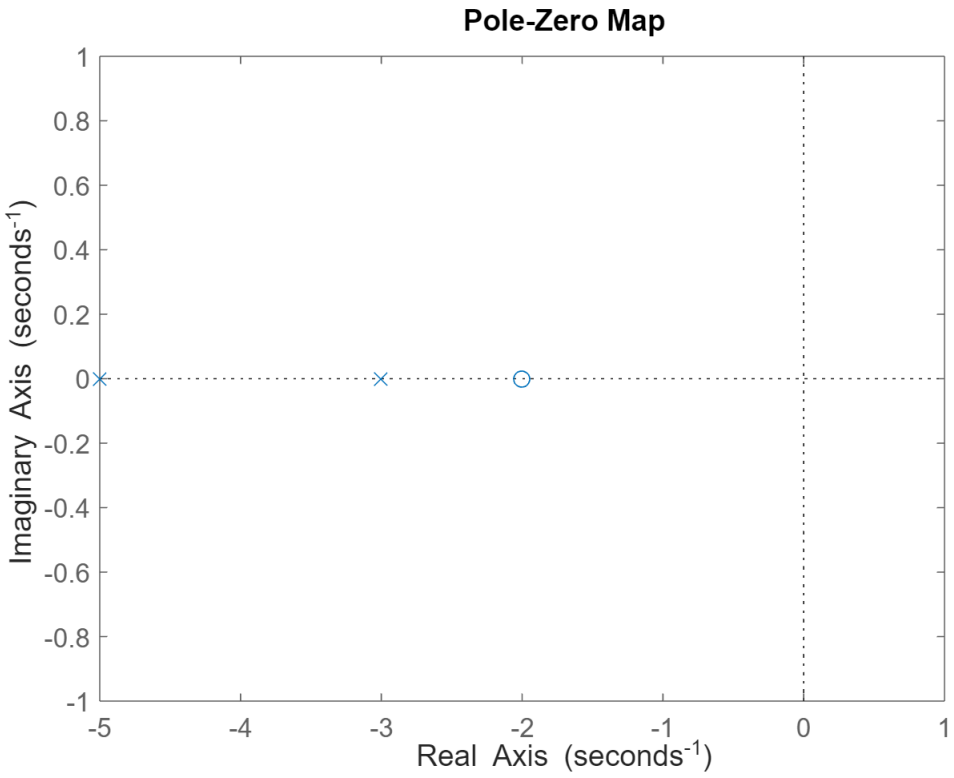
Dibujemos los polos y los ceros de la función:

$$H(s) = \frac{2(s+2)}{(s+3)(s+5)}$$

```
s = zpk('s');
H = 2*(s+2)/((s+3)*(s+5))
```

```
H =

    2 (s+2)
  -----------
  (s+3) (s+5)

Continuous-time zero/pole/gain model.
Model Properties
```

```
pzmap(H)
```



Tarea 8: Obtén la representación de zeros y polos del sistema empleado en la Tarea 7.

```
% Tu código aquí
```

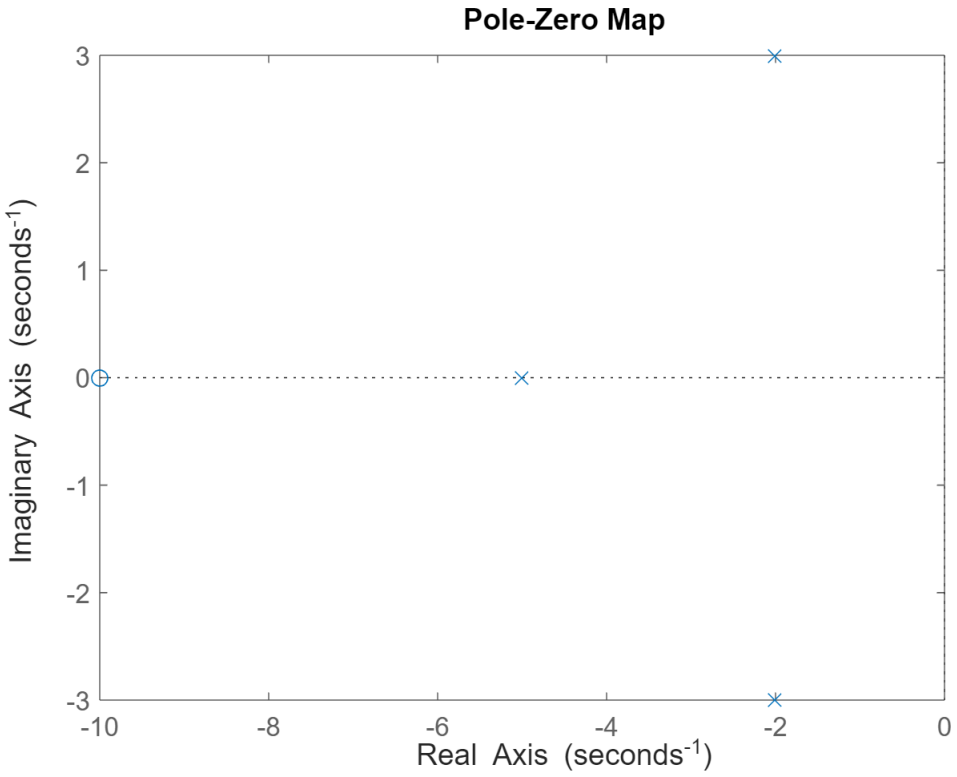
```
s = zpk('s')
```

```
s =

  s

Continuous-time zero/pole/gain model.
Model Properties
```

```
G2
```

