3º INGENIERÍA TÉCNICA INDUSTRIAL, ESPECIALIDAD MECÁNICA

AUTOMATIZACIÓN INDUSTRIAL

PRÁCTICA 6

INTRODUCCIÓN AL PROGRAMA MATLAB

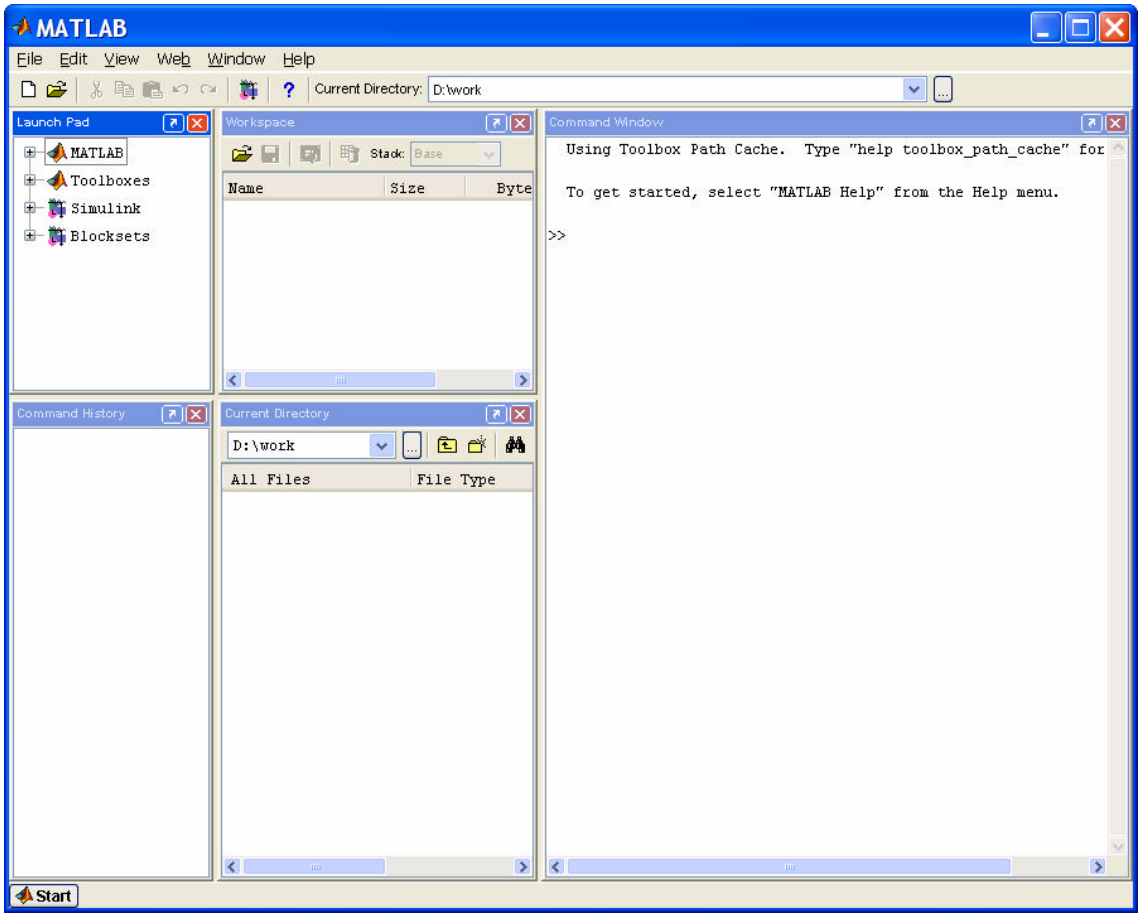
1. CARACTERÍSTICAS BÁSICAS DE MATLAB

- Funcionalidades básicas:
  - cálculo matricial
  - representaciones gráficas

- Librerías específicas: ***toolboxes***. A lo largo del curso se utilizarán las siguientes:
  - **Simulink**: simulación de sistemas dinámicos.
  - **Control**: análisis de sistemas y ajuste de bucles de control.
  - **Matemática simbólica**: opera con variables simbólicas.

