Universidad
Rey Juan Carlos

Área de Tecnología Electrónica

Práctica 2

MATLAB *Control System Toolbox*

Funciones de transferencia, transformadas de Laplace y álgebra de bloques con MATLAB

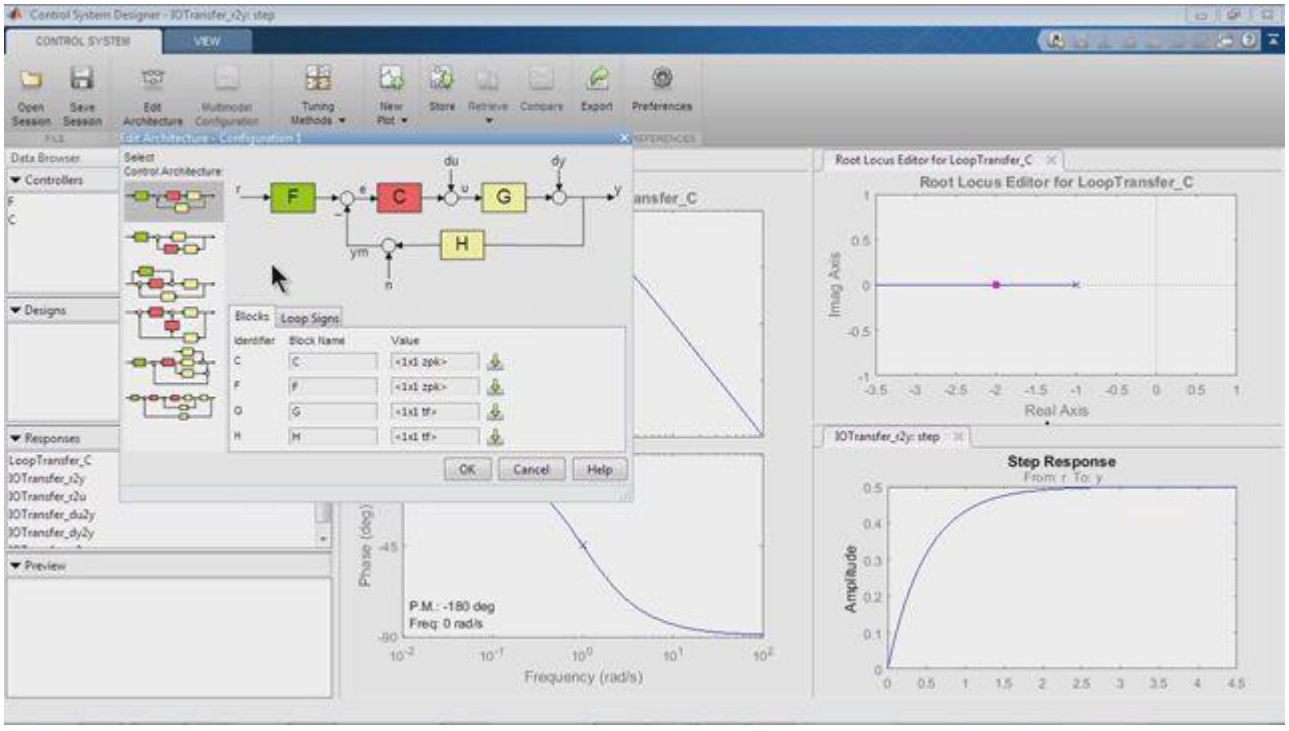
2022-2023

ÍNDICE

1. Introducción al *Control System Toolbox* de MATLAB

MATLAB proporciona un entorno muy completo para trabajar con sistemas de control. Dicho entorno se basa en las funciones, aplicaciones y algoritmos incluidos en el llamado *Control System Toolbox* del software.

Mediante el *Control System Toolbox* podremos crear modelos de sistemas de control LTI (lineales e invariantes en el tiempo) mediante representaciones de funciones de transferencia o modelos en el espacio de estados, ya sean sistemas continuos o sistemas discretos, y de una o varias entradas y salidas (SISO, MIMO). Además, dicho Toolbox contiene funciones que nos permitirán estudiar la respuesta transitoria y estacionaria de los sistemas a estímulos de entrada típicos como la función impulso, escalón, rampa, etc. También podremos analizar y diseñar sistemas de control mediante técnicas como el *lugar de las raíces* o la *respuesta en frecuencia*, así como diseñar y sintonizar controladores, estudiar el efecto de las perturbaciones sobre los sistemas, etc.



