Práctica 8. Aspectos avanzados Matlab-Simulink

Objetivo. Realizar operaciones de intercambio de datos entre el área de trabajo de Matlab y Simulink, configuración de los parámetros de la simulación y creación de subsistemas.

Introducción

En la práctica anterior se ha utilizado Simulink para la simulación de ecuaciones diferenciales mediante la construcción del diagrama de simulación basado en bloques integradores.

Para visualizar los resultados de la simulación se ha usado el bloque **scope**, el cual permite algunas funciones interactivas con el usuario para ayudar a analizar los resultados de la simulación, tales como: zoom horizontal y vertical, modificación de escalas, manejo de seis colores ordenados para graficas simultáneas: 1: Amarillo, 2: Magenta, 3: Cyan, 4: Rojo, 5: Verde, 6: Azul oscuro. Cuando el bloque contiene más de seis gráficas simultáneas, los colores vuelven a empezar en amarillo, magenta y así sucesivamente.

Exportando datos de Simulink a Matlab.

Cuando queremos hacer un procesamiento más detallado o una representación gráfica alternativa de los resultados de una simulación, el bloque **scope** puede configurarse para que al mismo tiempo que se visualizan los resultados de una simulación se vallan guardando esos resultados en una variable de Matlab, de modo que se puedan procesar sus valores desde el ambiente de trabajo de Matlab en un proceso *posterior* a la simulación.

Ejemplo:

Retomaremos el ejemplo de la práctica anterior, el cual era un sistema mecánico consistente en una masa **M** sometida a una fuerza externa **f(t)** y unida a un resorte de constante elástica **K**, y con un rozamiento viscoso **B**, como se describe en la figura 8.1

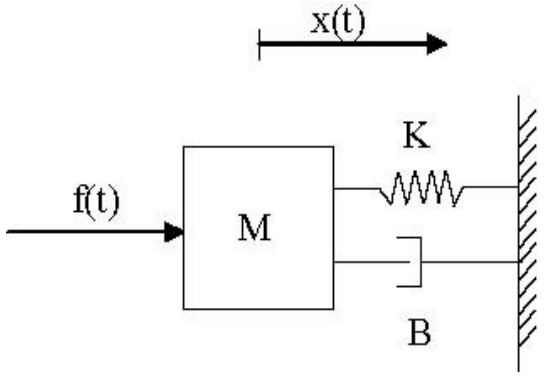

Figura 8.1.- Sistema mecánico masa-resorte-amortiguador.

La ecuación que describe la dinámica del sistema es

$$M\frac{d^2x(t)}{dt^2}+B\frac{dx(t)}{dt}+Kx(t)=f(t) \tag{8.1}$$

Y su diagrama de simulación correspondiente en Simulink es el mostrado en la figura 8.2

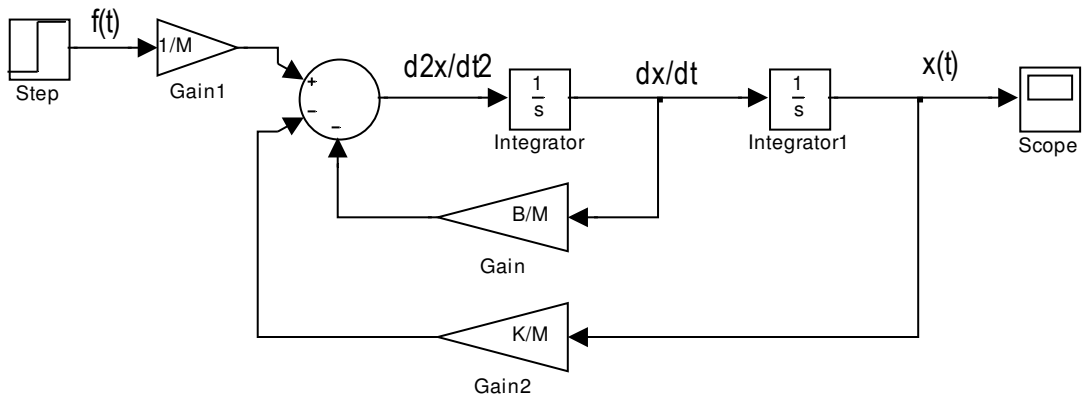

Figura 8.2.- Diagrama de simulación para el sistema masa-resorte-amortiguador.

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

- Tiempo de simulación: 10 seg.
- Amplitud del escalón de entrada: 5 unidades.
- tiempo inicial del escalón: 0 seg.
- Valores de las constantes del sistema M=1Kg. K=10 Nt seg/m, B=1 Nt.

En la figura 8.3 se muestra el comportamiento de la salida $x(t)$ del sistema, visualizado en el bloque **Scope** después de la simulación.