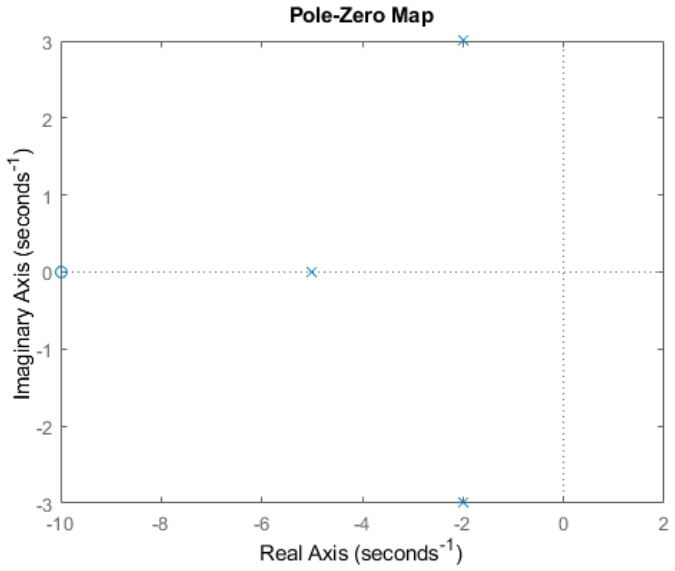
```
G2 =

        14 (s+10)
  ---------------------
  (s+5) (s^2 + 4s + 13)

Continuous-time zero/pole/gain model.
Model Properties
```

```
pzmap(G2)
```




```

```

Resultado esperado:



5. Acceso a los datos de un modelo de sistema (tfdata, zpkdata).

Una vez que una variable contiene el modelo de un sistema, es posible acceder a los correspondientes datos del mismo. Independientemente del formato empleado en la creación de dicho modelo, es posible acceder a los datos característicos que el mismo tendría en caso de ser expresado con un formato diferente.

- Si se desea conocer, con formato *tf*, los datos de un modelo de un sistema almacenado en una variable denominada *Sys*, se ha de hacer: *[num, den] = tfdata(Sys,'v').* Ejemplo:

```
% La v indica que la salida sean vectores en vez de celdas
[num,den] = tfdata(H,'v')
```

```
num = 1×3
         0    2.0000    4.0000
den = 1×3
    1.0000    8.0000   15.0000
```

- Si se desea conocer, con formato *zpk*, los datos de un modelo de un sistema almacenado en una variable denominada *Sys*, se ha de hacer: *[z,p,k] = zpkdata(Sys,'v')*. Ejemplo:

```
[z,p,k]=zpkdata(G,'v')
```

```
z = -10.0000
p = 3×1 complex
  -5.0000 + 0.0000i
  -2.0000 + 3.0000i
  -2.0000 - 3.0000i
k = 14
```

- Si se desea conocer sólo los ceros de un modelo de un sistema almacenado en una variable denominada *Sys*, se ha de hacer: *z = zero(Sys)*. Ejemplo:

```
z = zero(H)
```

```
z = -2.0000
```

```
z = zero(G)
```

```
z = -10.0000
```

- Si se desea conocer sólo los polos de un modelo de un sistema almacenado en una variable denominada *Sys*, se ha de hacer: *p = pole(Sys)*. Ejemplo:

```
p = pole(H)
```

```
p = 2×1
   -3.0000
   -5.0000
```

Tarea 9: De un sistema se conoce que su modelo ha sido asignado a una variable H así:

```
H = tf([2 2],[1 3 5])
```

```
H =

     2 s + 2
  -------------
  s^2 + 3 s + 5

Continuous-time transfer function.
Model Properties
```

Obtener, con formato *tf*, los datos del modelo del sistema.

Pista: ¡Recuerda la famosa 'v'!

```
% Tu código aquí
[num,den] = tfdata(H,'v')
```

```
num = 1×3
     0     2     2
den = 1×3
     1     3     5
```

```

```

Resultado esperado:

```
num = 1×3
       0     2     2

den = 1×3
       1     3     5
```

Tarea 10: Obtener, con formato zpk, los datos del modelo del sistema.

```
% Tu código aquí
```

```
[z,p,k]=zpkdata(H,'v')
```

```
z = -1
p = 2×1 complex
  -1.5000 + 1.6583i
  -1.5000 - 1.6583i
k = 2
```

```

```

Resultado esperado:

```
z = -1
p = 2×1 complex
    -1.5000 + 1.6583i
    -1.5000 - 1.6583i
k = 2
```

Tarea 11: Obtener sólo los ceros del modelo del sistema:

```
% Tu código aquí
z = zero(H)
```

```
z = -1
```

```