2. EL ASPECTO DE MATLAB 6.5

Matlab 6.5 ofrece un aspecto configurable en función de la información que se desee mostrar en la pantalla. En la figura siguiente se muestra un posible aspecto:



Existen cinco elementos fundamentales, tal y como puede apreciarse en la imagen:

- Ventana de comandos ***(Command window)***: es un interfaz en modo texto sobre el que se introducen por teclado instrucciones Matlab. Será el elemento fundamental a utilizar en esta práctica.
- Ventana de variables ***(Workspace)***: en ella aparece un listado de todas las variables que se han empleado durante la sesión, de modo que pueden visualizarse su tamaño y su tipo.
- Historial de comandos ***(Command history)***: ofrece un listado de todas las instrucciones tecleadas durante la sesión.

- Navegador o ***(Current directory)***: indica los contenidos del directorio actual y permite navegar por la estructura de directorios del PC.
- Lanzador de aplicaciones ***(Launch pad)***: permite ejecutar de un modo rápido aplicaciones presentes en el entorno Matlab.

Es posible seleccionar cuáles de las ventanas se desean visualizar desde el menú ***View***. Para la presente práctica se desactivará la visualización de todas las ventanas excepto la ventana de comandos.

3. CÓMO ENCONTRAR AYUDA EN MATLAB

Existen distintas formas de localizar ayuda en el entorno de Matlab:

- **Ayuda en línea**

Se accede a través de la ventana de comandos tecleando *help nombrefunción*. La ayuda se obtiene en modo texto. Como ejemplo, se visualizará la ayuda de la función que permite invertir matrices tecleando:

```
>> help inv
```

- **Navegador de ayuda**

Se accede desde el menú ***Help***, seleccionando la opción ***Matlab help***. Constituye una manera más sencilla de localizar la misma información: las funciones están agrupadas en bloques y se proporciona un interfaz para navegar. Además ofrece información adicional como ejemplos e instrucciones de uso.

- **Ejemplos**

Matlab proporciona ejemplos y demostraciones de sus principales funcionalidades. Siempre es accesible el código fuente, con lo que puede ser directamente reutilizado. Se accede a ellos a través del menú ***Help***, seleccionando la opción ***Demos***.

- **Comando *lookfor* (búsqueda de palabras clave)**

Aunque más complicado de utilizar, proporciona en ocasiones información extra. El comando *lookfor* permite buscar entre las descripciones de todas las funciones de Matlab, aquellas que contienen la palabra clave que indiquemos. Como ejemplo, buscaremos todas las funciones de Matlab relacionadas con la transformada de Fourier tecleando:

```
>> lookfor fourier
```

4. VARIABLES Y MATRICES EN MATLAB

Matlab soporta nombres de variable de hasta 19 caracteres, y distingue entre mayúsculas y minúsculas.

El tipo de las variables puede ser:

- Entero
- Real
- Complejo
- Carácter
-

... y es asignado automáticamente.
Una sentencia de creación de variable es, por ejemplo:

```
>> pepe = 7

pepe =
      7
```

Esta sentencia crea la variable entera **pepe** y le asigna el valor **7**. Matlab muestra en pantalla el resultado de cada operación. Para evitarlo basta poner un punto y coma después de cada sentencia:

```
>> pepe = 7;
```

Todas las variables en Matlab son consideradas matrices. Las posibilidades que utilizaremos son:

- Matriz **n x m**: matriz bidimensional
- Matriz **n x 1** ó **1 x n**: vector (se maneja exactamente igual que una matriz)
- Matriz **1 x 1**: escalar (también se maneja exactamente igual que una matriz).

La forma de definir una matriz en Matlab es elemento a elemento:

```
>> A = [1 2 3; 4 5 6; 7 8 9]

A =
     1     2     3
     4     5     6
     7     8     9
```

Como puede apreciarse en el ejemplo, los distintos elementos de una fila se separan mediante espacios (o comas) y las distintas filas se separan mediante puntos y coma.

Algunas posibilidades de manejo de variables que ofrece Matlab:

- comprobar el contenido de alguna variable: basta con teclear su nombre en la ventana de comandos

```
>> pepe

pepe =
     7
```

- listar todas las variables existentes en un determinado momento: comando who.

```
>> who

Your variables are:

A         pepe
```

*Nota: la versión 6.5 de Matlab permite visualizar las variables existentes en la ventana **Workspace**.*

- eliminar alguna variable de memoria: comando clear.

```
» clear pepe
» who

Your variables are:

A
```

Podemos observar cómo la variable **pepe** ha desaparecido de la memoria.

