TEMA 5

Simulink

1. Introducción

SIMULINK es una *toolbox* especial de MATLAB que sirve para simular el comportamiento de los sistemas dinámicos. Puede simular sistemas lineales y no lineales, modelos en tiempo continuo y tiempo discreto y sistemas híbridos de todos los anteriores. Es un entorno gráfico en el cual el modelo a simular se construye clicando y arrastrando los diferentes bloques que lo constituyen. Los modelos SIMULINK se guardan en ficheros con extensión `*.mdl`.

Con las nuevas versiones, SIMULINK ha ido ampliando sus librerías de bloques (*blocksets*) y capacidades. En concreto, destaca el paquete STATEFLOW, que permite la simulación de máquinas de estados.

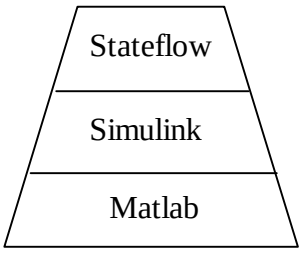


Fig. 1. Jerarquía de Matlab, Simulink, Stateflow

Otras *blocksets* de interés son, por ejemplo, las de comunicaciones (*Communications Blockset*, *CDMA Reference Blockset*, *RF Blockset*) que incluyen bloques que simulan estaciones de telefonía móvil o dispositivos tales como los PLLs; las de aplicaciones específicas (*Aerospace Blockset*, *Signal Processing Blockset*, *Video and Image Processing Blockset*); y las de soporte (*Gauges Blockset*). Hay muchas demos y efectos (ver, por ejemplo, las demos de *SimMechanics* o *Virtual Reality Toolbox* `>>mech_conveyor_vr`, `>>mech_airbag_vr` …).

Además algunas *toolboxes* de MATLAB incorporan también bloques de SIMULINK. Es el caso, por ejemplo, de la *Control Systems Toolbox*, *Neural Network Toolbox*, *Fuzzy Logic Toolbox*, *System Identification Toolbox*,… Finalmente, también existen librerías de bloques que permiten interactuar con tarjetas de adquisición de dados y DSPs: *Real-Time Workshop, Embedded Targets for Motorola and TI*, *xPC Target*.

Teclear `>>ver` en la ventana de comandos de MATLAB para ver qué versión de SIMULINK y qué *blocksets* están instaladas.

2. Simulink

2.1 Cómo iniciar Simulink

Para abrir la librería de bloques SIMULINK puede hacerse escribiendo `>>simulink` en la ventana de comandos de MATLAB, o bien clicando en el icono  de la barra de menús de MATLAB.