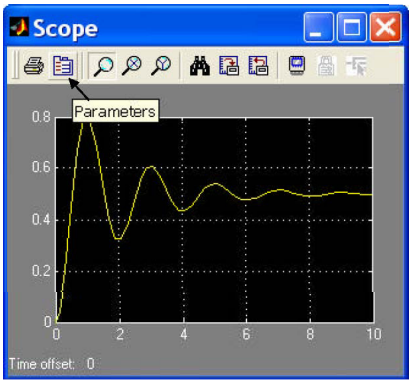

Figura 8.3.- Bloque **Scope** después de la simulación mostrando el botón **Parameters**.

Haciendo click sobre el botón **Parameters** del elemento **Scope**, para abrir la ventana de configuración de parámetros aparece la ventana de configuración deberemos seleccionar la pestaña **Data History** y en ella marcaremos la opción **'Save data to workspace',** es decir, 'guardar los datos en el espacio de trabajo de Matlab' como se muestra en la figura 8.4

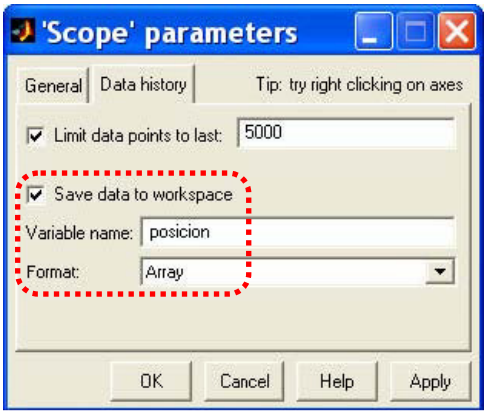

Figura 8.4.- Configuración de los parámetros del bloque Scope para exportar datos a Matlab

Será necesario indicar dos parámetros:

- Tendremos que asignar un nombre a la variable de Matlab en la que deseamos guardar los datos. Por default, esta variable es **'ScopeData'**, pero cambiaremos ese nombre por **'posicion'** (sin acento) que corresponde a la variable $x(t)$ que visualiza este osciloscopio en nuestro sistema.

- También se debe especificar un formato para los datos. Simulink ofrece tres posibilidades: **'Structure with time'**, **'Structure'** y **'Array'**. Elegiremos este último formato, que quiere decir que los datos se guardarán en forma de vector.

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

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

Se han creado dos variables nuevas: **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 número puede variar según la computadora usada)

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

La primera de las columnas contiene los instantes de tiempo t calculados en la simulación y la segunda contiene los valores que toma la variable $x(t)$ que se representa en el osciloscopio en cada instante. Para comprobar esto, podemos visualizar los primeros 10 pares de valores t, $x(t)$ como sigue (se usa la instrucción **`format long`** para mostrar más cifras decimales):

```
>> format long;
>> posicion(1:10,:)
```

```
ans =
                   0                   0
   0.000000000000000   0.000000000000000
   0.000040190182904   0.000000004038073
   0.000241141097425   0.000000145360881
   0.001245895670029   0.000003879023912
   0.006269668533048   0.000098063591952
   0.031388532848144   0.002435533506525
   0.132449768341823   0.041378878201275
   0.311148914122735   0.202039697904474
   0.511148914122735   0.447797118045490
```

La primera de las columnas, como se ha dicho, corresponde a los instantes de tiempo; y podemos observar que no están separados uniformemente.

Ya que tenemos los valores de la variable $x(t)$ en una variable de Matlab, podemos procesarlos mediante los comandos disponibles. Por ejemplo, para visualizar el comportamiento de la variable $x(t)$ desde la ventana de comandos de Matlab, podemos usar el comando `plot` como sigue para obtener la gráfica de la figura 8.5. Compárese con la obtenida en Simulink (figura 8.3).

```
>> t=posicion(:,1);
>> x=posicion(:,2);
>> plot(t,x); grid on
>> title('Respuesta de la posición del sistema ante un escalón de 5
unidades');
>> xlabel('tiempo (seg)');
>> ylabel('Posición x(t)');
```

Figura 8.5.- Gráfica de la variable $x(t)$ mediante el comando `plot`.

Podemos hacer cualquier cálculo deseado sobre los valores de la variable $x(t)$. Por ejemplo, para calcular el valor máximo de la señal, bastará con utilizar el comando `max` de Matlab:

```
>> [maximo, indice] = max(posicion(:,2))
maximo =  0.788347181331777
```

```
indice = 12
```

Los valores pueden ser ligeramente distintos en diferentes ordenadores debido a las precisiones de los cálculos. Para verificar 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.

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

Es decir, la posición $x(t)$ de la masa del sistema mecánico masa-resorte-amortiguador alcanza un valor máximo de 0.7883 metros en el instante 0.9111 segundos.

Configuración de parámetros de simulación.