```

Resultado esperado:

```
z = -1
```

Tarea 12: Obtener sólo los polos del modelo del sistema.

```
% Tu código aquí

z = pole(H)
```

```
z = 2×1 complex
  -1.5000 + 1.6583i
  -1.5000 - 1.6583i
```

Resultado esperado:

```
z = 2×1 complex
    -1.5000 + 1.6583i
    -1.5000 - 1.6583i
```

Descripción externa e interna

Descripción interna de sistemas

Table of Contents

La **teoría de control clásica** trabaja con sistemas SISO y LTI, emplea métodos de respuesta en frecuencia y el lugar de las raíces, y conduce a sistemas estables que satisfacen un conjunto de requisitos pero que no son óptimos desde ningún punto de vista.

No obstante, las plantas modernas presentan cada vez una mayor complejidad, debido a que realizan cada vez tareas más complejas y de mayor precisión. La necesidad de controlar este tipo de plantas, que típicamente presentan multiples entradas y salidas (MIMO), y pueden ser no lineales y variantes en el tiempo, junto con la diponibilidad de computadoras digitales, impulsó allá por 1960 el desarrollo de la **teoría de control moderna**. Esta teoría considera la **descricpción interna** de los sistemas a controlar en lugar de la descripción externa empleada por la teoría clásica. Veamos en qué consiste.

1. Modelado en el Espacio de Estados

La descripción interna de sistemas se basa en el concepto de **estado**: conjunto más pequeño de **variables** (llamadas **de estado**) de forma que si se conoce su valor en $t = t_0$ y la(s) entrada(s) al sistema para $t \geq t_0$, estos determinan completamente la evolución de dichas variables de estasdo y la(s) funcione(s) de salida mediante las ecuaciones:

- Ecuación de estado: $x'(t) = f(x(t), u(t))$
- Ecuación de salida: $y(t) = g(x(t), u(t))$

donde:

- x es **vector de estado**, el cual contiene las recién presentadas variables de estado, y refleja el estado del sitema en un instante de tiempo,
- $f(\cdot)$ es la **función de transición**, la cuál permite conocer la evolución en función del estado actual y la entrada, y
- $g(\cdot)$ es la **función de salida**, que permite conocer la salida del sistema en función del estado y de su entrada.

Aunque se puede definir un espacio de estados que no tenga una relación evidente con el funcionamiento del sistema, es conveniente que las variables de estado tengan un significado físico. **¡OJO!** Una de las principales virtudes de la descripción interna es realmente esta. Al trabajar con la función de transferencia, la descripción externa "sólo" aporta informacón sobre la relación entre la transformadas de Laplace de la salida y de la

entrada del sistema. Por contra, la descripción interna nos permite trabajar con otras variables de interés. Ya veremos ejemplos de esto.

Si un sistema es lineal, las ecuaciones de estado pueden expresarse de la siguiente forma:

- Ecuación de estado: $x'(t) = Ax(t) + Bu(t)$
- Ecuación de salida: $y(t) = Cx(t) + Du(t)$

Además, si el sistema tiene n variables de estado, p entradas y m salidas, entonces:

- A, también llamada **matriz dinámica** (o de **estado**), es de dimensión $n\,x\,n$.
- B, también llamada **matriz de control** (o de **entrada**), es de dimensión $n\,x\,p$.
- C, también llamada **matriz de salida**, es de dimensión $m\,x\,n$.
- D, también llamada **matriz de influencia directa**, es de dimensión $m\,x\,p$.

Cabe destacar que cualquier estado puede representarse mediante un punto en el **espacio de estados**: espacio n-dimensional donde cada una de las $n$variables de estados se correspondonde con un eje de coordenadas.

1.1 El comando ss

MATLAB nos facilita el trabajo con la representación interna de sistemas con el comando **ss**. Su sintaxis es:

- *sys = ss(A,B,C,D):* Crea un modelo de espacio de estado representando el modelo continuo visto anteriormente, esto es, mediante las matrices *A*, *B*, *C*, y *D,* que definen las ecuaciones de estado y de salida. Ejemplo:

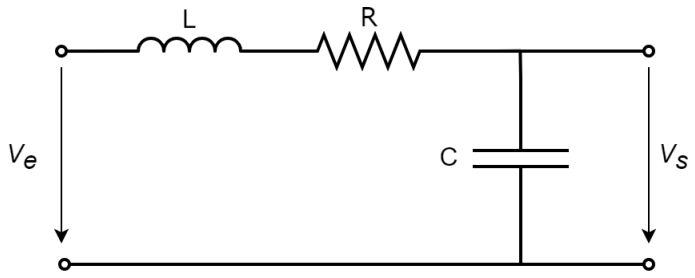
```
A = [0 1;-5 -2]; % Matriz dinámica o de estado
B = [0;3];       % Matriz de control o de entrada
C = [0 1];       % Matriz de salida
D = 0;           % Matriz de influencia directa
sys_ss = ss(A,B,C,D)
```

```
sys_ss =

  A =
       x1  x2
   x1   0   1
   x2  -5  -2

  B =
       u1
   x1   0
   x2   3

  C =
       x1  x2
   y1   0   1

  D =
       u1
   y1   0
```

```
Continuous-time state-space model.
Model Properties
```

1.2 Ejemplo descripción interna: Circuito LRC serie

El siguiente esquema muestra un circuito LRC típico compuesto por una bobina, una resistencia y un condensador. En este circuito, la entrada al sistema es la tensión de entrada V_e, y la de salida es la caída de tensión en el condensador, V_s. Es decir, nuestro circuito cuenta con una tensión de entrada variable, y estamos interesados en controlar la caída de tensión en el condensador.



