TEORÍA DE SISTEMAS

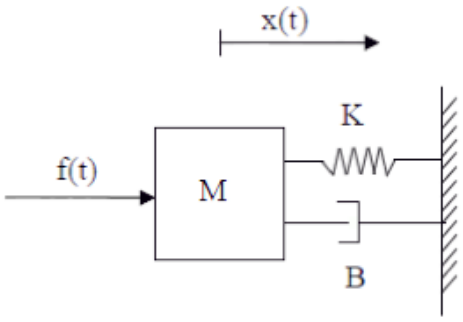
PRÁCTICA 3: ASPECTOS AVANZADOS MATLAB/SIMULINK

1. REPRESENTACIÓN EN MATLAB DE LOS RESULTADOS DE UNA SIMULACIÓN

En la práctica previa se ha utilizado el elemento **Scope** para representar los valores que toma una señal cualquiera en un esquema Simulink. Sería interesante disponer, no sólo de la representación de esas señales sino también de sus valores numéricos, para poder trabajar posteriormente sobre ellos (por ejemplo, para poder calcular el valor máximo y mínimo de una señal y los instantes de tiempo en que se producen, etc).

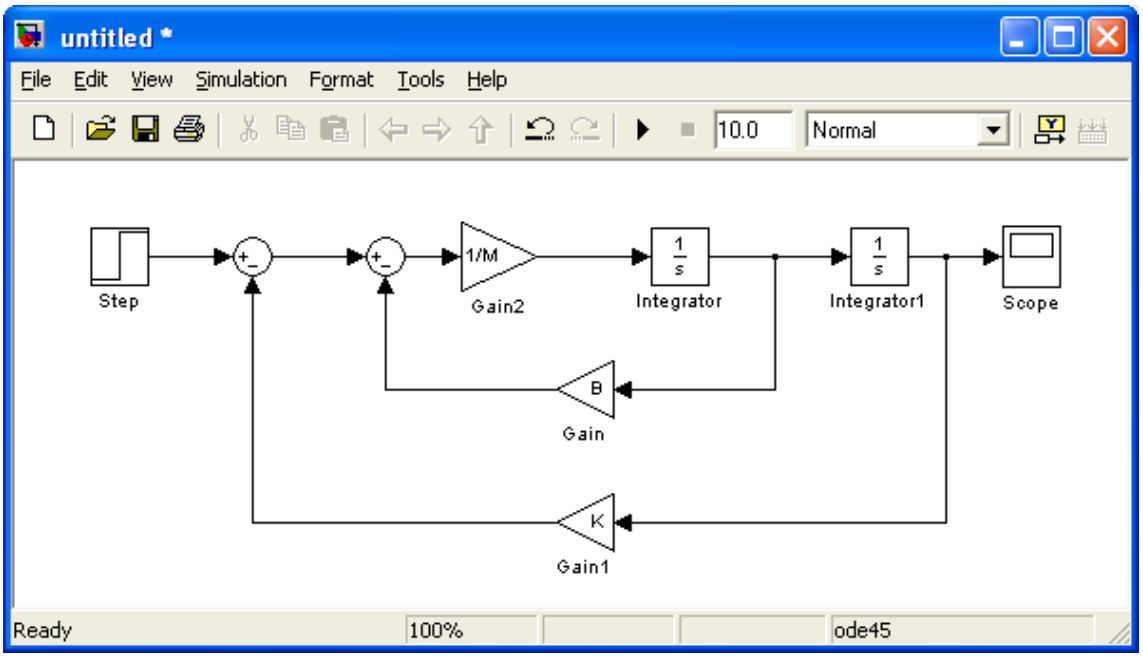
El elemento **Scope** permite, al mismo tiempo que representa los resultados de una simulación, guardar esos resultados en una variable de Matlab, de modo que se puedan consultar sus valores desde la ventana de comandos.

Como ejemplo volveremos a trabajar sobre el esquema utilizado en la práctica anterior para simular el comportamiento de un conjunto masa-muelle amortiguador. El aspecto del esquema es el que se muestra a continuación:



$$f(t) = M \cdot \frac{d^2x(t)}{dt^2} + B \cdot \frac{dx(t)}{dt} + K \cdot x(t)$$