Si observamos el resultado de la simulación del ejemplo anterior, podemos apreciar que es un poco burda, ya que aparece segmentada en líneas rectas (se aprecia mejor en la figura 8.5), esto se debe a que los parámetros de la simulación (especialmente el paso de integración) se dejaron los que Simulink pone por default. Si deseamos un resultado más preciso en la simulación debemos modificar estos parámetros accediendo a la ventana de configuración (ver figura 8.6).

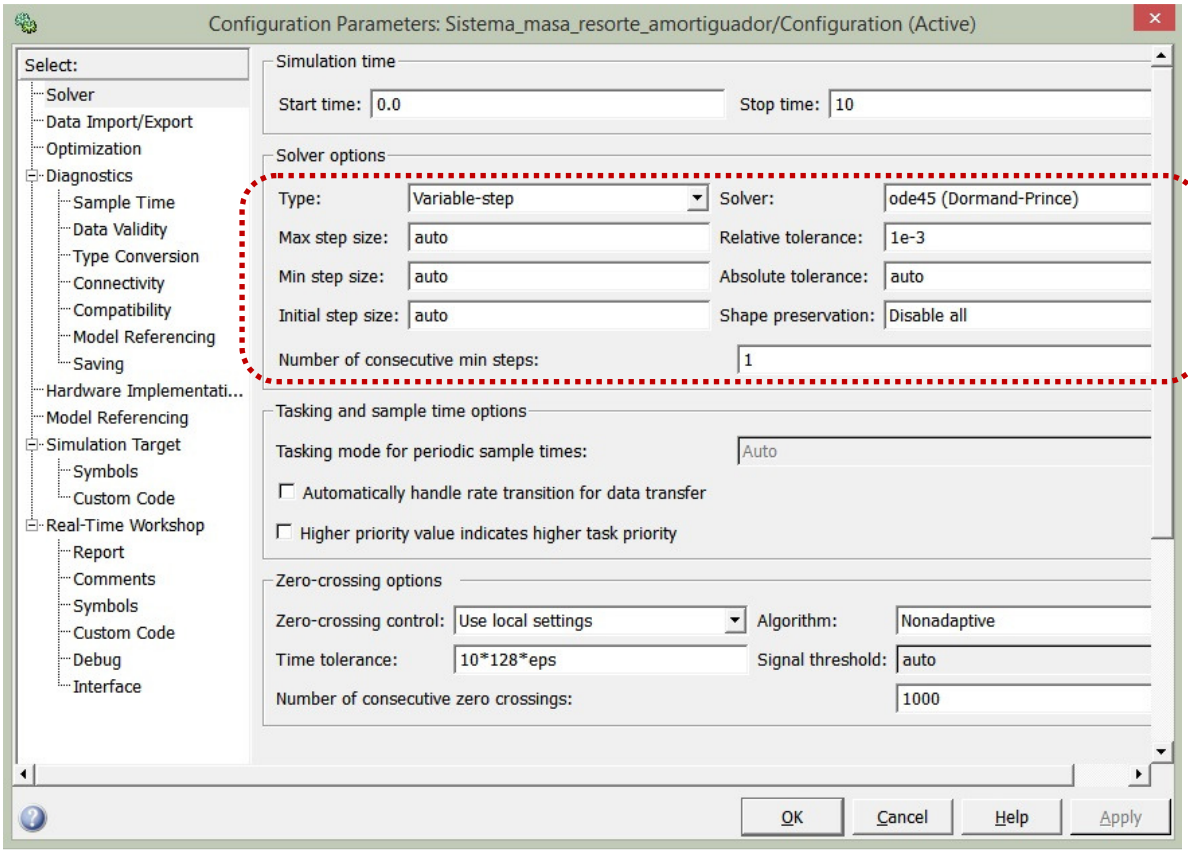

Figura 8.6.- Ventana para modificar los parámetros de la simulación.

En la figura 8.6, en la sección '**Solver options**' se observa que se tienen varias alternativas de parámetros a configurar. A continuación se enlistan los más importantes y algunas recomendaciones de cómo elegirlos:

- **Solver**: Es el método numérico a utilizar. Se recomienda probar siempre como primera opción ode45 (paso variable)
- **Max step size**: Es el tamaño máximo del paso de integración permitido. (Simulink mantiene siempre el paso de integración menor al valor especificado aquí).
- **Relative tolerance**: Es el tamaño de error máximo permitido individualmente en cada uno de los estados. (Este es el primer parámetro a modificar para mejorar la exactidud en una simulación)
- **Absolute tolerance**: Es el tamaño de error máximo permitido totalizado de todos los estados.