Como ingenieros de control, el primer paso para conseguirlo es modelar matemáticamente el comportamiento dinámico del sistema. Para ello se van a obtener sus ecuaciones diferenciales a partir de las leyes físicas que gobiernan el sistema. En este caso, aplicando la ley de tensiones de Kirchhoff obtenemos:

$$(1)\ V_e(t) = L \cdot \frac{\partial i}{\partial t} + R \cdot i(t) + \frac{1}{C}\int i\,\mathrm{dt}$$

$$(2)\ V_,(t) = \frac{1}{C} \cdot \int i\,\mathrm{dt}$$

Donde (1) expresa que la fuente de tensión del sistema $V_e(t)$ tiene que ser igual a la caída de tensión sufrida en cada uno de los componentes del mismo, y (2) expresa el valor de la caída de tensión en el condensador $V_s(t)$.

En este punto hay varias maneras de proceder en función de las variables de estado elegidas, pero en este caso perseguimos una ecuación diferencial expresada en función de la entrada y salida del sistema, por lo cual hay que buscar reemplazar la corriente $i(t)$por una expresión en función de estas. Para ello la despejamos de (2):

$$V_s(t) = \frac{1}{C} \cdot \int i\,\mathrm{dt} \quad \Rightarrow \quad C \cdot V_s(t) = \int i\,\mathrm{dt} \quad \Rightarrow C \cdot \frac{\partial V_s(t)}{\partial t} = i(t)$$

Sustituyendo $i(t)$ y su derivada en (1) obtenemos:

$$V_e(t) = L \cdot C \cdot \frac{\partial^2 V_s(t)}{\partial t^2} + R \cdot C \cdot \frac{\partial V_s(t)}{\partial t} + V_s(t)$$

Despejando la derivada de mayor orden (la derivada de mayor orden ha de quedar multiplicada por 1), y cambiando la notación de las derivadas por simplicidad, obtenemos:

$$\ddot{V}_s + \frac{R}{L}\dot{V}_s + \frac{1}{LC}V_s = \frac{1}{LC}V_e$$

Llegados a este punto ya podemos definir las **variables de estado**. En este caso vamos a optar por definir como primera variable de estado la salida del sistema, y como segunda su derivada. Es decir, vamos a estar interesados en la evolución y control de la caída de tensión en el condensador y de la tendencia de esta, obtenida mediante su derivada:

- $x_1 = V_s$
- $x_2 = \dot{V}_s$

Con el fin de definir la función de transición (esto es, encontrar las matrices dinámicas y de control), la cual define evolución del sistema, obtenemos las derivadas de dichas variables de estado:

- $\dot{x}_1 = \dot{V}_s = x_2$
- $\dot{x}_2 = \ddot{V}_s = \frac{1}{LC}V_e - \frac{R}{L}\dot{V}_s - \frac{1}{LC}V_s$

Y si además definimos como variable de entrada la tensión de entrada ($u = V_e$), y como variable de salida la tensión de salida ($y = V_s = x_1$), obtenemos:

- $\dot{x}_1 = x_2$
- $\dot{x}_2 = -\frac{1}{LC}x_1 - \frac{R}{L}x_2 + \frac{1}{LC}u$
- $y = x_1$

Y expresadas de manera matricial:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix}}_{A} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{\mathrm{LC}} \end{bmatrix}}_{B} \cdot u$$