Para obtener una descripción más precisa de la potencia y versatilidad del *Control System Toolbox*, consulta el siguiente enlace:

https://es.mathworks.com/products/control.html

O visualiza el siguiente vídeo introductorio (en inglés):

https://www.youtube.com/watch?v=NfSPmBwD71g

2. Modelado de sistemas LTI mediante Funciones de Transferencia

En este apartado vamos a aprender a crear funciones de transferencia en MATLAB mediante diferentes comandos pertenecientes al *Control System Toolbox*. También veremos cómo pasar de una representación del sistema a otra.

Nota: resuelve los ejercicios prácticos que encontrarás a continuación utilizando scripts (.m) o LiveScripts (.mlx) de MATLAB.

2.1 Creación de funciones de transferencia en forma polinómica (TF)

Una función de transferencia G(s) en el dominio de la frecuencia compleja (dominio s) típicamente se expresa de forma polinómica o polinomial, es decir, como un cociente de polinomios de s, G(s) = num(s)/den(s):

$$G(s) = \frac{2s + 3}{4s^2 + 5s + 6}$$

Por ello, el comando más utilizado para definirlas es **G = tf(num,den)**, donde "num" y "den" son los polinomios del numerador y del denominador de la función de transferencia, respectivamente:

```
>> num = [2 3];
>> den = [4 5 6];
>> G = tf (num, den)
```

También se pueden pasar los polinomios directamente como argumentos del comando TF:

```
>> G = tf ([2 3], [4 5 6])
```

Ejercicio práctico 1. Definición de funciones de transferencia mediante TF

1. Define las tres siguientes funciones de transferencia en MATLAB:

$$G_1(s) = \frac{1}{s^2 + 1} \qquad G_2(s) = \frac{s + 1}{s^2 - 5s + 6} \qquad G_3(s) = \frac{s^2 + 2s + 3}{s^5 + 4s^3 + 5s + 6}$$

2. Una vez definidas, observa qué tipo de objeto usa MATLAB para almacenarlas (columna *Class* del Workspace)

Si el numerador o el denominador de la función de transferencia no están expresados en forma de un único polinomio, puedes ayudarte del comando de convolución de vectores en MATLAB, que permite realizar una multiplicación de polinomios:

```
>> conv (poli_1, poli_2)
```

Ejercicio práctico 2. Definición de funciones de transferencia no polinómicas

1. Define las dos siguientes funciones de transferencia ayudándote de conv():

$$G_4(s) = \frac{10(s+5)}{(s+1)(s-3)(s^2+3s+6)} \qquad G_5(s) = \frac{s^2+3}{(s^2+s+1)(s^3+2s+3)}$$

2.2 Creación de funciones de transferencia en forma factorizada (ZPK)

En otras ocasiones las funciones de transferencia no se presentan en forma polinominal sino que lo hacen en forma factorizada, es decir, como producto de las raíces $\mathbf{z}_i$ del numerador (que denominaremos ceros del sistema) entre el producto de las raíces $\mathbf{p}_i$ del denominador (que en este caso son los **polos del sistema**):

$$G(s) = \frac{K(s-z_0)(s-z_1)\dots(s-z_m)}{(s-p_0)(s-p_1)\dots(s-p_n)}$$

La función de transferencia, expresada en función de los polos y los ceros del sistema se puede definir mediante el comando **zpk** *(zero-pole-gain)*. Para ello se pasan como argumentos tres vectores, el que contiene los ceros (z), el de los polos (p) y el factor de ganancia estática (K). En un sistema que no contenga ceros el vector correspondiente al vector z debe estar vacío, indicándose con [].