La regla teórica general para mejorar la exactitud de la simulación es: Elegir el paso de integración lo más pequeño posible, sin embargo esto no se debe hacer, pues entre más pequeño el paso más tiempo tarda la simulación. Por ello, Simulink, al igual que Matlab proporcionan maneras de mejorar la exactitud sin disminuir demasiado el paso de integración. UNa regla más práctica entonces sería:

☞ Disminuir la tolerancia relativa (Relative tolerance) hasta obtener la precisión deseada. Si no se mejora lo suficiente la exactitud entonces disminuir el paso de integración (Max step size) poco a poco hasta obtener la exactitud deseada.

☞ **Precauciones**: Nunca disminuir demasiado el paso de integración pues la simulación se podría quedar "colgada" por requerir demasiado tiempo en cada paso.

☞ Conforme se disminuye el paso de integración Simulink calcula más puntos de la solución. Se puede llegar a un punto en que el número de puntos almacenados en el bloque 'Scope' no sean suficientes para visualizar todos los puntos calculados. En este caso se deberá desmarcar la casilla '**Limit data points to last**' de la configuración del bloque 'Scope' como se muestra en la figura 8.7

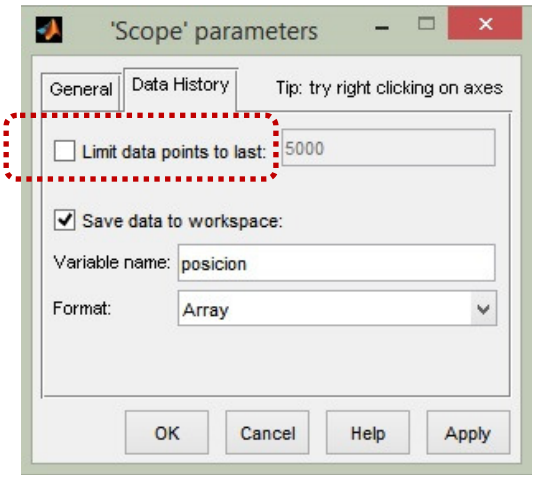

Figura 8.7.- Configuración de puntos visualizados en el bloque 'Scope'

En la figura 8.8 (compárese con la figura 8.5) se muestra el resultado de simular el ejemplo del sistema mecánico masa-resorte-amortiguador con los parámetros por default y con los parámetros modificados siguientes: Relative tolerance: 1e-10; Max step size: 1e-3.

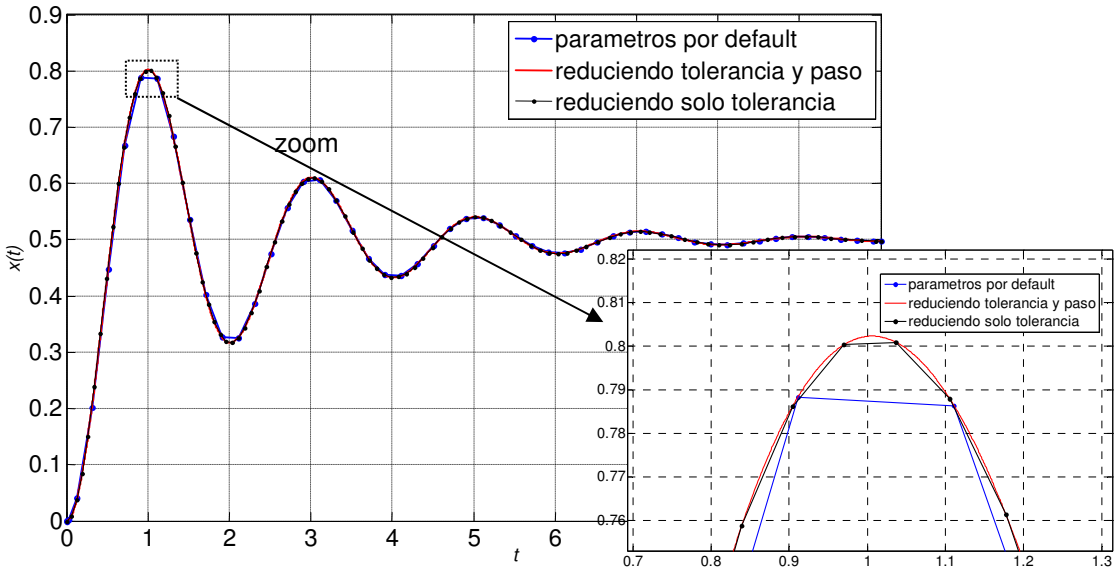

Figura 8.8.- Mejoramiento de la exactitud de la simulación modificando los parámetros.

Disminuyendo sólo la tolerancia relativa a 1e-10 se logra una mejora ligera pero apreciable en la figura 8.8. Simulink, en este caso Simulink calcula 99 puntos de la solución. Al disminuir tanto la tolerancia como el paso de integración se obtiene una mejora aun mayor (apreciable en el zoom), sin embargo, el costo es muy alto por haber disminuido el paso de integración, ya que Simulink ahora calcula 10,003 puntos de la solución, en lugar de sólo los 99 puntos que se habían calculado al disminuir solo la tolerancia relativa.