5. MANEJO DE MATRICES

Matlab ofrece bastantes facilidades para el manejo de matrices. Volviendo al ejemplo anterior:

```
>> A = [1 2 3; 4 5 6; 7 8 9]

A =
     1     2     3
     4     5     6
     7     8     9
```

- Podemos acceder a cualquier elemento de la matriz especificando fila y columna:

```
» A (1,3)

ans =
     3
```

*Nota: **ans** es la variable por defecto donde Matlab guarda cualquier resultado; si hubiéramos deseado utilizar otra variable deberíamos haberlo especificado:*

```
» k = A(1,3)

k =
     3
```

- También se puede acceder a toda una fila o toda una columna, utilizando el operador dos puntos.

Este primer comando muestra todos los elementos de la fila 2:

```
»  A(2,:)

ans =
     4     5     6
```

Este segundo comando muestra todos los elementos de la columna 3:

```
» A (:,3)

ans =
     3
     6
     9
```

- O bien a grupos de filas y/o columnas:

Este comando muestra los elementos de las filas 1 hasta la 2 y de las columnas 2 hasta la 3:

```
» A(1:2,2:3)

ans =

     2     3
     5     6
```

- También es posible modificar cualquier elemento de una matriz:

```
» A(1,1) = 9

A =

     9     2     3
     4     5     6
     7     8     9
```

- E incluso añadir elementos a una matriz dada:

```
» A(4,4) = 1
```

```
A =

     9     2     3     0
     4     5     6     0
     7     8     9     0
     0     0     0     1
```

Podemos ver cómo los elementos no especificados se rellenan con ceros.

6. PRINCIPALES OPERADORES ARITMÉTICOS

Matlab ofrece una serie de operadores aritméticos válidos tanto para cálculo matricial como para cálculo escalar:

- Suma: +
- Resta: -
- Producto: *
- División: /
- Traspuesta: ‘
- Potencia: ^

En algunas ocasiones podrán presentarse ambigüedades. Por ejemplo, al multiplicar dos matrices caben dos posibilidades: producto matricial o producto elemento a elemento. Veamos cómo se resuelven:

```
» A = [1 2;3 4]

A =

     1     2
     3     4

» B = [2 4; 6 8]

B =

     2     4
     6     8

» C = A*B         % producto matricial

C =

    14    20
    30    44
»  D = A.*B       % el punto indica operación elemento a elemento

D =

     2     8
    18    32
```

La información sobre operadores puede ser encontrada en la ayuda de Matlab 6.5 dentro de:
Matlab Help -> Getting started -> manipulating matrices -> expressions -> operators

Además de los operadores comentados, existen una serie de funciones muy útiles en cálculo matricial:

- obtención de la matriz inversa: función **inv**:

```
» A = [1 2;3 4]

A =

     1     2
     3     4


» B = inv(A)

B =

   -2.0000    1.0000
    1.5000   -0.5000
```

- creación de un vector de términos crecientes o decrecientes:

```
» a = [0:1:5]     % inicio 0, fin 5, salto 1

a =

     0     1     2     3     4     5

» a = [5:-1:0]    % inicio 5, fin 0, salto -1

a =

     5     4     3     2     1     0

» a = [0:.2:1]    % inicio 0, fin 1, salto .2

a =

         0    0.2000    0.4000    0.6000    0.8000    1.0000
```

  Podemos crear cualquier vector creciente o decreciente que deseemos. Esta operación será bastante útil para formar bases de tiempo sobre las que evaluar el valor de funciones.

La información sobre operaciones matriciales puede ser encontrada en la ayuda de Matlab 6.5 en:
Matlab Help -> Getting started -> manipulating matrices -> working with matrices
Matlab Help -> Getting started -> manipulating matrices -> more about matrices and arrays

7. REPRESENTACIONES GRÁFICAS EN MATLAB

Matlab ofrece facilidades para la creación de gráficos 2D y 3D. Estudiaremos en primer lugar la función **plot**, el medio más sencillo para realizar representaciones bidimensionales.