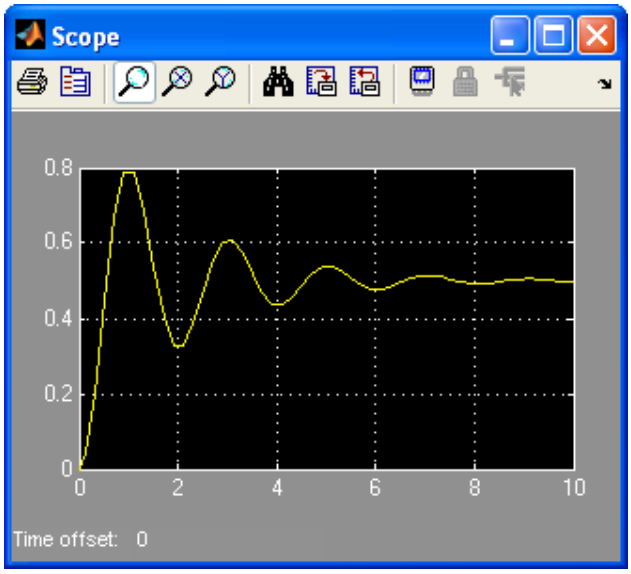
$$K = 10;\ B = 1;\ M = 1;$$



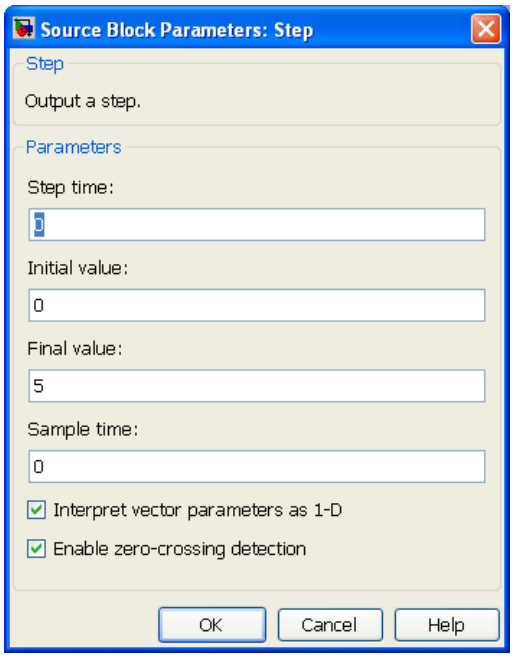
Se realizará la simulación de este esquema con los siguientes parámetros:

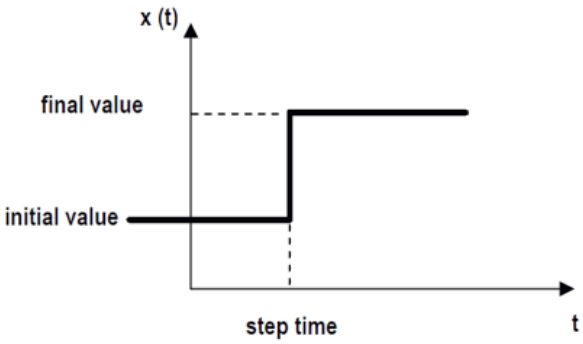
- Tiempo de simulación: 10 segundos.
- Amplitud del escalón de entrada: 5 unidades.

El resultado debería ser igual al que muestra la siguiente figura:



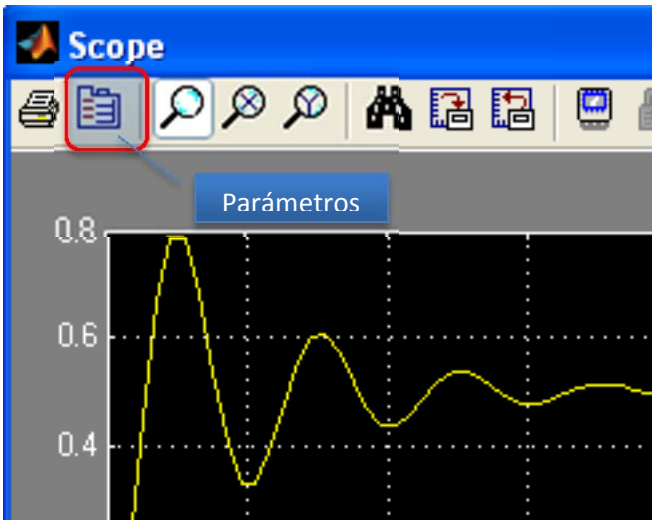
***Nota**: en caso de no obtener exactamente estos resultados es posible que la configuración del bloque escalón no sea la correcta. En la figura siguiente se muestran los parámetros del bloque escalón y su significado sobre la representación gráfica del mismo:*





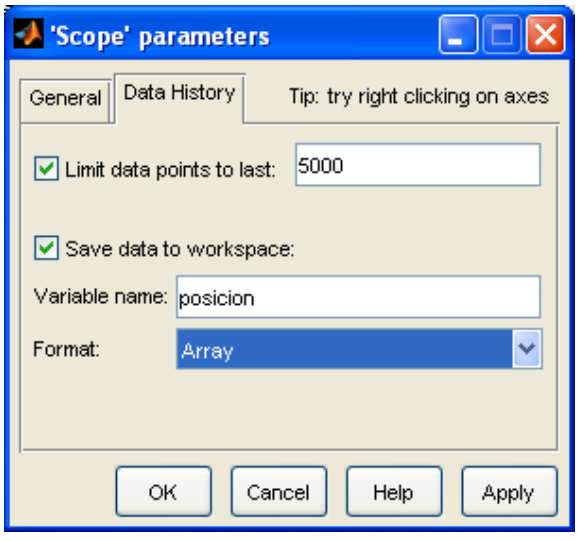
*En general, el escalón se producirá en el instante cero (no estará retrasado) con lo que el parámetro **step time** deberá ser siempre cero; el valor inicial del escalón (**initial value**) también será cero; y el valor final (parámetro **final value**) deberá indicar la amplitud del escalón, que en este caso particular es cinco.*

Ahora haremos click sobre el botón **Parameters** del elemento **Scope**, para abrir la ventana de configuración:



En la ventana de configuración deberemos seleccionar la solapa **Data History** y en ella marcaremos la opción '**Save data to workspace'** o, lo que es lo mismo, 'guardar los datos en el espacio de trabajo de Matlab'. Será necesario indicar dos parámetros:

- En primer lugar, tendremos que asignar un nombre a la variable en la que deseamos guardar los datos. Por defecto, esta variable es '**ScopeData'**, pero nosotros cambiaremos ese nombre por **'posicion'** (sin acento para evitar problemas) que es el dato que representa ese osciloscopio en nuestro sistema (posición del extremo del conjunto muelle-amortiguador).
- En segundo lugar, será necesario especificar un formato para los datos. Simulink ofrece tres posibilidades: **'Structure with time'**, '**Structure'** y '**Array'**. Nosotros elegiremos este último formato, que quiere decir que los datos se guardarán en forma de vector.