Importación de variables de Matlab a Simulink.

En los ejemplos de simulación anteriores se han utilizado valores de constantes que aparecen en el diagrama de simulación de simulink y cuyo valor es asignado desde Matlab. Por ejemplo, en el diagrama de simulación de la figura 8.2 las constantes K, M y B fueron asignadas desde Matlab, previo a la ejecución de la simulación con el comando **`sim`** de Matlab. Sin embargo, si en lugar de constantes queremos usar variables (señales que varían respecto al tiempo), el procedimiento es muy diferente.

Simulink dispone de muchas señales de entrada, válidas para la mayoría de las aplicaciones: señales escalón, rampa, sinusoidal, tren de pulsos, señales aleatorias, etc. Si se desea utilizar señales de forma arbitraria, no disponibles directamente en Simulink. Por ejemplo, si quisiéramos 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 en la figura 8.9

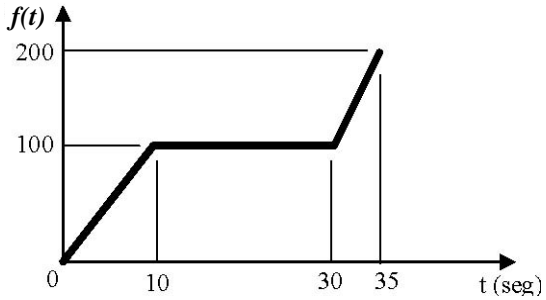

Figura 8.9.- Señal de entrada de forma especial.

Una manera de introducir una señal como la de la figura 8.9 en Simulink es generarla en una variable de Matlab y luego mediante el bloque '**From Workspace**' de la categoría **Sources** introducirla al diagrama de simulación de Simulink. En la figura 8.10 se muestra el aspecto que debe tener el diagrama de simulación de la práctica una vez sustituido el bloque escalón por el bloque '**From workspace**'

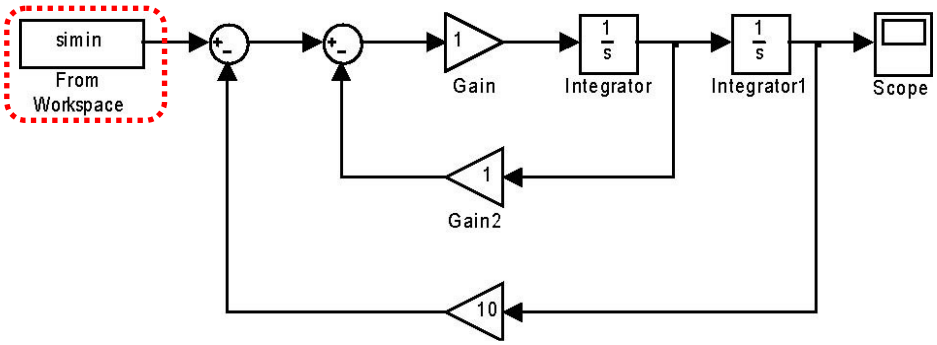

Figura 8.10.- Sistema Masa-resorte-amortiguador con entrada obtenida de un bloque 'From workspace'

Una vez modificado el esquema Simulink, haremos click sobre el elemento **'From Workspace'** para acceder a sus parámetros de configuración. Aparecerá una ventana como la que se muestra en la figura 8.11.

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 se denomina **`'simin'`**. Cambiaremos este nombre por otro más ilustrativo:

Dado que la entrada, en el sistema considerado, es la fuerza aplicada a la masa del sistema, llamaremos a esta variable **`'fuerza'`**. Esta variable debe ser un arreglo de dos columnas:

- la primera columna debe contener los valores de los instantes de tiempo
- la segunda columna debe contener los valores de la señal de fuerza.

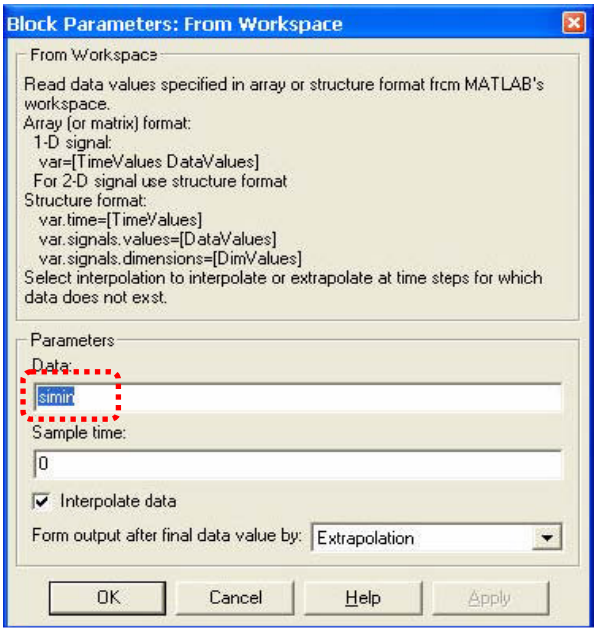

Figura 8.11.- Ventana de configuración del bloque 'From Workspace'