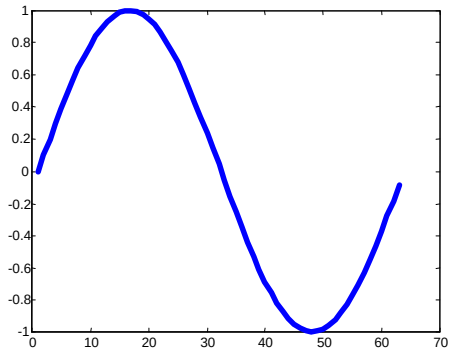
Existen diferentes sintaxis para la función **plot**. Intenteramos mostrar su funcionamiento con un ejemplo:

Supongamos que partimos de los siguientes datos iniciales:

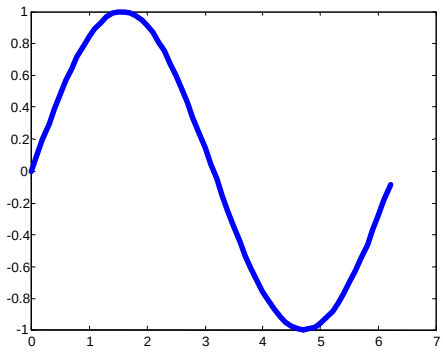
```
» a = [0:0.1:2*pi];     % a: contiene 63 ángulos entre 0 y 2p
» b = sin(a);           % b: contiene los valores del seno de a
» c = cos(a);           % c: contiene los valores del coseno de a
```

Comparemos dos formas de representar la función seno:

```
>> plot (b);
```




```
>> plot (a,b);
```



El resultado es aparentemente el mismo, pero existe una gran diferencia que es posible observar comparando los ejes x de ambas gráficas:

plot (b) representa los valores del vector b en el eje y frente a los índices (números de orden) de ese vector en el eje x; por eso el eje x toma valores que van desde 1 hasta 63.
plot (a,b) representa los valores del vector b en el eje y frente a los valores correspondientes del vector a en el eje x; por eso el eje x toma valores entre 0 y 2π.

Normalmente nos interesará más la segunda opción y la magnitud **a** representará la escala de tiempos.

Veamos ahora de qué forma podríamos representar a la vez el seno y el coseno, bien sobre un gráfico o sobre 2 gráficos distintos.

Si llamamos repetidamente a la función plot, el segundo gráfico borrará el primero, con lo cual no lograremos nuestro objetivo:

```
>> plot (a,b);
>> plot (a,c);
```

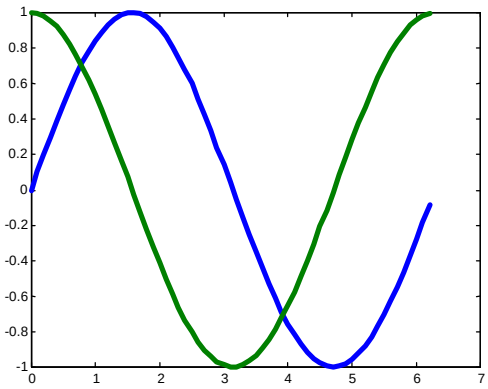
Si deseamos que el segundo gráfico se muestre sobre una ventana distinta, debemos intercalar la instrucción **'figure'**. Esta instrucción crea una nueva ventana de dibujo sobre la que se mostrarán todos los gráficos que se pidan a continuación:

```
>> plot (a,b);
>> figure;
>> plot (a,c);
```

En el caso de que deseemos representar ambas funciones sobre un mismo gráfico, será necesario intercalar la instrucción **'hold on'**. Esta instrucción permite dibujar nuevos datos sobre los datos anteriores, sin borrarlos:

```
>> plot (a,b);
>> hold on;
>> plot (a,c);
```