Una vez hecho esto, lanzaremos de nuevo la simulación. El resultado sobre la ventana de **Simulink** será el mismo, pero en la ventana de comandos de **Matlab** podremos comprobar cómo se han creado dos nuevas variables. Para ello bastará teclear el comando **who**:

```
>> who
Your variables are:
posicion tout
```

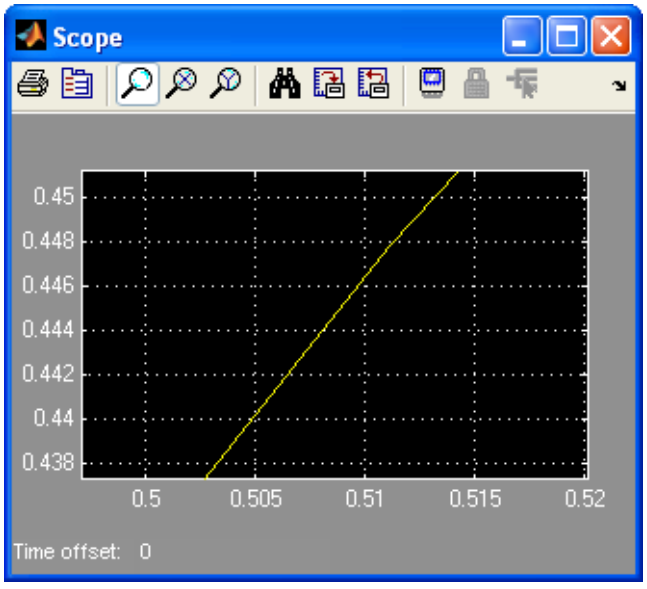
Se han creado dos variables: **posicion** y **tout**. La única variable que nos interesa es la primera de ellas. Si comprobamos el tamaño de la variable con el comando **size**, veremos que se trata de un vector de 2 columnas y 58 filas (este último dato puede variar según los ordenadores):

```
>> size(posicion)
ans =
      58 2
```

La primera de las columnas contiene instantes de tiempo y la segunda contiene los valores que toma la variable que se representa en el osciloscopio en cada instante. Si representamos las primeras 10 filas de la variable podremos comprobar esto (nota: se usa la instrucción **'format long'** para mostrar más cifras decimales):

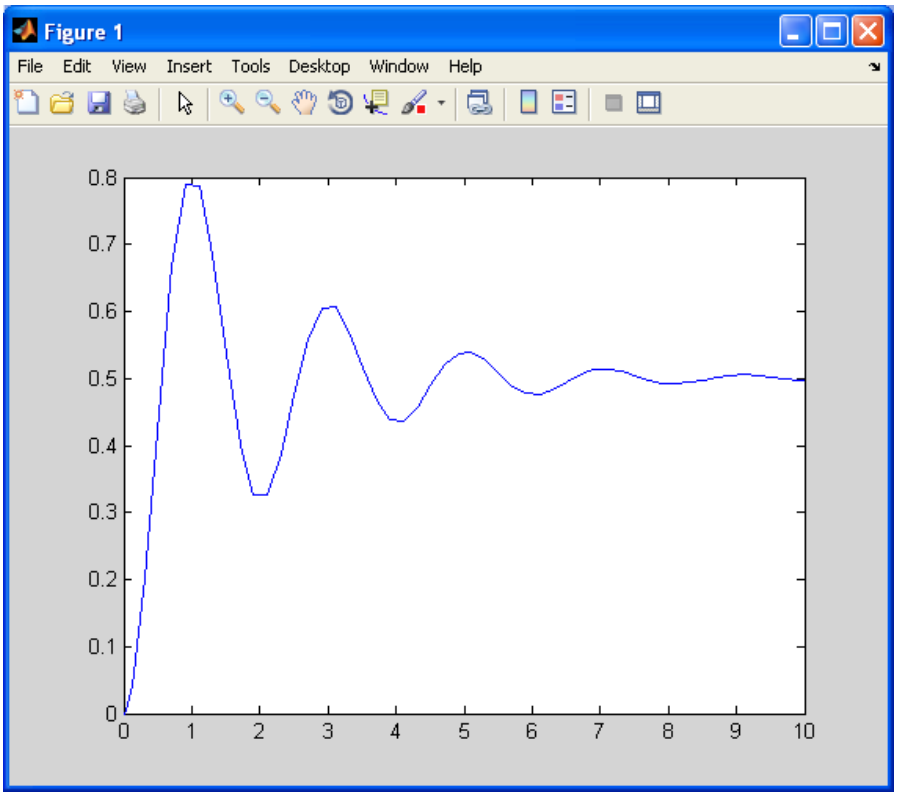
```
>> format long
>> posicion (1:10,:)
ans =
      0                   0
      0.00000000000000 0.00000000000000
      0.00004019018290 0.00000000403807
      0.00024114109742 0.00000014536088
      0.00124589567003 0.00000387902391
      0.00626966853305 0.00009806359195
      0.03138853284814 0.00243553350653
      0.13244976834872 0.04137887820543
      0.31114891413343 0.20203969791657
      0.51114891413343 0.44779711805874
```

La primera de las columnas, como se ha dicho, corresponde a los instantes de tiempo; y podemos comprobar cómo no están separados uniformemente. La segunda columna corresponde a los valores que toma la señal, y podemos comprobar que los datos son correctos haciendo zoom sobre el gráfico que aparece en el elemento **Scope** (por ejemplo, en el instante **0.511** segundos el valor de la señal es **0.448** aproximadamente):