De acuerdo a la figura 8.9, la señal de fuerza se puede definir de acuerdo a las ecuaciones de los tres segmentos de recta que la conforman en cada uno de los tres sub intervalos como sigue

$$f(t)=\begin{cases} 10t & 0\le t\le 10 \\ 100 & 10< t\le 30 \\ 20t-500 & 30< t\le 35 \end{cases} \tag{8.2}$$

Por lo tanto, en Matlab podemos definir $f(t)$ aprovechando que las desigualdades tienen valores lógicos 0 ó 1, como sigue

```
>> t=0:0.1:35;
>> f=10*t.*(0<=t & t<=10) + 100*(10<t & t<=30) + (20*t-500).*(30<t & t<=35);
>> plot(t,f); grid on;
```

Que produce el resultado mostrado en la figura 8.12

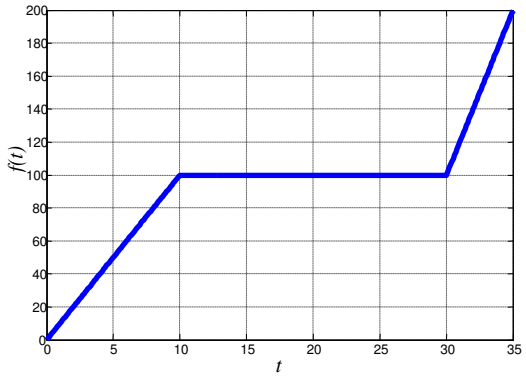

Figura 8.12.- Señal de fuerza generada mediante la expresión (8.2).

Una vez que se ha comprobado visualmente en la figura 8.12 que las variables **t** y **f** contienen

valores correctos, generaremos a partir de ellas la variable fuerza necesaria para el bloque '**From Workspace' como sigue**

```
>> fuerza=[t(:) f(:)];
```

Ahora es posible lanzar la simulación. El tiempo de simulación se deberá fijar exactamente en 35 segundos (en la opción **Configuration parameters** del menú **Simulation**). El resultado de la simulación se muestra en la figura 8.13. Obsérvese cómo la salida del sistema (posición) da un seguimiento aproximado a los valores de la entrada (fuerza aplicada) con pequeñas oscilaciones.

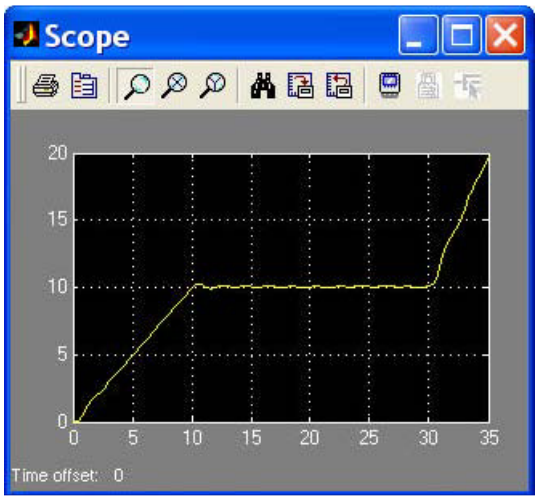

Figura 8.13.- Resultado de la simulación del diagrama de la figura 8.7 con la variable fuerza definida en Matlab.

Generación de funciones del tiempo en Simulink.

Una alternativa para introducir una señal especial como la del ejemplo anterior sin recurrir a una variable importada desde Matlab, es generarla directamente en Simulink.

Para generar una señal (función del tiempo) desde simulink, primeramente se debe generar la 'señal de tiempo', es decir, una señal cuyos valores en cada instante coincidan con el tiempo siimulado, es decir, $f(t)=t$ lo cual corresponde a una señal rampa unitaria (pendiente uno, inicio en cero) como se muestra en la figura 8.14

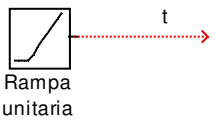

Figura 8.14.- Generación de la señal tiempo.

Una vez que se ha generado la señal de tiempo, se puede producir cualquier función del tiempo usando bloques de simulink para hacer la operación deseada, o de manera más sencilla mediante un bloque de **función definida por el usuario 'Fcn'**, por ejemplo, para generar la misma señal del ejemplo anterior se podría hacer con el diagrama de la figura 8.15. Obsérvese que dentro del

bloque 'Fcn' se escribe la misma expresión que se usó en el código de Matlab para obtener el vector `f`, salvo que ahora se reemplaza `t` por `u(1)` y se usan signos dobles `&&` en lugar de signos sencillos `&`.