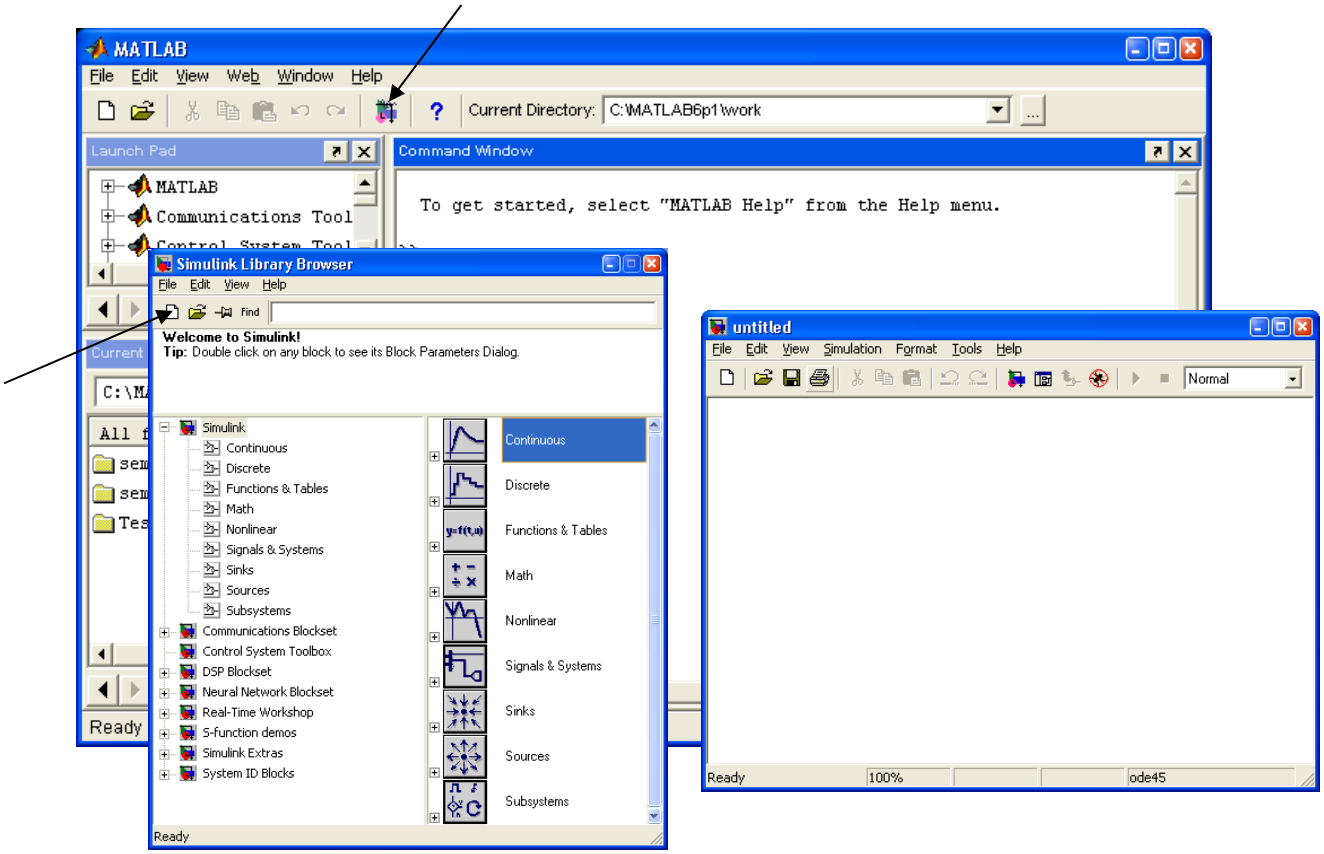


Fig. 2. Inicio de Simulink (en Matlab v6.1)

Si ya tenemos algún modelo SIMULINK creado, `modelo1.mdl`, basta hacer doble clic en su nombre para que primero se abra MATLAB y a continuación se abra la ventana `modelo1` que contiene a dicho modelo. O bien, si se tiene el MATLAB abierto, basta con teclear `>>modelo1` en la ventana de comandos.

Si vamos a crear un modelo nuevo hay que seleccionar `File` → `New` → `Model` o bien clicar en el icono  de la librería de SIMULINK.

2.2 Construcción de modelos

Ventana de modelo: Cada modelo (o submodelo) se construye en una ventana diferente. Por ello, para construir un nuevo modelo hay que abrir una nueva ventana de modelo `untitled` (ver Fig. 2) A partir de ahí, se trata de arrastrar los bloques que compondrán el modelo desde la librería de SIMULINK a dicha ventana. Antes de empezar a trabajar con SIMULINK, se sugiere echar un vistazo a las opciones de la barra de menús y la barra de herramientas de la ventana de modelo.

Interconectar bloques: Las interconexiones entre bloques se realizan arrastrando el ratón entre los puertos de entrada y salida de dichos bloques. También es posible seleccionar un bloque y, manteniendo la tecla `<ctrl>` presionada, clicar en el otro bloque. Se puede poner texto en cualquier sitio (haciendo doble clic en el sitio deseado), se pueden cambiar los nombres de los bloques y se pueden usar distintos colores (`Format` → `Foreground Color`). También se pueden rotar bloques (`Format` → `Flip block, Rotate block`), etc.

Ejemplo 1. Construcción de un modelo sencillo

La siguiente figura muestra la construcción de un modelo sencillo que nos permite la simulación de la respuesta indicial (al escalón) del sistema en tiempo continuo de segundo orden $H(s) = \dfrac{1}{s^2 + 0.5s + 1}$.