Desde la ventana de comandos de **Matlab**, también será posible representar mediante la instrucción **plot** la señal guardada en la variable posición. Dado que la primera columna contiene valores de tiempo y la segunda columna contiene valores de posiciones, la forma correcta de la instrucción **plot** debería ser:

```
>> plot(posicion(:,1), posicion(:,2))
```



Supongamos que nos interesa calcular el valor máximo de la señal. Dado que los valores de la señal están contenidos en la segunda columna de la variable **posicion**, bastará con utilizar el comando **max** de Matlab (se recomienda consultar la ayuda de Matlab acerca de este comando):

```
>> [maximo, indice] = max(posicion(:,2))
maximo =
      0.78834718133495
indice =
      12
```

Los valores pueden ser ligeramente distintos en diferentes ordenadores debido a las precisiones de los cálculos. En cualquier caso, el comando **max** recorre la segunda columna de la variable **posicion** y nos devuelve su valor máximo y cuál es la fila en la que se produce ese valor (lo que hemos denominado índice). Si queremos comprobar a qué instante de tiempo corresponde ese máximo, bastará con comprobar el valor que toma la primera columna de la variable **posicion** para ese mismo índice (nota: se utiliza la instrucción **format short** para mostrar menos cifras decimales).

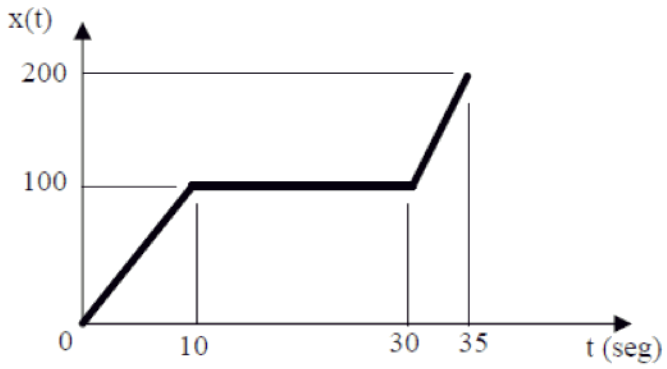
```
>> format short
>> posicion(indice, :)
ans =
      0.9111 0.7883
```

En resumen, hemos obtenido como resultado que la posición de la masa móvil (x) en el ejemplo del sistema muelle-amortiguador alcanza un valor máximo de **0.7883** metros en el instante **0.9111** segundos.

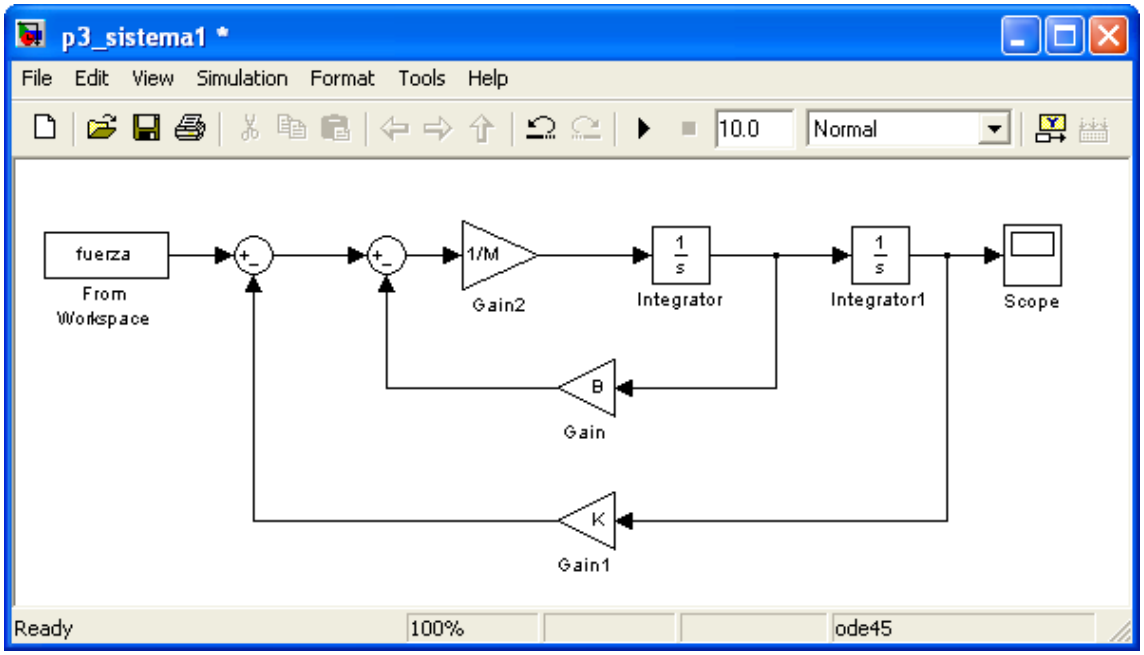
2. INTRODUCCIÓN DE SEÑALES ARBITRARIAS EN SIMULINK

En la práctica anterior se comprobó como Simulink dispone de muchas señales de entrada, válidas para la mayor parte de las aplicaciones: se dispone de las señales escalón, rampa, señal senoidal, tren de pulsos, etc.