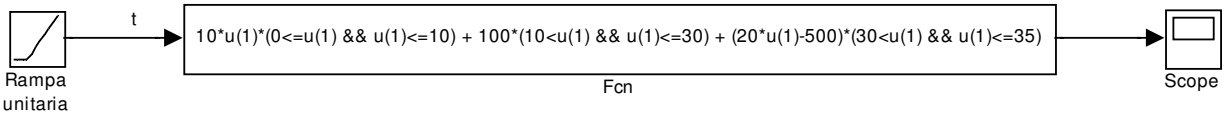

Figura 8.15.- Generación de la señal del ejemplo anterior mediante Simulink.

Uso de parámetros simbólicos en bloques de Simulink.

En los ejemplos de simulación anteriores se han utilizado valores simbólicos de constantes que aparecen en el diagrama de simulación de Simulink y cuyo valor es asignado desde Matlab. Por ejemplo, en el diagrama de simulación de la figura 8.2 las constantes K=10, M=1 y B=1 fueron asignadas desde Matlab previo a la ejecución de la simulación con el comando **`sim`** de Matlab.

Podemos utilizar constantes simbólicas en los bloques de Simulink sin recurrir al comando **`sim`** de Matlab insertando código de inicialización mediante 'Callbacks'. Para acceder a esta opción hay que seleccionar la opción 'Model Properties' en el menú 'File' del diagrama de simulación como se muestra en la figura 8.16.

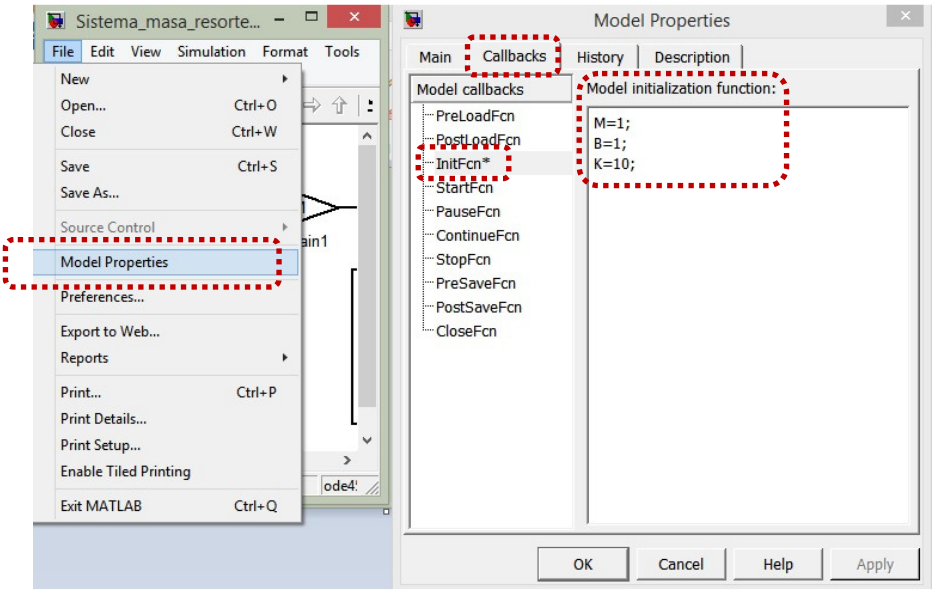

Figura 8.16.- Inserción de código de inicialización mediante la ventana de Propiedades del Sistema.

Al seleccionar la opción 'Model properties' se despliega la ventana de propiedades del sistema, en la cual hay que seleccionar la pestaña 'Callbacks' y dar click en la opción 'InitFcn' como se muestra en la parte derecha de la figura 8.16. En este punto se puede escribir el código de inicialización de las constantes en el área de escritura 'Model initialization function:'. Una vez hecho esto el diagrama de simulación se puede ejecutar sin recurrir a Matlab.

Creación de Subsistemas.

Cuando un diagrama de simulación es my grande se vuelve impráctico conservar todos los bloques a la vista del usuario. En este caso conviene definir conjuntos de bloques como parte de un subsistema y esconderlos a la vista del usuario, mostrando solamente la entrad ay la salida de dicho subsistema. Para Crear un subsistema en simulink se seleccionan con el cursor todos los bloques que se encerrarán en un bloque común y se accede a la opción 'Create Subsystem' del menú 'Edit' (Atajo en el teclado: [ctrl-G]).