La señal de excitación `Step` la hemos arrastrado desde la librería `Sources`, el visualizador `Scope` desde la librería `Sinks` y el sistema en tiempo continuo `Transfer Fcn` desde la librería `Continuous`.

Puesto que los parámetros de nuestro sistema continuo son diferentes a los que vienen por defecto en la librería de SIMULINK, hemos hecho doble clic en el bloque `Transfer Fcn` de nuestra ventana para editar dicho bloque con nuestro numerador y denominador particulares.

Para iniciar la ejecución del modelo basta con clicar en el icono  o seleccionar `Simulation` → `Start`. Para cambiar los parámetros de simulación (tiempo final, muestreo, etc.) seleccionar `Simulation` → `Configuration Parameters…` (o bien `Simulation` → `Simulation Parameters…` según la versión).

Para ver el resultado de la simulación clicar sobre el bloque `Scope` a fin de abrirlo y, una vez en él, clicar en el icono  para realizar un autoescalado de los ejes.

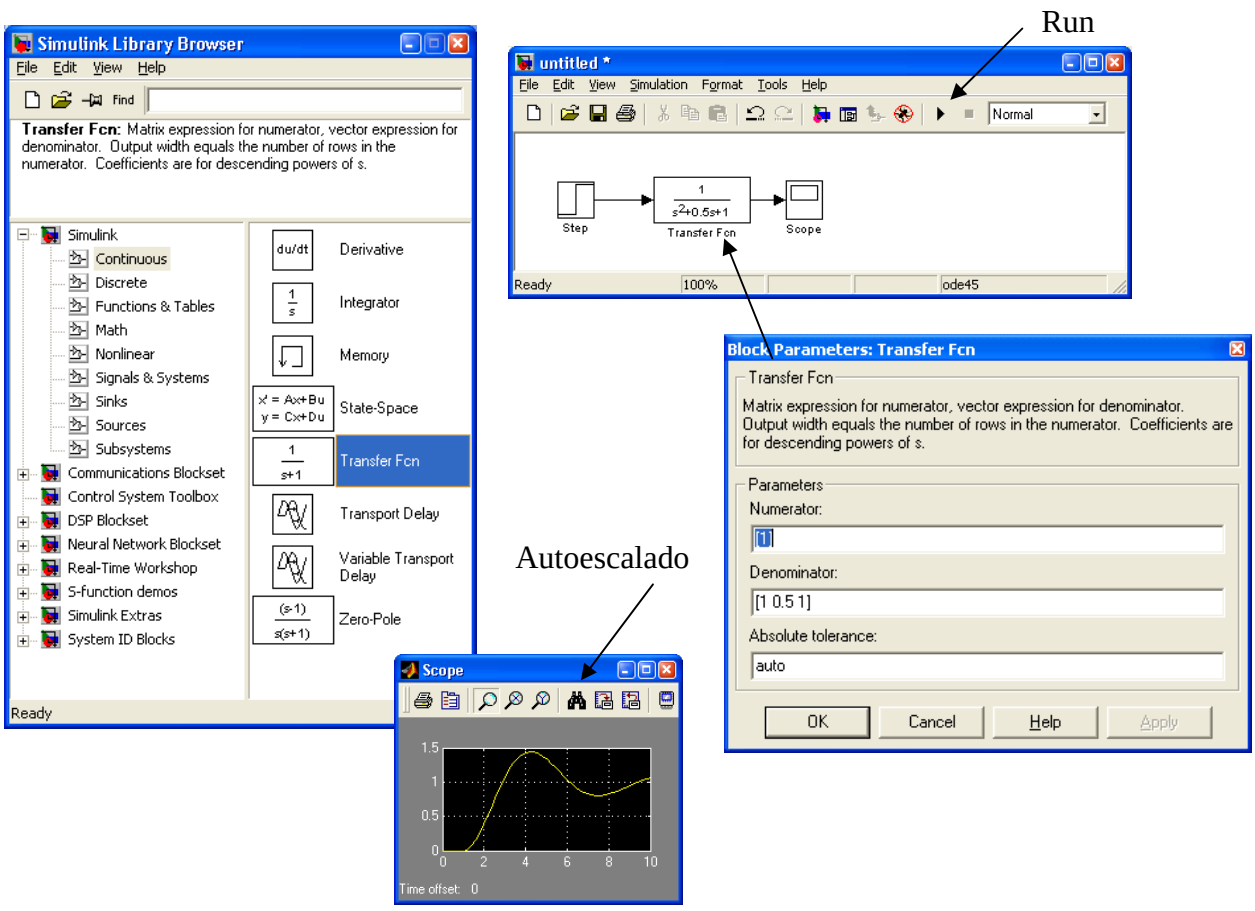


Fig. 3. Construcción y simulación de modelos

2.3 Parámetros de simulación y bloque Scope

Parámetros de simulación: En la v6, los parámetros de la simulación se pueden modificar desde la barra de menú con `Simulation` → `Simulation parameters...`

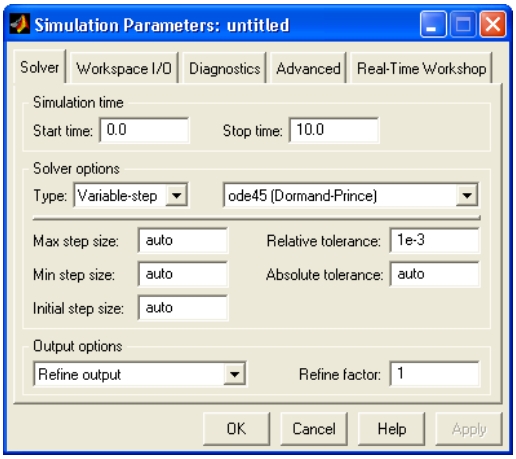


Fig. 4. Parámetros de la simulación (matlab v6)

En la v7, los parámetros de la simulación se pueden modificar desde la barra de menú con `Simulation → Configuration parameters...`

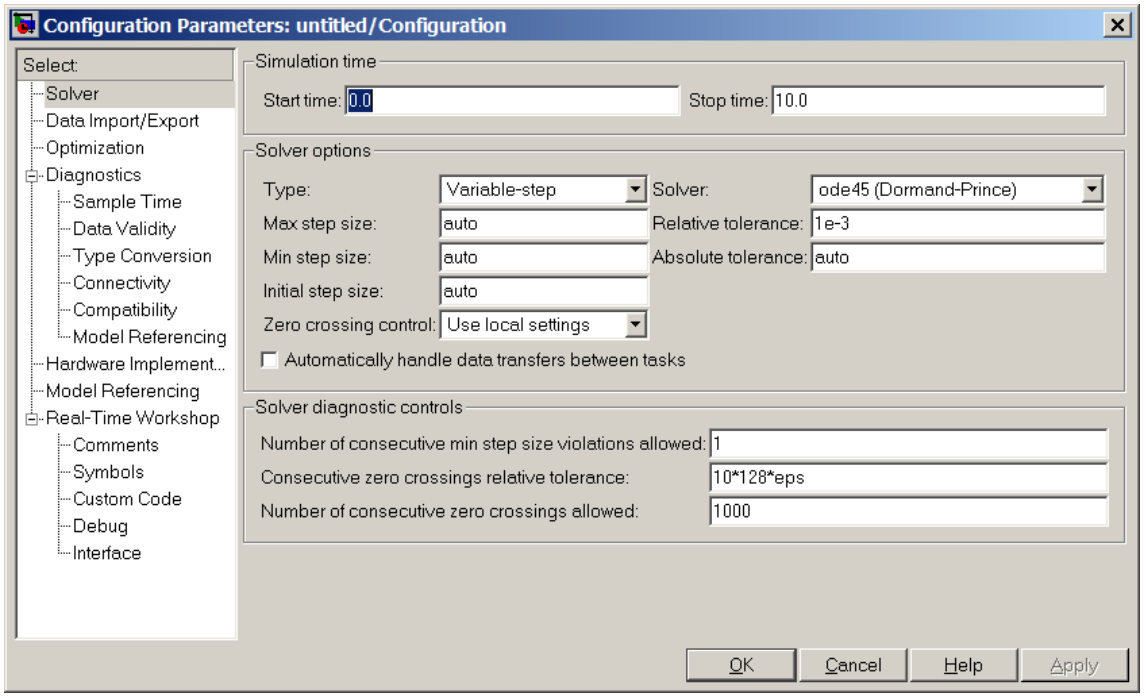


Fig. 5. Parámetros de la simulación (matlab v7)