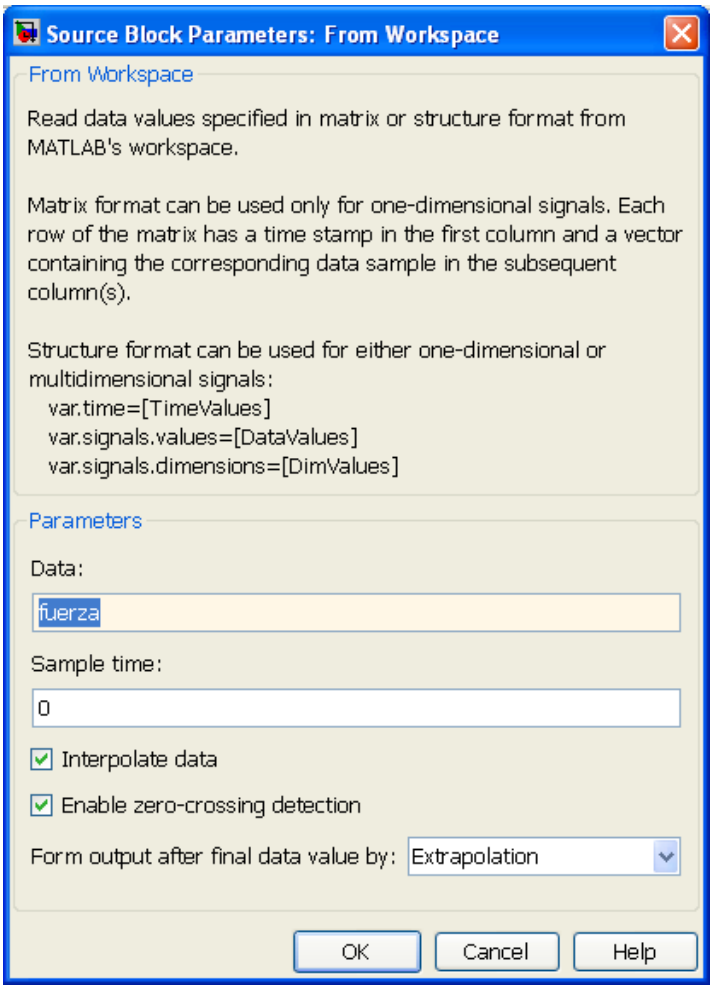
No obstante, en algunas ocasiones se desea utilizar señales de forma arbitraria, no disponibles directamente en Simulink. Por ejemplo, podríamos desear utilizar una señal de entrada (fuerza aplicada al conjunto muelle-amortiguador en el sistema sobre el que se está trabajando en esta práctica) como la que se muestra a continuación:



La forma más sencilla de introducir una señal como la propuesta es mediante el bloque **From Workspace** de la categoría **Sources**. A continuación se muestra el aspecto que debe tener el esquema **Simulink** de la práctica una vez sustituido el bloque escalón por el bloque **From workspace**:



Una vez modificado el esquema **Simulink**, haremos clic sobre el elemento **'From Workspace'** para acceder a sus parámetros de configuración. Aparecerá una ventana como la que se muestra a continuación:



El único parámetro que nos interesa por el momento es el nombre de la variable de **Matlab** de la que **Simulink** tomará los datos para utilizarlos como entrada. Por defecto, esta variable es **simin**. Nosotros cambiaremos este nombre por otro más explicativo: dado que la entrada, en el sistema considerado, es la fuerza aplicada sobre el conjunto muelle-amortiguador, llamaremos a esta variable **fuerza**.

El formato que debe tener esta variable es similar al de las variables guardadas en **Matlab** a través del bloque **Scope**: la primera columna debe contener los valores de los instantes de tiempo y la segunda los valores de la señal.

Un dato importante que hay que fijar es el intervalo entre cada dos instantes de tiempo. Cuanto más pequeño sea este intervalo, mejor será la definición de la señal. En este ejemplo utilizaremos un intervalo de **0.1** segundos.

A continuación pasaremos a definir la señal (véase el gráfico anterior de **x(t)**). Lo más cómodo será definirla por tramos, utilizando un tramo para cada trazo recto de la señal. Además, y para simplificar la creación de la señal, se usarán variables distintas para el tiempo y para la señal. Los tiempos se guardarán en la variable **tmp** y los valores de la señal (o datos) en la variable **dat**.

Primer tramo: entre 0 y 10 segundos. La variable tiempo (vector **tmp1**) deberá tomar valores entre 0 y 10 a intervalos de 0.1; y la señal (vector **dat1**) deberá responder a la ecuación de la recta **x=10t**.
Esto lo conseguimos con las siguientes sentencias de Matlab:

```
>> tmp1 = [0:0.1:10];
>> dat1 = 10*tmp1;
```

Segundo tramo: entre los instantes 10.1 segundos y 30 segundos. la variable tiempo (vector **tmp2**) deberá tomar valores entre 10.1 y 30 a intervalos de 0.1; y la señal (vector **dat2**) deberá responder a la ecuación de la recta **x=100**. Esto lo conseguimos con las siguientes sentencias de Matlab:

```
>> tmp2 = [10.1:0.1:30];
>> dat2 = 100*ones(size(tmp2));
```

La segunda instrucción crea un vector de 200 elementos, tantos elementos como tiene el vector **tmp2,** todos ellos con valor 100.