>> G = zpk ([z_m … z_1 z_0], [p_n … p_1 p_0], K)

Ejercicio práctico 3. Definición de funciones de transferencia factorizadas

1. Define la función de transferencia $G_6(s)$ para un sistema que tiene una ganancia K igual a 2, un cero situado en -1 y tres polos situados en -2, -3 y -4
2. Repite el apartado anterior definiendo $G_7(s)$ para un sistema que tiene una ganancia K igual a 5, un cero situado en +1 y tres polos situados en -2, +3 y -4
3. Define la función de transferencia $G_8(s)$ que no tiene ceros y tiene un polo triple en -1 (con K=1)
4. Define la función de transferencia $G_9(s)$ que no tiene ceros y tiene tres polos, uno en -1 y los otros dos polos complejos conjugados -1 + 2i, -1 -2i (con K=1)

2.3 Conversión entre funciones de transferencia polinómicas y factorizadas

Los comandos tf y zpk también pueden tomar argumentos que ya sean una función de transferencia, de tal manera que:

- Si GP es una función de transferencia en forma polinómica, el comando **GZPK = zpk(GP)** convierte la función GP a forma factorizada.

- Si GZPK es una función de transferencia en forma factorizada, el comando **GP = tf(GZK)** convierte la función GP a forma polinomial.

Ejercicio práctico 4. Conversión de funciones de transferencia

1. Convierte a forma factorizada las funciones de transferencia G1, G2, G3, G4 y G5 definidas en los ejercicios prácticos 1 y 2
2. Convierte a forma polinómica las funciones de transferencia G6, G7, G8 y G9 definidas en los ejercicios prácticos 1 y 2

2.4 Forma simplificada: uso de s = tf('s')

Una vez definidas varias funciones de transferencia, MATLAB permite realizar operaciones algebraicas simples entre varias funciones (suma, resta, multiplicación y división de funciones de transferencia) y/o producto de funciones de transferencia por escalares.

Por ello, una última forma muy simplificada de definir funciones de transferencia se basa en definir la variable "s" como función de transferencia (mediante **s = tf ('s')**) y a continuación especificar la función buscada mediante una **expresión racional** que incluya la variable s (es decir, con sumas, restas, multiplicaciones y divisiones de "s"), por ejemplo:

```
>> s = tf ('s')

>> G = (s + 1)/(s^3 + 3*s + 1)
```

El anterior método sería equivalente al uso de la siguiente expresión

```
>> G = tf ( [1 1] , [1 0 3 1] )
```

Ejercicio práctico 5. Uso de s = tf('s')

1. Utilizado este método simplificado para definir una función de transferencia del ejercicio práctico 1, las dos funciones del ejercicio práctico 2 y una función del ejercicio práctico 3.

3. Transformadas inversas de Laplace en MATLAB

3.1 Descomposición (expansión) en fracciones simples

Con MATLAB podemos realizar descomposiciones (también denominadas expansiones) en fracciones simples del cociente de dos polinomios, utilizando el comando **residue**.

Para ello, si queremos descomponer una transformada de Laplace Y(s) con **residue**, ésta tiene que estar en forma polinómica Y(s) = b(s)/a(s). La sintaxis habitual es la siguiente:

>> [r, p, k] = residue (b, a)

- **b** y **a** son los polinomios del numerador y del denominador, respectivamente.
- **r** es un vector que contiene los residuos buscados.
- **p** es otro vector que contiene las raíces (polos) del denominador.
- En caso de que el grado del numerador b(s) sea igual o mayor que el del denominador a(s), el vector **k** contendrá el polinomio resultado del **cociente entre ambos**, y la descomposición se realizará sobre el resto de dicho cociente. Si dicho grado del numerador es inferior al del denominador, **k** será un vector vacío.

$$\frac{b(s)}{a(s)} = \frac{b_m s^m + b_{m-1}s^{m-1} + \ldots + b_1 s + b_0}{a_n s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0} = \frac{r_n}{s - p_n} + \ldots + \frac{r_2}{s - p_2} + \frac{r_1}{s - p_1} + k(s)$$

Ejercicio práctico 6. Descomposición en fracciones simples

1. Calcula los residuos y los polos de las descomposición en fracciones simples de las siguientes funciones Y(s):

$$Y_1(s) = \frac{32}{s^3 + 12s^2 + 32s} \qquad Y_2(s) = \frac{6s^2 + 22s + 18}{s^3 + 6s^2 + 11s + 6}$$

$$Y_3(s) = \frac{s + 7}{s^2 + 10s + 25} \qquad Y_4(s) = \frac{2s^3 + 25s^2 + 95s + 110}{s^3 + 9s^2 + 26 + 24}$$

2. Una vez descompuestas, calcula sus transformadas inversas de Laplace (tabla).

Residue también opera de forma inversa, es decir, si se le proporcionan como argumentos los valores de las raíces, los polos y el vector cociente k, proporciona como salida el numerador y denominador de la función original, con la siguiente sintaxis:

>> [num, den] = residue (raices, polos, k)

Para más información y ejemplos, consulta la ayuda de MATLAB (**doc residue**).