Run: Para iniciar la simulación basta con clicar en el icono  o seleccionar `Simulation → Start` en la ventana del modelo (ver Ejemplo 1).

Función sim: También es posible ejecutar un modelo Simulink desde la ventana de comandos o desde un fichero M. La instrucción es `sim`. En nuestro ejemplo:

```
>>sim('untitled')

Warning: Using a default value of 0.2 for maximum step size.  The
simulation step size will be equal to or less than this value.
You can disable this diagnostic by setting 'Automatic solver
parameter selection' diagnostic to 'none' in the Diagnostics page
of the configuration parameters dialog.
```

Interacción con el workspace de MATLAB: Desde un modelo SIMULINK, es posible usar funciones `*.m`, `*.mat` y variables del *workspace* de MATLAB. Los bloques relacionados son los bloques *From File*, *From Workspace* (en la librería *Sources*), *To File*, *To WorkSpace* (en la librería *Sinks*) y *MATLAB Fcn* (en la librería *Functions & Tables*).

También es posible poner nombres de variables del *workspace* dentro del modelo Simulink. Por ejemplo:

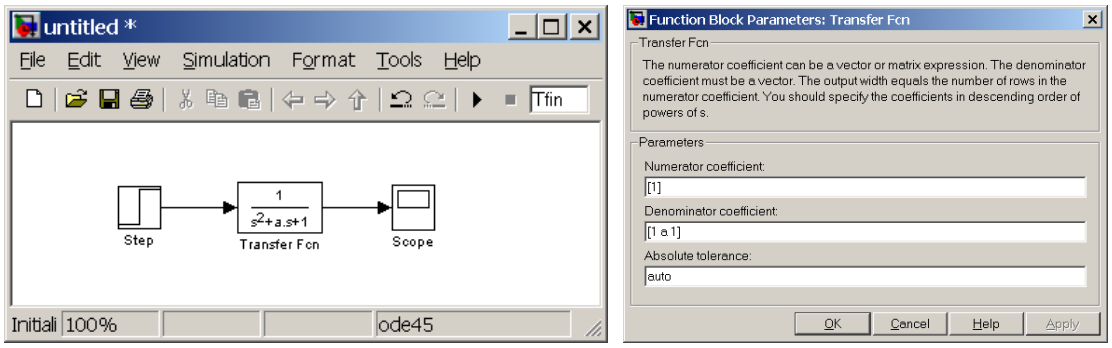


Fig. 6. Modelo Simulink con variables del *workspace* (`a` y `Tfin`)

Antes de ejecutar el modelo, estas variables deben tener asignado un valor:

```
>>a=0.5;Tfin=10;
>>sim('untitled')
```

Model Explorer: Esta utilidad se abre al clicar en el icono  de la ventana de modelo. En concreto, el objeto `Base Workspace` contiene las variables a las que pueden acceder todos los modelos Simulink.