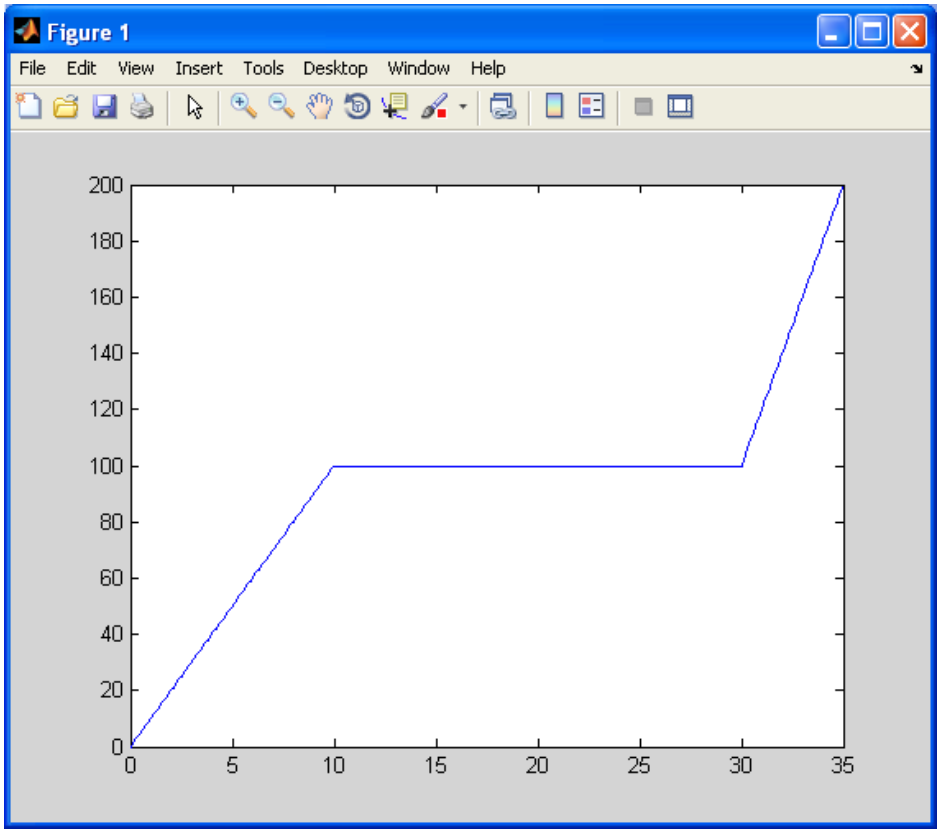
Tercer tramo: entre 30.1 y 35 segundos. La variable tiempo (vector **tmp3**) deberá tomar valores entre 30.1 y 35 a intervalos de 0.1; y la señal (vector **dat3**) deberá responder a la ecuación de la recta **x=20t-500**. Esto lo conseguimos con las siguientes sentencias de Matlab:

```
>> tmp3 = [30.1:0.1:35];
>> dat3 = 20*(tmp3-30)+100;
```

El último paso es crear las variables **tmp** y **dat** como concatenación de las variables **tmp1, tmp2, tmp3** y **dat1, dat2, dat3** disponibles:

```
>> tmp = [tmp1, tmp2, tmp3]
>> dat = [dat1, dat2, dat3]
>> plot(tmp, dat)
```

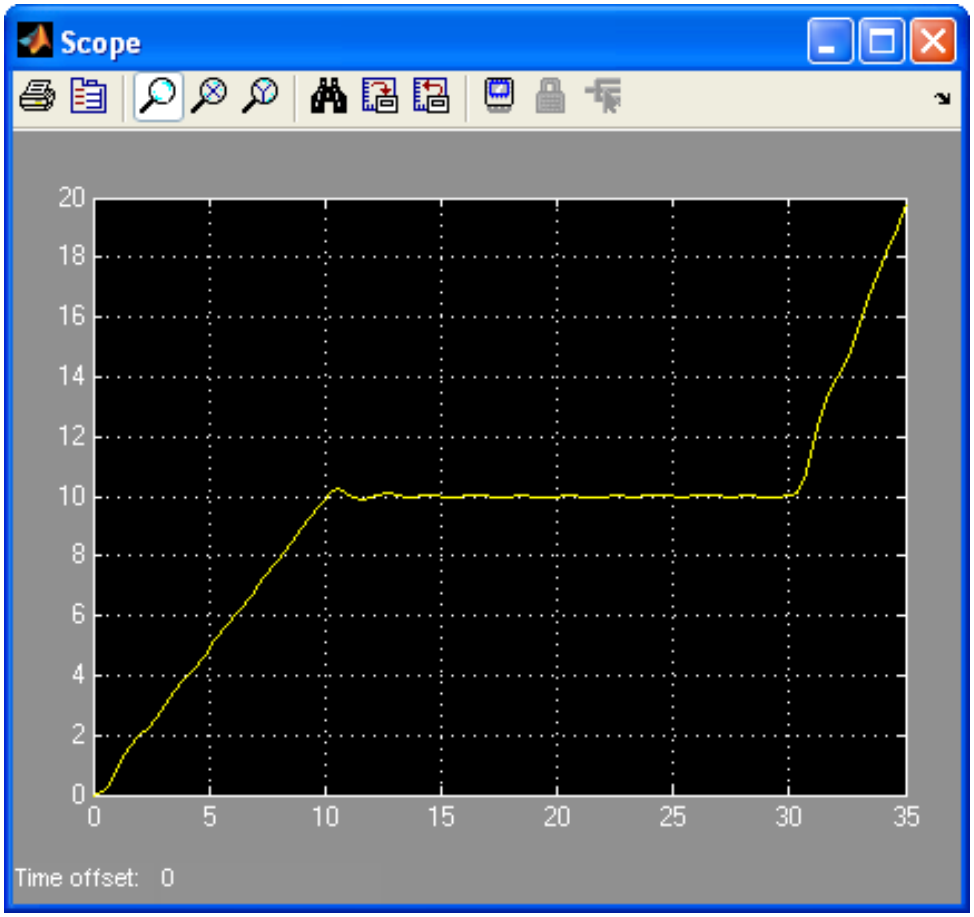
Para comprobar que la señal se ha generado correctamente, lo más sencillo es utilizar el comando **plot** de Matlab: la instrucción **plot(tmp, dat)** debería producir el siguiente resultado:



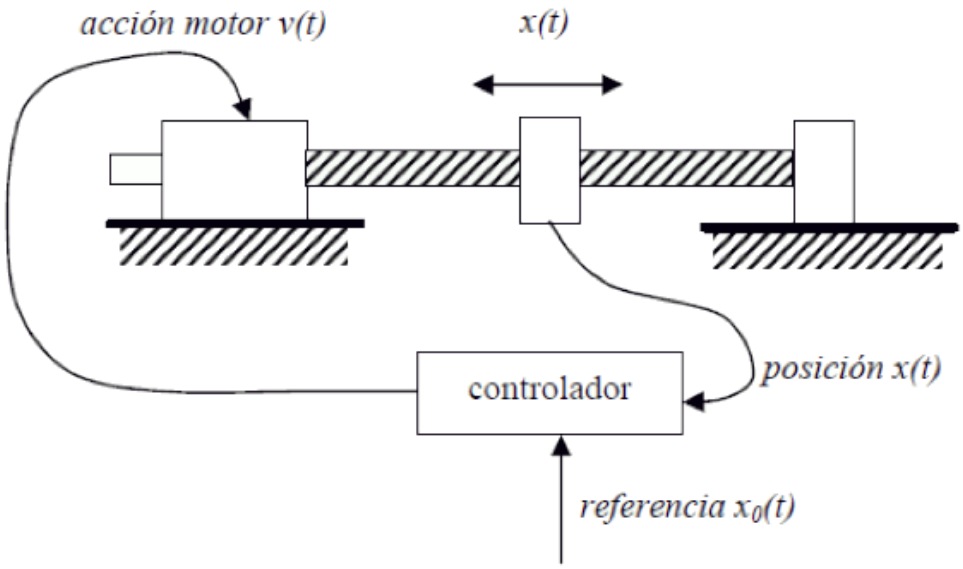
Una vez comprobado que las variables **tmp** y **dat** contienen valores correctos, generaremos a partir de ellas la variable fuerza necesaria para el bloque **From Workspace**. La primera columna deberá contener los tiempos y la segunda los datos, dado que tanto **tmp** como **dat** son vectores fila, se deberán trasponer, tal y como indican las siguientes sentencias de Matlab (atención a la comilla final que indica la operación de trasposición):

```
>> fuerza(:,1)=tmp';
>> fuerza(:,2)=dat';
```

Con la variable fuerza creada, es posible lanzar la simulación. Dado que la señal está definida durante 35 segundos, el tiempo de simulación se fijará exactamente en 35 segundos (desde la opción **Simulation parameters** del el menú **Simulation**). El resultado sobre el osciloscopio se muestra en la figura siguiente, y se debe apreciar cómo la salida del sistema (posición) reproduce aproximadamente los valores de la entrada (fuerza aplicada) con pequeñas oscilaciones.



EJERCICIO A REALIZAR: SERVOMECANISMO DE POSICIÓN



El objetivo es simular el comportamiento de un servomecanismo de posición. El funcionamiento del mecanismo es el siguiente: existe un controlador al que el usuario introduce como dato una referencia de posición (posición deseada para la pieza móvil). El controlador actúa sobre un motor eléctrico que, a través de un husillo, mueve la pieza. Para calcular la tensión a aplicar al motor, el controlador compara en cada instante la posición real de la pieza con la posición pedida y en función de la diferencia entre las posiciones aplica más o menos tensión, en uno u otro sentido. Cuando la posición es la pedida, el error es cero y por tanto el controlador deja de aplicar tensión al motor.

Las ecuaciones del sistema son las siguientes:

$$v(t) = Ri(t) + fcem(t)$$

$$fcem(t) = K_v \frac{dx(t)}{dt}$$

$$p(t) = K_p \cdot i(t)$$

$$p(t) = J\frac{d^2x(t)}{dt} + B\frac{dx(t)}{dt}$$

$$v(t) = K_c \cdot [x_0(t) - x(t)] \qquad \textit{Ecuación controlador}$$

Las variables que aparecen representan las siguientes magnitudes:

- **x(t)**: posición del elemento móvil del servomecanismo (variable de **salida**)
- **x_0(t)**: posición de referencia (variable de **entrada**)
- **v(t)**: tensión entre los terminales del motor
- **i(t)**: intensidad que circula por el motor
- **fcem(t)**: fuerza contraelectromotriz en el motor
- **p(t)**: par producido por el motor

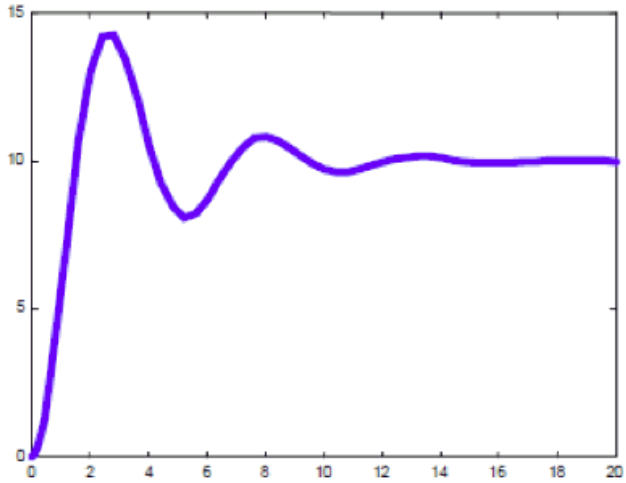
Los valores que tomaremos inicialmente para las constantes serán los siguientes:

- **R** = 1.25 (resistencia de los devanados del motor)
- **J** = 0.8 (momento de inercia del conjunto)
- **B** = 0.5 (coeficiente de rozamiento viscoso)
- **K_P** = 3 (constante de par del motor)
- **K_C** = 0.5 (constante proporcional del controlador)
- **K_V** = 0.01 (constante de velocidad del motor)

Supondremos que tanto **x(t)** como $\mathbf{x_0(t)}$ están expresadas en centímetros, y que el resto de las variables y las constantes están expresadas en unidades coherentes entre si, de modo que no se realizará ninguna conversión de unidades.