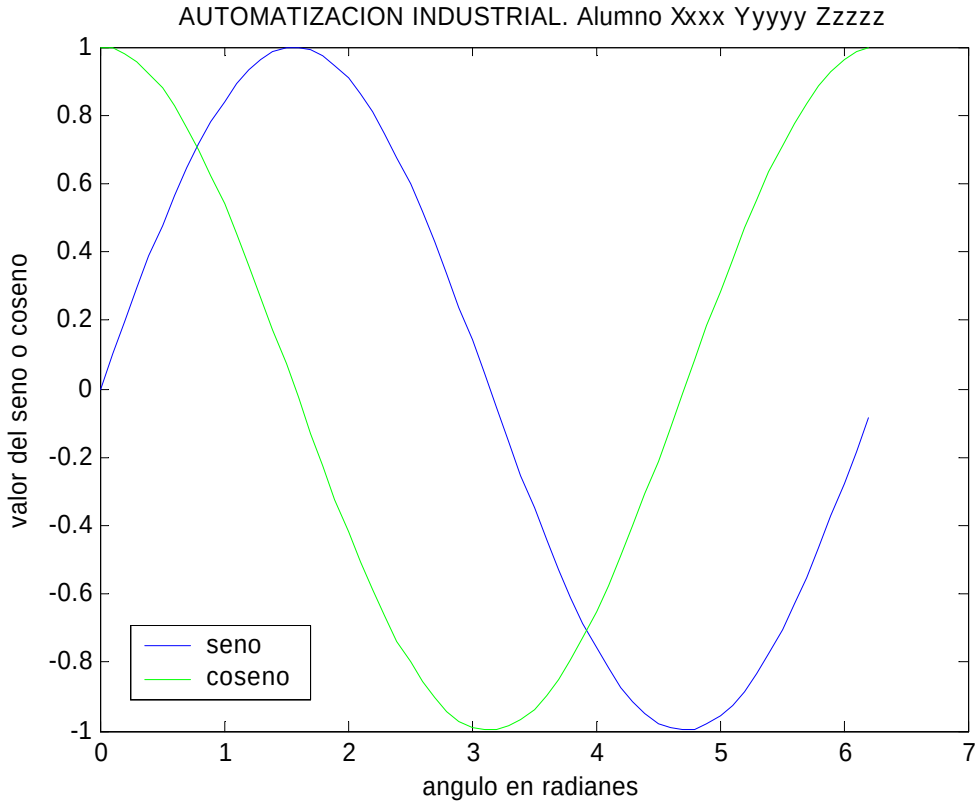
El resultado de las instrucciones anteriores debería ser similar al siguiente:



Un último aspecto que será importante a la hora de obtener representaciones gráficas será la forma de incluir textos sobre estas representaciones. Las principales instrucciones a utilizar son:

- **title:** escribe un título para el gráfico (en la parte superior).
- **xlabel:** da un nombre al eje x del gráfico.
- **ylabel:** da un nombre al eje y del gráfico.
- **legend:** leyenda: indica lo que representa cada trazo del gráfico.

La forma de utilizar estas instrucciones se puede encontrar en la ayuda de Matlab. Utilizaremos fundamentalmente las tres primeras instrucciones. Con ellas, y si se desea, modificando los colores, se puede obtener un resultado como el siguiente sobre el último gráfico realizado:



De ahora en adelante, todos los gráficos que se incluyan en los informes de prácticas deberán incluir un título en el que se indiquen asignatura, curso y nombre de alumno similar al mostrado en este ejemplo.