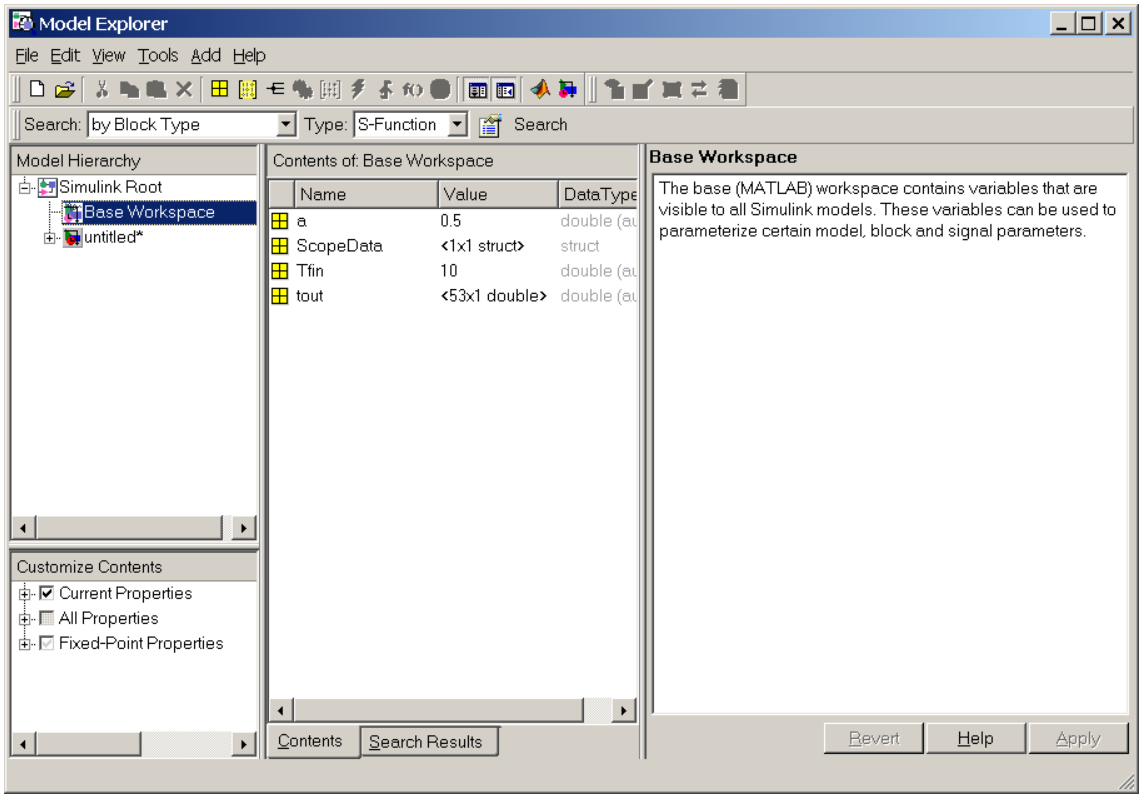


Fig. 7. Model Explorer

Bloque Scope: Para visualizar el resultado de una simulación hay que clicar en el bloque `Scope` a fin de abrir la ventana de visualización. El autoescalado se consigue

clicando en . Para hacer un *zoom*, clicar en . Los iconos y permiten hacer un zoom a lo largo del eje *x* o del eje *y* respectivamente.

Si queremos poner un título, se puede clicar con el botón derecho sobre la representación a fin de abrir un menú de contexto y, en él, seleccionar la opción `Axes properties…`:

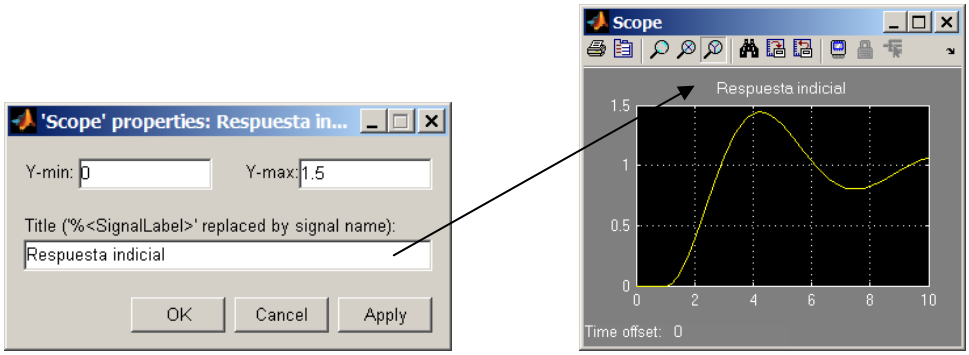


Fig. 8. Títulos en el Scope

Los parámetros del `Scope` se pueden modificar clicando en . Por defecto, el `Scope` representa sólo las últimas 5000 muestras. Si nuestra simulación tiene más de 5000 muestras y las queremos ver todas hay que deseleccionar la opción `Limit data points to last:` de la ficha `Data history` (ver Fig. 9).