3.2 Transformadas simbólicas usando el *Symbolic Math Toolbox*

En MATLAB también existe la posibilidad de trabajar con variables simbólicas, utilizando el *Symbolic Math Toolbox*. Este potente Toolbox contiene comandos que permiten manipular expresiones, resolver ecuaciones simbólicas, realizar derivadas, integrales, etc. Entre sus capacidades también se encuentra el manejo de transformadas de Laplace.

Para usar el *Symbolic Math Toolbox* es imprescindible definir previamente aquellas variables que serán simbólicas utilizando el comando **syms**. Por ejemplo, para definir la variable s como simbólica, usaremos:

```
>> syms s
```

Nota: no es compatible definir s como variable simbólica (**syms s**) y como función de transferencia (**s = tf('s')**) simultáneamente.

Para calcular transformadas inversas de Laplace usaremos el comando **ilaplace.** Por ejemplo, para calcular la salida en el dominio del tiempo de un sistema con una función de transferencia G(s) ante una entrada escalón R(s) = 1/s, haremos:

```
>> syms s
>> G = 1/(s^2 + 3*s + 2)
>> R = 1/s
>> Y = G*R
>> y = ilaplace (Y)
```

Para trabajar con expresiones simbólicas resulta muy recomendable utilizar **LiveScritps de MATLAB**, principalmente porque la salida de cada comando se muestra con un renderizado tipográfico avanzado. Si queremos obtener un formateado similar por línea de comandos, podremos usar la función **pretty()**, aunque el resultado es bastante peor que el que se obtiene con un LiveScript.

Ejercicio práctico 7. Transformadas inversas de Laplace con MATLAB

1. Utilizando un LiveScript, calcula la respuesta en el dominio del tiempo de las siguientes funciones de transferencia ante una entrada escalón unitario:

$$G_1(s) = \frac{1}{(s+1)} \qquad G_2(s) = \frac{5}{(s+1)(s+5)}$$

$$G_3(s) = \frac{1}{(s+1)^3} \qquad G_4(s) = \frac{1}{s^2+s+1}$$

2. Calcula también la respuesta en el dominio del tiempo de las funciones de transferencia anteriores ante una entrada rampa unitaria.

Se pueden realizar gráficas de una función simbólica en un intervalo determinado utilizando el comando **fplot**. Para incluir varias funciones en la misma gráfica, el primer argumento de fplot debe ser un vector que contenga dichas funciones. Por ejemplo:

```
>> syms t
>> x = cos (t);
>> y = sin (t);
```

>> fplot ([x, y] [0, 4*pi])

>> legend ("cos(t)",'sin(t)") *% Permite modificar y mostrar la leyenda en el gráfico.*

Nota: Si las funciones argumento de **fplot** provienen de una transformada inversa de Laplace obtenida con el comando **ilaplace**, no es necesario especificar que t es una variable simbólica (es decir, no hace falta introducir **syms t**).

Ejercicio práctico 8. Gráficas de funciones simbólicas

1. Añade el comando fplot al LiveScript del ejercicio práctico 7 para realizar la gráfica, por separado, de cada una de las transformadas inversas de Laplace que has ido calculando, en el intervalo [0, 10].
2. Realiza una única gráfica en la que aparezcan las respuestas ante una entrada escalón de las tres funciones de transferencia siguientes. ¿Observas alguna relación entre ellas?

$$G_1(s) = \frac{1}{(s+1)} \qquad G_2(s) = \frac{5}{(s+5)} \qquad G_3(s) = \frac{5}{(s+1)(s+5)}$$

El *Symbolic Math Toolbox* también nos permite calcular transformadas de Laplace, mediante el comando **laplace()**. Esto resulta de mucha utilidad si no tenemos a nuestro alcance la tabla de transformadas. Por ejemplo:

>> syms t

>> y = t

>> Y = laplace (y)

Ejercicio práctico 9. Transformadas de Laplace con MATLAB

1. Calcula las transformadas de Laplace de las siguientes funciones del tiempo:

$$x_1(t) = t^2 \qquad x_2(t) = \cos(t) \qquad x_3(t) = \sin(2t)$$

$$x_4(t) = e^{-2t}\cos(t) \qquad x_5(t) = t^2 e^{-2t}\sin(t)$$

Por último, indicar que existen funciones en el *Symbolic Math Toolbox* que nos permiten pasar de expresiones simbólicas a polinomios (**sym2poly**) y viceversa (**poly2sym**), lo que nos permitiría, en caso de que fuera necesario, conectar los dos Toolboxes que hemos utilizado en esta práctica. Para más información, consulta la ayuda de MATLAB.

4. Álgebra de bloques mediante MATLAB

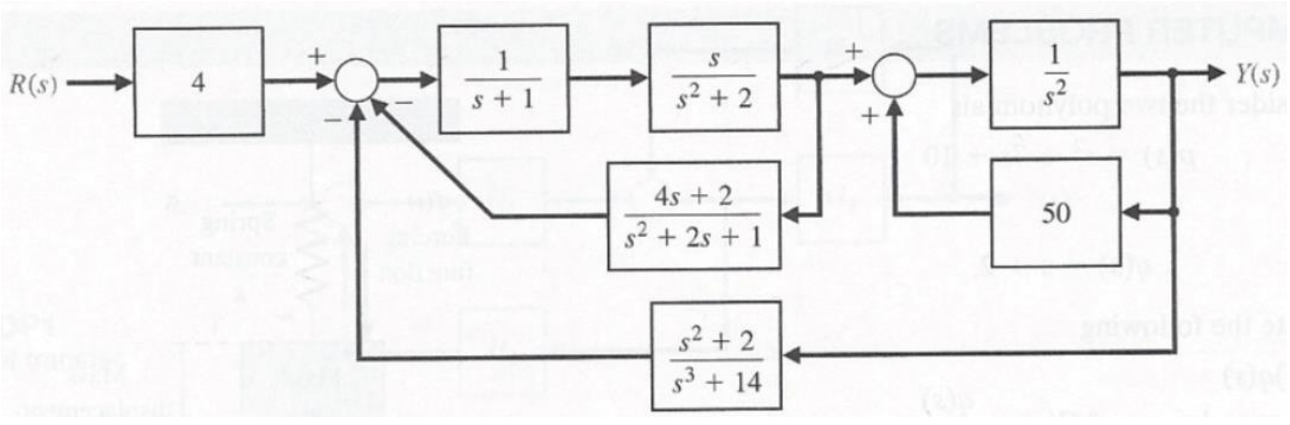
4.1 Herramientas para la interconexión y reducción de diagramas de bloques

En el *Control System Toolbox* encontramos varios comandos que nos permiten tanto conectar como reducir diagramas de bloques. De todos ellos destacan los tres siguientes:

- **Asociación en cascada o serie:** el comando **series (G1, G2)** obtiene la función de transferencia equivalente del conjunto de dos bloques G1 y G2 conectados en serie.
- **Asociación en paralelo:** el comando **parallel (G1, G2)** obtiene la función de transferencia equivalente del conjunto de dos bloques G1 y G2 conectados en paralelo.
- **Sistemas realimentados:** el comando **feedback (G, H)** obtiene la función de transferencia equivalente de un sistema realimentado. A este comando se le pasan, por este orden, la función de transferencia G del lazo directo y H, la de la realimentación, separadas por comas. Por defecto se considera una realimentación negativa. En el caso de la realimentación positiva, se debe añadir un 1 a continuación de las funciones de transferencia, es decir, la sintaxis debe ser **feedback (G, H, 1)**

Ejercicio práctico 10. Reducción de diagramas de bloques

1. Utiliza los comandos de reducción de diagramas de bloques para calcular la función de transferencia en lazo cerrado del siguiente sistema de control:



Otros comandos menos utilizados en el contexto de esta asignatura para la interconexión de bloques son **connect** y **append**. Utiliza la ayuda de MATLAB para descubrir su utilidad y sintaxis.