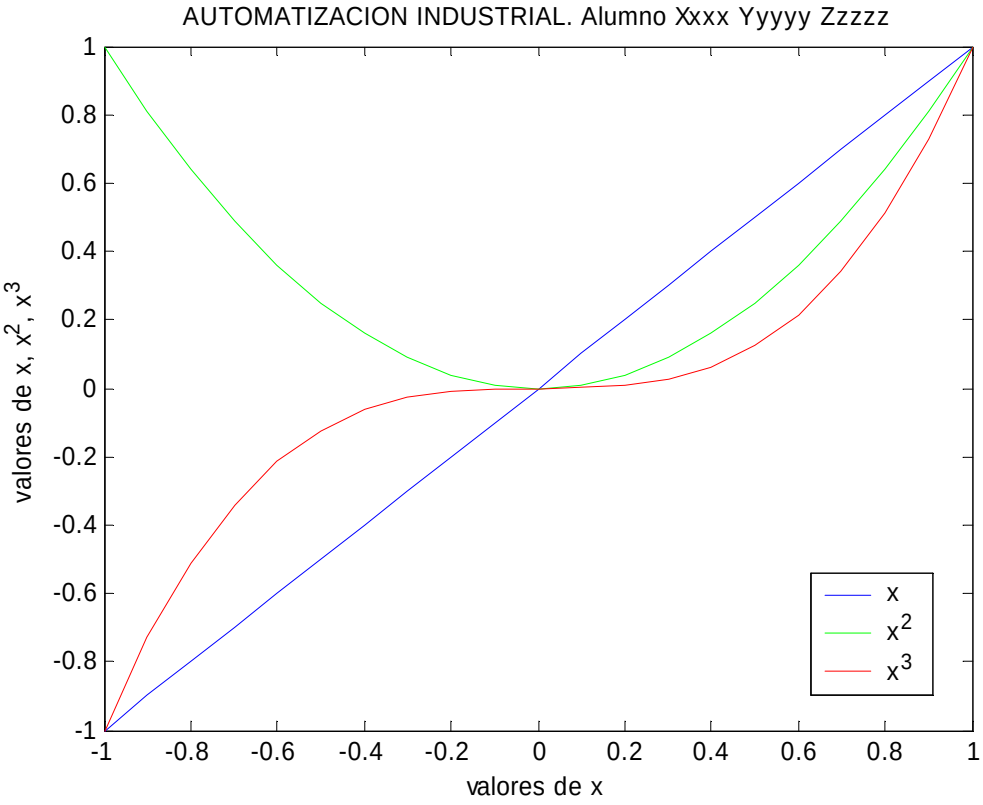
*La información necesaria sobre la función **plot** puede ser encontrada en la ayuda de Matlab 6.5 en:*

Matlab help -> Graphics -> Basic plotting

Matlab help -> Graphics -> Formatting graphs

EJERCICIO 1

Obtener un gráfico con un formato similar al anterior pero que, en lugar de representar las funciones seno y coseno entre 0 y 2*pi radianes, represente las funciones x, x2 y x3 en el intervalo -5<x<5. El aspecto del gráfico debe ser similar al que se muestra, donde Xxxx, Yyyy y Zzzz deben ser el nombre y los dos apellidos del alumno:



8. TRANSFORMADAS Y ANTITRANSFORMADAS

Matlab permite obtener transformadas y antitransformadas de Laplace mediante su módulo de matemática simbólica.

Procedimiento:

- Declarar una variable simbólica con la instrucción **syms**
- Obtener la transformada para una expresión definida utilizando la variable simbólica anterior

Instrucciones de Matlab correspondientes a cada una de las transformadas:

- **laplace** transformada de Laplace
- **ilaplace** transformada inversa de Laplace

Ejemplo:

Obtendremos la antitransformada de Laplace de la siguiente expresión: $$F(s)=\frac{2}{s\cdot(s+0.5)}$$

Las instrucciones de Matlab que utilizaremos serán:

```
» syms s
» ilaplace (2/(s*(s+0.5)))

ans =
4-4*exp(-1/2*t)
```

Por lo tanto, hemos obtenido como respuesta: $f(t)=4-4\cdot e^{-0.5t}$

Comprobación: haciendo el procedimiento inverso buscaremos la transformada de Laplace de f(t):

```
» syms t
» laplace (4-4*exp(-0.5*t))

ans =
 4/s-4/(s+1/2)
```

Se obtiene como resultado:

$$F(s)=\frac{4}{s}-\frac{4}{(s+0.5)}=\frac{4(s+0.5)}{s\cdot(s+0.5)}-\frac{4\cdot s}{s\cdot(s+0.5)}=\frac{2}{s\cdot(s+0.5)}$$

... que es la misma función F(s) de partida

Como ejercicio, se obtendrán antitransformadas de Laplace para las siguientes funciones:

$$F_1(s)=\frac{s^2+2s+3}{(s+1)^3} \qquad f_1(t)=t^2\cdot e^{-t}+e^{-t}$$

$$F_2(s)=\frac{5s+5}{s\cdot(s^2+2s+5)} \qquad f_2(t)=1-e^{-t}\cdot\left[\cos(2t)-2\cdot sen(2t)\right]$$

9. REPRESENTACIÓN DE SISTEMAS CONTINUOS

Matlab también permite obtener la respuesta de sistemas continuos ante distintas señales de entrada.