También es posible guardar las variables de la simulación en el *workspace* desde la ventana del *Scope*:

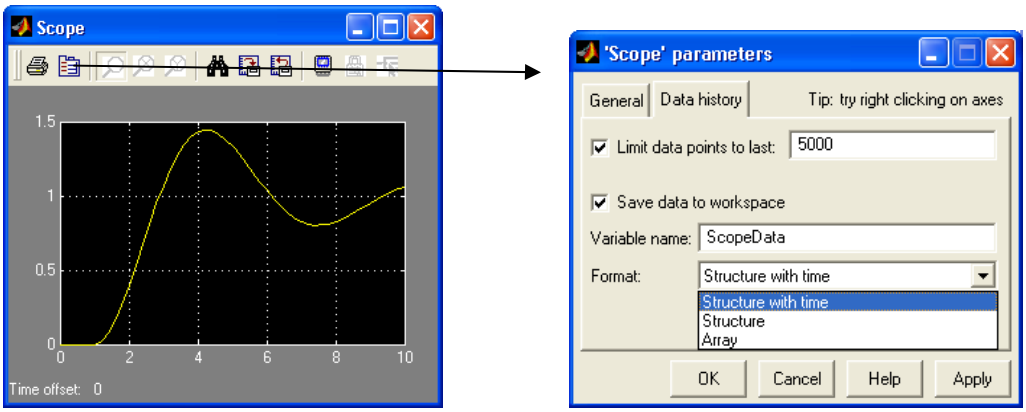


Fig. 9. Cómo pasar datos al *workspace* desde el bloque Scope

La opción `Save data to workspace` permite guardar en el *workspace* los valores representados en el `Scope`. Hay tres formatos posibles: `Structure with time`, `Structure` y `Array`.

Formato Structure with time: Una vez realizada la simulación en Simulink, si el formato elegido es *Structure with Time*, podemos comprobar que en el *workspace* de Matlab aparecen dos nuevas variables

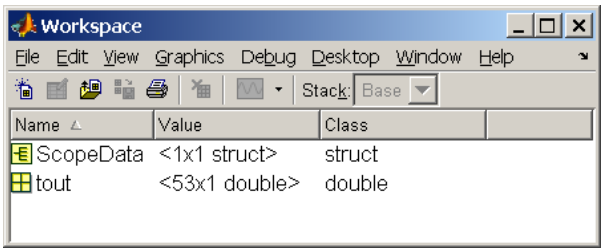


Fig. 10. Formato *Struct with time* en el *workspace* de Matlab

La variable `ScopeData` es una estructura (`struct array`). Para ver su contenido, basta con hacer:

```
>> ScopeData
ScopeData =
         time: [53x1 double]
      signals: [1x1 struct]
    blockName: 'untitled/Scope'
```

`'time'`, `'signals'` y `'blockName'` son los tres campos (*fields*) que componen la estructura (*struct*) `ScopeData`. Vemos que el campo `'signals'` es, a su vez, otra estructura.

Funciones fieldnames, isfield, getfield: Es posible consultar qué campos tiene `ScopeData` con ayuda de `fieldnames` (esta función guarda el resultado en un *cell array*)

```
>> nombres=fieldnames(ScopeData)
nombres =
    'time'
    'signals'
    'blockName'
```

También se puede ver si un determinado campo forma parte de una estructura: `isfield(ScopeData, 'time')`.