Se pide, en primer lugar, crear un esquema Simulink que represente el comportamiento del sistema de acuerdo con las ecuaciones planteadas. Tal y como se indica, la variable de entrada será la posición de referencia $\mathbf{x_0(t)}$ y la variable de salida la posición real del elemento móvil **x(t).**

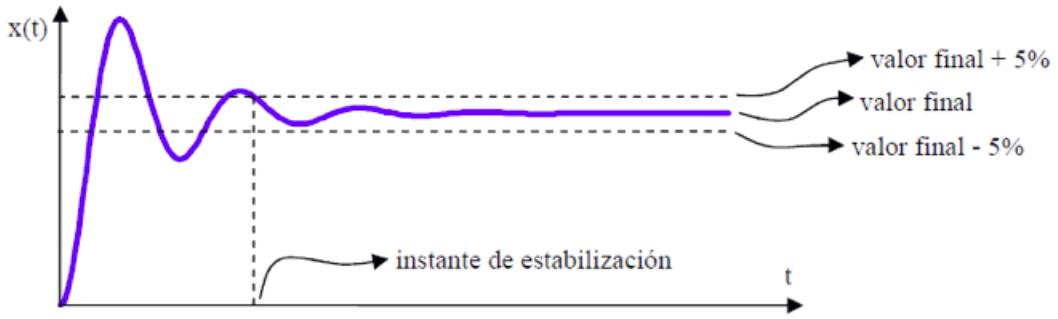
El gráfico siguiente muestra la salida que debe producir el sistema ante una entrada en forma de escalón de 10cm (la referencia aumenta bruscamente de 0 a 10cm). El sistema se ha simulado durante los primeros 20 segundos de funcionamiento:



Debe comprobarse como la salida sigue a la entrada: una variación de 10cm en la posición de referencia ocasiona en régimen permanente una variación de 10cm en la posición del elemento móvil. Las sobreoscilaciones que se producen en los instantes iniciales se analizarán posteriormente.

Una vez comprobado el funcionamiento del sistema se deberá llevar la señal de salida a una variable de Matlab, utilizando el bloque **Scope**, y crear un programa Matlab que se guardará como **analiza.m** y que deberá realizar los siguientes cálculos sobre la señal de salida:

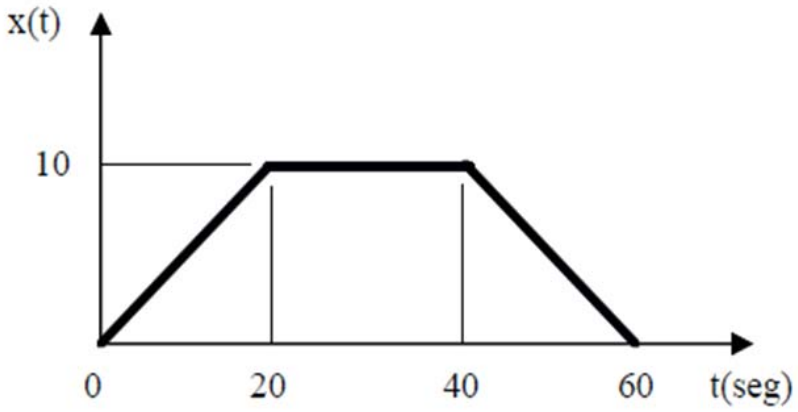
* Cálculo del valor final de la señal (valor en régimen permanente)
* Cálculo del valor máximo (sobreoscilación) y el instante en el que se produce.
* Cálculo del instante a partir del cual la salida se puede considerar estabilizada. Este instante lo consideraremos como el punto a partir del cual la salida permanece dentro de una franja del ±5% del valor final, tal y como muestra el gráfico siguiente:



La declaración de la función debe ser como la siguiente:

```
function [xfin, xmax, tmax, testab] = analiza (posicion)
```

Como último punto de la práctica, se comprobará la respuesta del sistema ante una señal como la mostrada en el gráfico siguiente, que deberá ser introducida al sistema mediante el bloque **From Workspace**:



Ayuda: Ejemplo de utilización del bucle **for** y la sentencia **if** en Matlab. Se recomienda copiar y probar el siguiente programa:

```
% ejemplo de utilización de las sentencias if y for en Matlab

x = [0:0.1:8*pi]'; %vector columna
y = sin(x);
plot(x, y, 'b') %gráfico del seno
[ndatos, col] = size(y); %filas (num. de elementos) y columnas del vector

%recorrer el vector y desde el final al principio
for i=ndatos:-1:1 %i toma valores desde el número de elementos de y hasta 1
      if y(i)>=0
            y(i)=0.5; %cualquier valor positivo de y se convierte en +0.5
      elseif y(i)<0
            y(i)=-0.5; %cualquier valor negativo de y se convierte en -0.5
      end
end
hold
plot(x, y, 'r') %superponemos el gráfico del seno modificado
```

Se recuerda que para probar este programa será necesario guardar el mismo con el nombre 'ejemplo.m', actualizar el path de matlab, y lanzar el programa desde la línea de comandos tecleando:

```
>> ejemplo
```