La forma de representar un sistema continuo es Matlab es mediante su función de transferencia en **s**, a través de la instrucción **tf.**

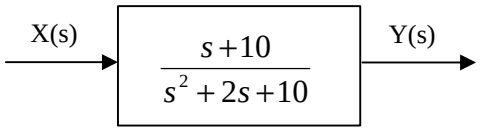
Forma de utilizar la instrucción tf:

```
nombre_sistema = tf (numerador, denominador)
```

- cada sistema que creemos se identificará mediante su nombre
- vector con los coeficientes del polinomio del numerador
- vector con los coeficientes del polinomio del denominador

Ejemplo:

Queremos representar el siguiente sistema continuo:



La instrucción de Matlab a utilizar será:

```
» sis1 = tf([1 10], [1 2 10])

Transfer function:
     s + 10
---------------
s^2 + 2 s + 10
```

A partir de este momento, la variable sis1 representará en Matlab el sistema correspondiente a esa función de transferencia.

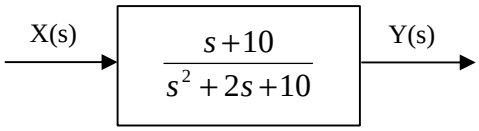
10. RESPUESTA A LA SEÑAL ESCALÓN

La instrucción **step** sirve para calcular la respuesta a escalón de cualquier sistema previamente definido. Caben dos posibilidades:

- Obtener la representación gráfica de la respuesta
- Obtener los valores numéricos de la respuesta

Representación gráfica de la respuesta:

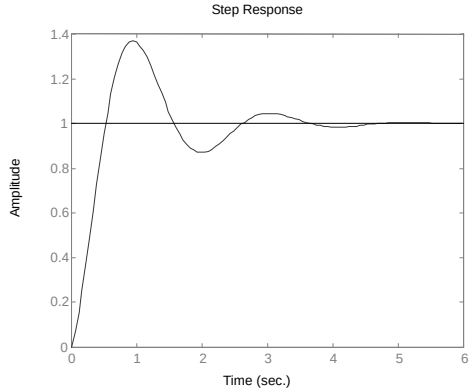
Obtendremos como ejemplo la respuesta a escalón del sistema continuo definido anteriormente:



Si suponemos que cuando creamos el sistema con la instrucción **tf** le dimos el nombre **sis1**, entonces la instrucción Matlab a teclear será la siguiente:

```
» step(sis1)
```

Y el resultado será el siguiente gráfico:



El gráfico representa el valor de la salida del sistema ante entrada escalón

Obtención de los valores numéricos de la respuesta:

En este caso lo que deseamos no es obtener el gráfico sino descargar en una variable el valor de la respuesta en cada instante de tiempo.

El formato para la instrucción es, en este caso:

```
» [y,t] = step(sis1);
```

Y el resultado será el siguiente:

- El vector **t** contendrá los instantes de tiempo para los que se ha calculado el valor de la salida
- El vector **y** contendrá los valores de la salida correspondientes a cada instante de tiempo

NOTA: recordemos que el punto y coma al final de la expresión sirve para que Matlab no muestre en pantalla el resultado de la operación; en otro caso habrían aparecido las ristras de valores de las variables.

Como comprobación, podemos consultar un valor cualquiera de la variable t y de la variable y; por ejemplo el valor que hace el número 25 de todos los calculados:

```
» t(25)

ans =
    1.3252

» y(25)

ans =
    1.1786
```

Vemos que el valor 25 corresponde al instante de tiempo 1.3252 segundos y que el valor de la señal de salida en ese instante de tiempo es 1.1786. Podemos comprobar estos valores aproximadamente sobre el gráfico de la respuesta que obtuvimos antes.

Disponer de los valores numéricos de los datos es útil para realizar cualquier tipo de operación matemática, como buscar el máximo, obtener el valor exacto en un instante de tiempo concreto, etc.

Forma de obtener la respuesta para un escalón no unitario:

Lo visto anteriormente considera que al sistema se le aplica un escalón de valor uno. Para escalones no unitarios basta con multiplicar el sistema por el valor del escalón.