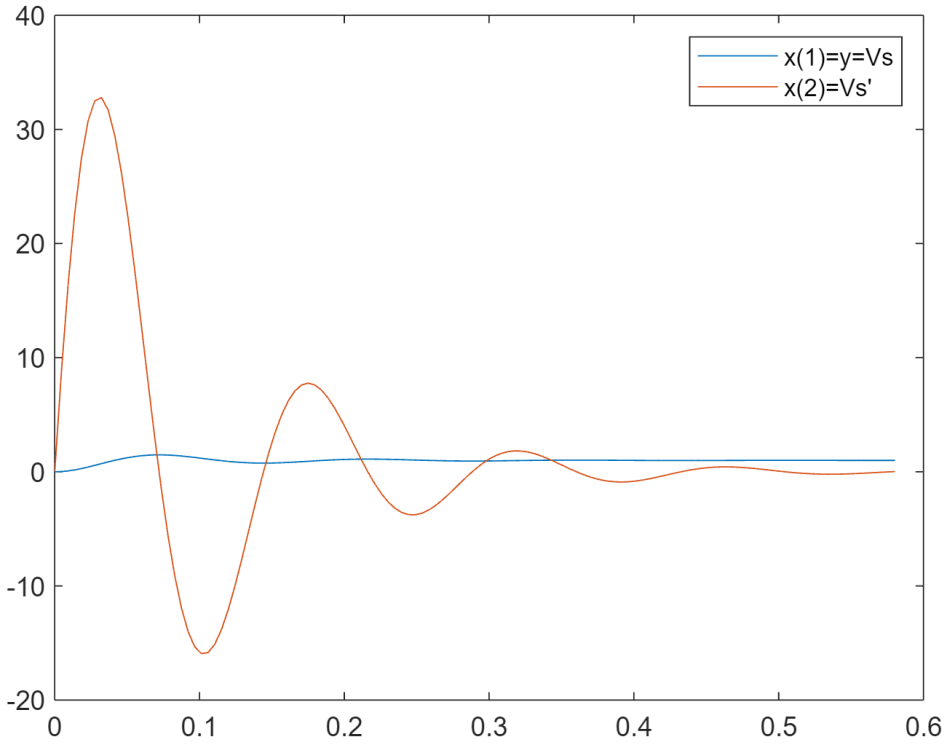
$$y = \underbrace{[1 \quad 0]}_{C} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{[0]}_{D} \cdot u$$

Definamos estas matrices en Matlab, fijando las constantes del circuito a $L = 0,5H$, $R = 10\Omega$, y $C = 0.001F$.

```
% Constantes del sistema
L = 0.5;
R = 10;
C= 0.001;

% Matrices del espacio de estados
A = [0 1; -1/(L*C) -R/L]; % matriz dinámica
B = [0; 1/(L*C)];         % matriz de control
C = [1 0];                % matriz de salida
D = [0];                  % matriz de influencia directa

% Creación del modelo del sistema mediante ss
sys_LRC = ss(A,B,C,D)
```

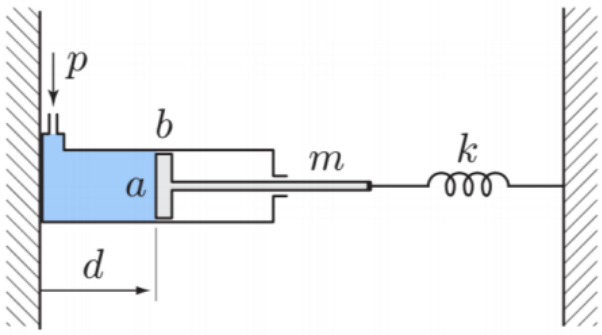
```
sys_LRC =

  A =
         x1     x2
   x1     0      1
   x2  -2000    -20

  B =
        u1
   x1    0
   x2  2000

  C =
       x1  x2
   y1   1   0

  D =
       u1
   y1   0

Continuous-time state-space model.
Model Properties
```

```
% El siguiente comando, que veremos con más detalle más adelante, simula
% la respuesta del sistema ante una entrada escalón unitario, y devuelve
% el valor de la salida y en los instantes de tiempo t, así como los
% valores de las variables de estado en x
[y,t,x] = step(sys_LRC);
% Mostramos la evolución de las variables de estado
plot(t,x), legend('x(1)=y=Vs',"x(2)=Vs'")
```



¡Apliquemos lo aprendido!

Tarea 1: En la figura se representa un cilindro hidráulico, cuyo émbolo de sección $a = 0,1m^2$ y masa $m = 1\text{kg}$ es accionado por una presión *p* de entrada.



En este sistema se mide la distancia d (salida) que se comprime un muelle de constante $k = 0,4\text{kg}/s^2$, el cual se halla en reposo con $d = 0$. Además, se halla presente una resistencia al movimiento de coeficiente $b = 0,4\text{kg}/s$, de forma que la ecuación diferencial es:

$$m\ddot{d} = -b\dot{d} - kd + ap$$

Se pide:

1. Obtener una representación interna del sistema.

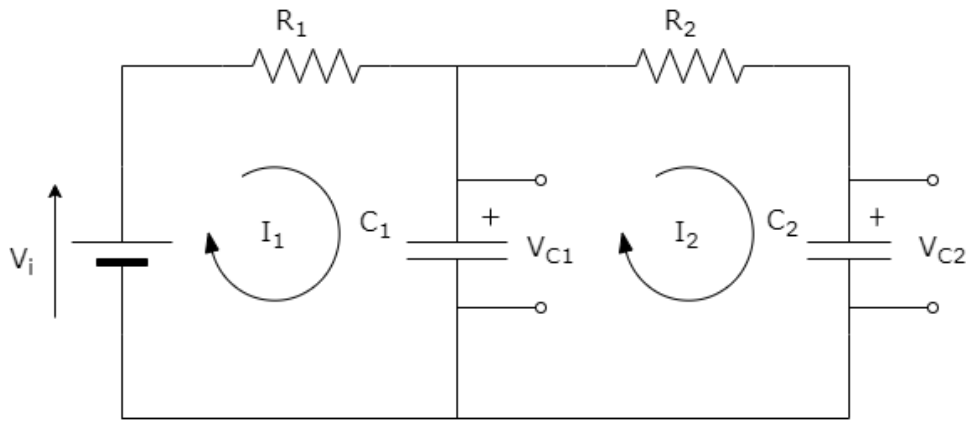
```
% Tu código aquí

a = 0.1;
m = 1;
k = 0.4;
d = 0;
b = 0.4;
A = [0 1;-k/m -b/m];
B = [0;a/m];
C = [1 0];
D = [0];
sys = ss(A,B,C,D)
```

```
sys =

  A =
         x1    x2
   x1     0     1
   x2  -0.4  -0.4

  B =
        u1
   x1    0
   x2  0.1

  C =
       x1  x2
   y1   1   0

  D =
       u1
   y1   0

Continuous-time state-space model.
Model Properties
```