Por ejemplo, en el diagrama de la figura 8.2 sleccionamos con el cursor todos los elementos, excepto el bloque step y el bloque scope y presionamos [ctrl-G], entonces el diagrama se convierte en el mostrado en la figura 8.17. El cual aún contiene la información de los bloques originales. Si deseamos ver los bloques interiores del subsistema creado le damos doble click al bloque del susbsistema y se muestra el diagrama de la figura 8.18.

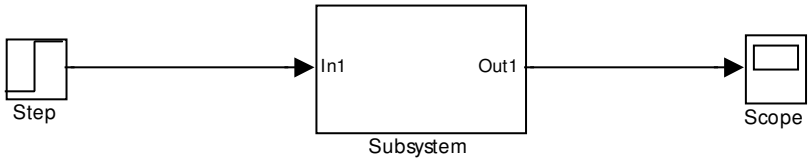

Figura 8.17 Creación de subsistema seleccionando bloques de la figura 8.2 y presionando [ctrl-G].

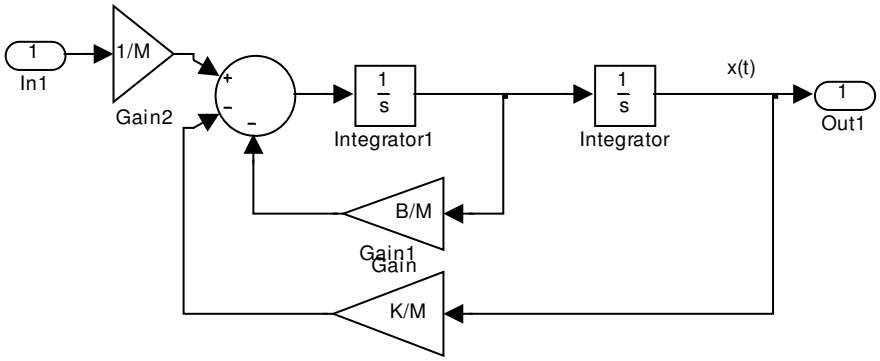

Figura 8.18.- Subsistema visualizado al dar doble click al bloque 'Subsystem' de la figura 8.17.

Sistemas Lineales en Simulink.

Aunque cualquier sistema, ya sea Lineal o No-lineal puede simularse obteniendo su diagrama de simulación con integradores a partir de sus ecuaciones de diferenciales, Simulink proporciona alternativas más sencillas para el caso de sistemas lineales que nos evitan la obtención del diagrama de simulación con integradores, mostradas en la figura 8.19.

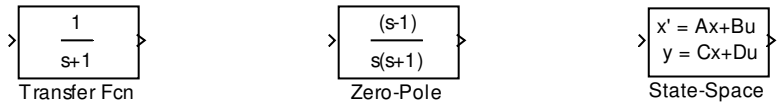

Figura 8.19.- Bloques 'Transfer Fcn', Zero-Pole' y 'State-Space'

- El bloque **'Transfer 'Fcn'** permite introducir un sistema lineal mediante su Función de

Transferencia en forma de división de polinomios en potencias descendentes de 's'.

- El bloque **'Zero-Pole'** permite introducir un sistema lineal mediante su Función de Transferencia en la forma de división de polinomios factorizados.
- El bloque **'State Space'** permite introducir un sistema lineal especificando las matrices A,B.C y D de su modelo en variables de estado.

Ejercicio.

1) Reemplazar el diagrama de simulación del sistema masa-resorte-amortiguador por un solo bloque de función de transferencia. Recordar que la función de transferencia de ese sistema es

$$\frac{X(s)}{F(s)} = \frac{1}{Ms^2 + Bs + K} \tag{8.3}$$

2) Considerando M=1, B=1, K=10, Obtener la respuesta del sistema ante un escalón de amplitud 5 y comparar la respuesta con la figura 8.3

3) Obtener la respuesta del sistema ante la entrada mostrada en la figura 8.18

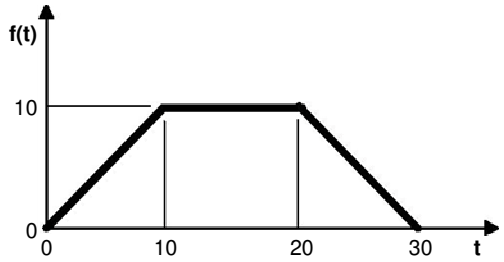

Figura 8.18.- Entrada de prueba del ejercicio.

4) Exportar la respuesta del inciso anterior a una variable de Matlab denominada `'Xt'` y Graficarla contra el tiempo mediante el comando `plot` junto con la gráfica de la entrada.

Desarrollo de la Práctica.

1. Probar todos los ejemplos propuestos por el profesor conforme los va explicando.
2. Realizar todos los ejercicios propuestos.

Reportar:

1) El diagrama de simulación elaborado para el ejercicio final de la práctica usando el bloque de función de transferencia y las gráficas obtenidas en cada inciso.