Por ejemplo, para obtener la respuesta a un escalón de amplitud 5 bastará con teclear estas instrucciones:

```
» step(5*sis1)
```

o bien:

```
» [y,t] = step(5*sis1);
```

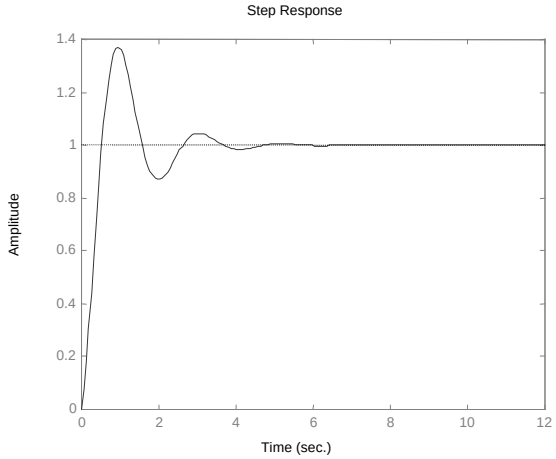
Forma de obtener la respuesta para instantes de tiempo posteriores:

En el ejemplo realizado, Matlab calcula la respuesta del sistema hasta el instante t = 6 segundos (se puede comprobar sobre el gráfico). Si se desea obtener la respuesta para instantes posteriores basta con especificar un valor para el tiempo final en la instrucción step.

Por ejemplo, si qeremos obtener la respuesta ante escalón del sistema **sis1** no hasta el instante t=6 sino hasta el instante **t=12** deberíamos teclear el siguiente comando Matlab:

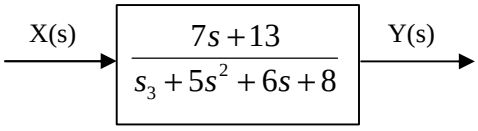
```
» step(sis1, 12)
```

Y el resultado sería el que mostramos en el gráfico que aparece a continuación:

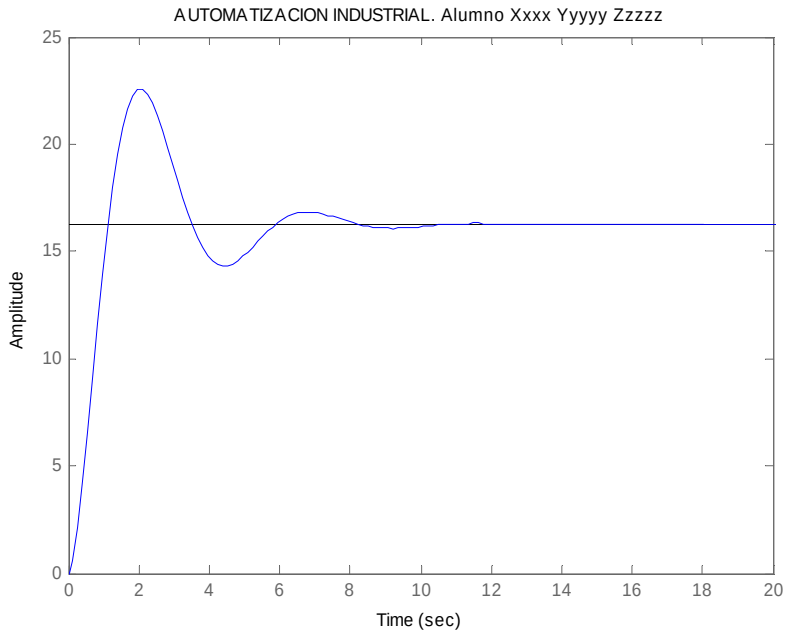


EJERCICIO 2

Obtener un gráfico que represente la respuesta ante una entrada escalón de amplitud **10** durante los primeros **20** segundos del sistema siguiente:



El resultado debe ser similar al mostrado en la figura (atención al título):



NOTA: EL INFORME COMPLETO DE LA PRÁCTICA DEBE CABER EN UNA SÓLA CARA Y SÓLO DEBE CONTENER EL NOMBRE Y LOS DOS GRÁFICOS PEDIDOS