Resultado esperado:

```
Gss =

  A =
         x1    x2
   x1     0     1
   x2  -0.4  -0.4

  B =
       u1
   x1   0
   x2 0.1

  C =
       x1  x2
   y1   1   0

  D =
       u1
   y1   0

Continuous-time state-space model.
```

Tarea 2: La siguiente figura muestra el esquema eléctrico de un circuito RCRC:



El comportamiento dinámico de dicho sistema, empleando la ley de tensiones de Kirchhoff, puede modelarse matemáticamente por medio de las siguientes ecuaciones diferenciales:

$$\text{malla 1)}\ V_i = V_{R1} + V_{C1}$$
$$V_i = I_1 R_1 + V_{C1}$$
$$\text{malla 2)}\ V_{C1} = V_{R2} + V_{C2}$$
$$V_{C1} = I_2 R_2 + V_{C2}$$

Teniendo en cuenta que $V_{C1} = \frac{1}{C_1}\int (I_1 - I_2)\mathrm{dt}$ y $V_{C2} = \frac{1}{C_2}\int I_2\,\mathrm{dt}$, obtener la representación en espacio de estados teniendo en cuentra que:

- la entrada al sistema es la tensión V_i,
- las variables de estado serán $x_1 = V_{C1}$ y $x_2 = V_{C2}$, es decir, la caída de tensión en los condensadores, y

- la salida del sistema es la caída de tensión en el primer condensador, $y = V_{C1}$.
- Los valores que toman las resistencias y las capacitancias se dan en la celda de código.

Pista: Trata de despejar las dos corrientes y de sustituirlas en las ecuaciones obtenidas en las mallas, así se podrán obtener dos ecuaciones donde aparezcan las derivadas de las dos tensiones, facilitando la construcción de la representación en espacio de estados.

```
R1 = 8; R2 = 800; C1 = 12*10^-6; C2 = 20*10^-6;
% Tu código aquí
%Hecho en foto
A=[-1/(C1*R2)-1/(R1*C1) 1/(C1*R2); 1/(R2*C2) -1/(R2*C2)];
B=[1/(R1*C1); 0];
C=[1 0];
D=[0];

sys2 = ss(A,B,C,D)
```

```
sys2 =

  A =
              x1          x2
   x1  -1.052e+04       104.2
   x2        62.5       -62.5

  B =
             u1
   x1  1.042e+04
   x2          0

  C =
       x1  x2
   y1   1   0

  D =
       u1
   y1   0

Continuous-time state-space model.
Model Properties
```

Resultado esperado:

```
circuito =

  A =
              x1          x2
   x1  -1.052e+04       104.2
   x2        62.5       -62.5

  B =
             u1
   x1  1.042e+04
   x2          0

  C =
       x1  x2
   y1   1   0

  D =
       u1
   y1   0

Continuous-time state-space model.
```

1.3 Puntos de equilibrio

Los puntos de **equilibrio**, también llamados puntos de **operación** o **consigna**, son puntos en el espacio de estados donde el sistema puede permanecer operando con actuaciones pequeñas y constantes alrededor de dicho punto. Los puntos que cumplen con esta caracteristica son aquellos en los que las derivadas de las variables de estado se anulan.

Estos puntos son especialmente útiles a la hora de linealizar sistemas, ya que nos permiten aproximar un sistema no lineal alrededor de dicho punto y tratarlo como lineal dentro de un cierto rango de operación. Tenemos una sesión práctica especialmente dedicada a la linealización de sistemas, pero vamos a ir adelantando trabajo con el procedimiento de obtención de **puntos de equilibrio**.

Vamos con un ejemplo. En el contexto de nuestro querido circuito LRC, recordemos cuales son las derivadas de las variables de estado:

- $\dot{x}_1 = x_2$
- $\dot{x}_2 = -\frac{1}{LC}x_1 - \frac{R}{L}x_2 + \frac{1}{LC}u$
- $y = x_1$

Si procedemos igualandolas a cero, obtenemos:

- $\dot{x}_1 = 0 \Rightarrow x_{2e} = 0$
- $\dot{x}_2 = 0 \Rightarrow -\frac{1}{LC}x_{1e} + \frac{1}{LC}u_e = 0 \Rightarrow x_{1e} = u_e$
- $y = u_e$

Estas operaciones también podemos realizarlas usando el fabuloso cálculo simbólico de MATLAB:

```
syms L C R x1 x2 ue
A = [0 1; -1/(L*C) -R/L]; % matriz dinámica
B = [0; 1/(L*C)];         % matriz de control
C = [1 0];                % matriz de salida
D = [0];                  % matriz de influencia directa
[x1e x2e] = solve(0==A*[x1;x2]+B*ue)
```

x1e =

$$ue$$

x2e =

$$0$$

```
ye = C*[x1e; x2e] + D*ue
```

ye =

$$ue$$

Tarea 3: Calcular los puntos de equilibrio de la descripción interna definida en la Tarea 1, tanto en papel como usando MATLAB.

Pista: Emplea `help syms` y `help solve` para obtener más información acerca de estos fantásticos comandos.

```
% Tu código aquí


syms x1 x2 ue
a = 0.1;
m = 1;
k = 0.4;
d = 0;
b = 0.4;
A=[0 1; -k/m -b/m];
B=[0; a];
C=[1 0];
D=[0];

[x1e x2e] = solve (0==A*[x1;x2]+B*ue)
```

x1e =

$$\frac{ue}{4}$$

x2e =

$$0$$

```
ye=C*[x1e; x2e] + D*ue
```

ye =

$$\frac{ue}{4}$$

```

```

Resultado esperado:

x1e =

$$\frac{ue}{4}$$

x2e = 0

ye =

$$\frac{ue}{4}$$

Descripción externa e interna

Equivalencia entre descripciones

Table of Contents

En los dos cuadernos anteriores hemos trabajado con la descripción externa (función de transferencia) e interna (espacio de estados) de sistemas. Tratándose de distintas representaciones de un mismo sistema, es posible establecer relaciones entre ellas.

1. Paso a descripción externa

Dado un sistema en descripción interna, es posible obtener su equivalente en descripción externa aplicando la transformada de Laplace a la ecuación de estado, con condiciones iniciales nulas. De este modo, se llega a la siguiente igualdad:

$$G(s) = C(\mathrm{s}\mathrm{I} - A)^{-1}B + D$$

Tarea 1: Obtener la función de transferencia $G(s)$del circuito LRC del cuaderno anterior expresado en espacio de estados, definido en la siguiente celda de código, empleando la equivalencia anterior.

```
% Constantes del sistema
L = 0.5; R = 10; C= 0.001;
% Matrices del espacio de estados
A = [0 1; -1/(L*C) -R/L]; % matriz dinámica
B = [0; 1/(L*C)];         % matriz de control
C = [1 0];                % matriz de salida
D = [0];                  % matriz de influencia directa
I = eye(2);

% Tu código aquí
s = tf('s');
G = C*((s*I-A)^-1)*B+D
```

```
G =

        2000
  -----------------
  s^2 + 20 s + 2000

Continuous-time transfer function.
Model Properties
```

Resultado esperado:

```
G =

        2000
  -----------------
  s^2 + 20 s + 2000

Continuous-time transfer function.
```

2. Paso a descripción externa

El paso de descripción interna a externa no es único, ya que existen infinitas posibilidades para realizarlo. Por ello, se suele recurrir a formas canónicas, que fijan un procedimiento para dicha conversión y presentan ventajas de cara a estudiar la estabilidad del sistema, controlabilidad, observabilidad, etc. Ejemplo de estas formas canónicas son la de Jordan, la de controlabilidad, o la de observabilidad.

3. Equivalencia entre descripciones en MATLAB

MATLAB nos permite convertir un modelo en descripción interna a su equivalente en descripción externa, esto es, obtener su función de transferencia. Esto se puede hacer simplemente llamando a la función **`tf`** e introduciendo el modelo del sistema en descripción interna como argumento. Su sintaxis sería:

```
G = tf(state_space)
```

Tarea 2: Obtener la función de transferencia del circuito LRC, empleando primero las matrices A, B, C y D para definir su representación en espacio de estados, y una vez en descripción externa, utilizar el comando **`tf`** para obtener su función de transferencia. Comprueba que obtienes el mismo resultado que usando la equivalencia de la sección 1 (Tarea 1).

```
% Tu código aquí
L = 0.5;
R = 10;
C= 0.001;
% Matrices del espacio de estados
A = [0 1; -1/(L*C) -R/L]; % matriz dinámica
B = [0; 1/(L*C)];         % matriz de control
C = [1 0];                % matriz de salida
D = [0];                  % matriz de influencia directa

%Usamos el comando ss para modelar su representación en espacio de estados
LRC = ss(A,B,C,D)
```

```
LRC =

  A =
         x1     x2
   x1     0      1
   x2  -2000    -20

  B =
```

```
       u1
 x1     0
 x2  2000

C =
     x1  x2
 y1   1   0

D =
     u1
 y1   0

Continuous-time state-space model.
Model Properties
```

```
G2 = tf(G)
```

```
G2 =

        2000
  -----------------
  s^2 + 20 s + 2000

Continuous-time transfer function.
Model Properties
```

```
G2 = tf(LRC)
```

```
G2 =

        2000
  -----------------
  s^2 + 20 s + 2000

Continuous-time transfer function.
Model Properties
```

¡OJO! La operación inversa, pasar de descripción externa a interna empleando el comando **ss**, puede no dar siempre el mismo resultado, ya que como sabemos la descripción externa de un sistema no es única: puede haber múltiples descripciones en función de las variables de estado escogidas, y MATLAB tiene libertad de elección. Aún así, el modelo obtenido sigue siendo válido. Ejemplo:

```
circuito_bis = ss(G)
```

```
circuito_bis =

  A =
         x1     x2
  x1    -20  -62.5
  x2     32      0

  B =
     u1
  x1   8
  x2   0

  C =
         x1     x2
  y1      0  7.812
```

```
  D =
       u1
   y1   0

Continuous-time state-space model.
Model Properties
```

Podemos comprobar como el resultado es una descripción interna distinta a la anterior, pero igualmente válida, ya que se diferencian en las variables de estado empleadas para caracterizar la evolución del sistema.

Tarea 3

$$\begin{bmatrix} \dot{d} \\ \ddot{d} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{M} & -\frac{b}{M} \end{bmatrix} \begin{bmatrix} d \\ \dot{d} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{a}{M} \end{bmatrix} p$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{pmatrix} d \\ \dot{d} \end{pmatrix} + 0 \; p$$

$x_1 = d$

$x_2 = \dot{d}$

$x_2 = \dot{x}_1 \rightsquigarrow \dot{d} = \dot{d}$

$\ddot{d} = -\frac{b}{M}\dot{d} - \frac{k}{M} d + \frac{a}{M} p$

Saco el punto de eq

$\dot{x}_1 = 0 \rightsquigarrow$ [redacted] $\boxed{\dot{d}_e = 0.}$

$$0 = -\frac{0,4}{1} d + \frac{0,1}{1} p$$

$$\frac{0,4}{1} d_e = \frac{0,1}{1} p \qquad d_e = \frac{0,1}{0,4} p \rightsquigarrow \boxed{d_e = \frac{p}{4}}$$

$$y = (1, 0) \begin{bmatrix} p/4 \\ 0 \end{bmatrix} \qquad \boxed{y_e = \frac{p}{4}}$$

Tarea 1

$$M\ddot{d} = -b\dot{d} - kd + ap$$

$$\ddot{d} = -\frac{b}{M}\dot{d} - \frac{k}{M} d + \frac{a}{M} p$$

$$\boxed{\begin{bmatrix} \dot{d} \\ \ddot{d} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{M} & -\frac{b}{M} \end{bmatrix} \begin{bmatrix} d \\ \dot{d} \end{bmatrix} + \begin{bmatrix} 0 \\ a/M \end{bmatrix} p}$$

Tarea #2

1) $V_i = V_{R1} + V_{C1}$

$V_i = I_1 \cdot R_1 + V_{C1}$.

2) $V_{C1} = V_{R2} + V_{C2}$.

$V_{C1} = I_2 R_2 + V_{C2}$.

$V_{C1} = \frac{1}{C_1}\int (I_1 - I_2)\,dt$

$V_{C2} = \frac{1}{C_2}\int I_2\,dt$. $\quad I_2 = \dot{V}_{C2} \cdot C_2$

$\dot{V}_{C1} \cdot C_1 = I_1 - I_2$.

$\dot{V}_{C1} \cdot C_1 = I_1 - \dot{V}_{C2} \cdot C_2$.

$I_1 = \dot{V}_{C1} \cdot C_1 + \dot{V}_{C2} \cdot C_2$.

Sustituyo.

$V_i = R_1 \cdot \dot{V}_{C1} \cdot C_1 + R_1 \cdot \dot{V}_{C2} C_2 + V_{C1}$.

$V_{C1} = R_2 \cdot \dot{V}_{C2} C_2 + V_{C2}$.

Despejo la derivada más grande.

$$\dot{V}_{C1} = \frac{1}{R_1 C_1} V_i - \frac{R_1 \cdot C_2}{R_1 C_1}\dot{V}_{C2} - \frac{1}{R_1 C_1} V_{C1}.$$

$$\dot{V}_{C2} = \frac{1}{R_2 \cdot C_2} \cdot V_{C1} - \frac{1}{R_2 C_2} \cdot V_{C2}.$$

Melo en $\dot{V}_{C1}$ $\dot{V}_{C2}$

$$\dot{V}_{C1} = \frac{1}{R_1 C_1} V_i - \frac{C_2}{C_1 R_2 C_2} V_{C1} + \frac{C_2}{C_1 R_2 C_2} V_{C2} - \frac{1}{R_1 C_1} V_{C1}$$

$$A = \begin{bmatrix} -\frac{1}{C_1 R_2} - \frac{1}{R_1 C_1} & \frac{1}{C_1 R_2} \\ \frac{1}{R_2 C_2} & -\frac{1}{R_2 C_2} \end{bmatrix} \begin{bmatrix} V_{C1} \\ V_{C2} \end{bmatrix} \quad B = \begin{bmatrix} \frac{1}{R_1 C_1} \\ 0 \end{bmatrix} V_i \quad C = [1 \; 0] \quad D = [0].$$