Hay dos maneras de acceder al contenido de los campos. Una es mediante "puntos" y la otra mediante la función `getfield`:

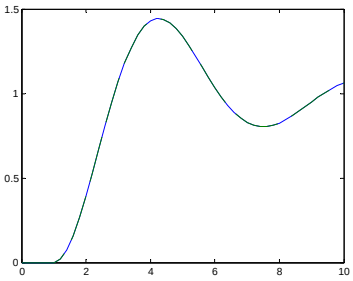
```
>> ScopeData.signals
ans =
        values: [53x1 double]
    dimensions: 1
         label: ''
         title: 'Respuesta indicial'
     plotStyle: 0

>> ScopeData.signals.dimensions
ans =
     1

>> y=getfield(ScopeData,'signals')
y =
        values: [53x1 double]
```

```
      dimensions: 1
           label: ''
           title: 'Respuesta indicial'
       plotStyle: 0
```

Así, al representar los datos, podemos ver que las instrucciones son equivalentes:




```
y1=getfield(ScopeData,'signals','values');
y2=ScopeData.signals.values;
plot(tout,y1,tout,y2,'--')
```

Formato Array: En el caso de haber escogido el formato *Array*, es posible acceder directamente a dichas variables:

```
>> size(ScopeData)
ans =
    53     2
>> size(tout)
ans =
    53     1
```

En concreto, las dos siguientes instrucciones hacen lo mismo:

```
>> plot(tout,ScopeData(:,2))
>> plot(ScopeData(:,1),ScopeData(:,2))
```

Otra forma de analizar las variables es a través de la ventana de *workspace* y del editor de variables:

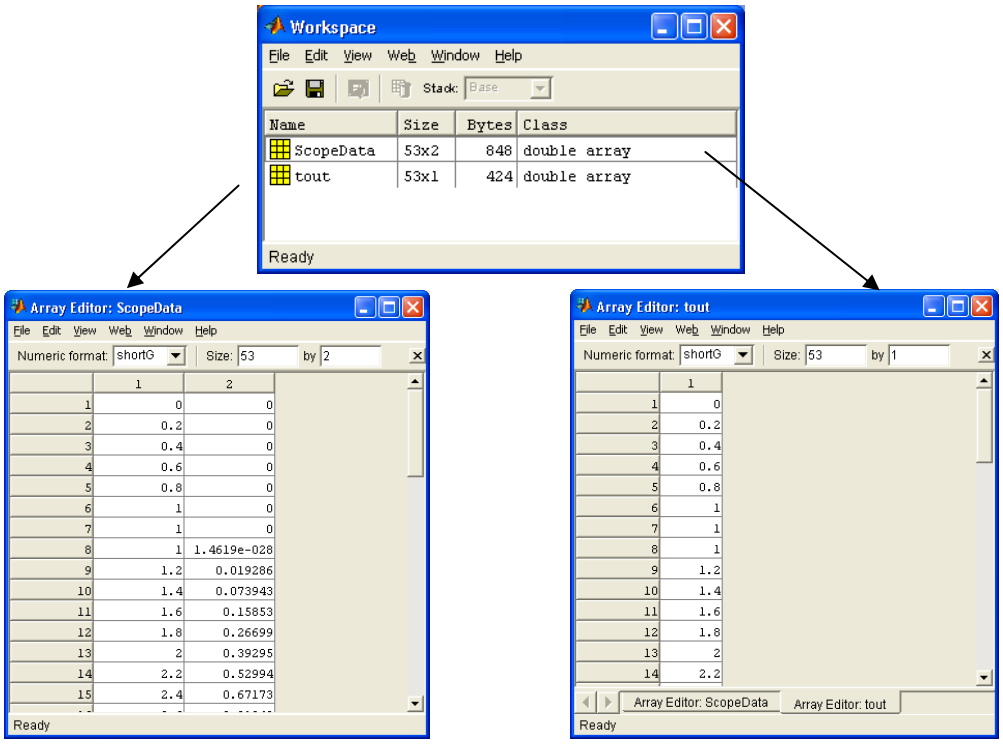


Fig. 11. Acceso a los resultados de la simulación (formato array)

Formato Structure: Si en el `Scope` se selecciona el formato *Structure*, en el *Workspace* aparecen las variables `ScopeData` y `tout`, pero la diferencia con el formato anterior (*Structure with time*) es que el campo `time` de `ScopeData` ahora está vacío.

2.4 Subsistemas y máscaras

Creación de subsistemas: Es posible agrupar diversos bloques para crear subsistemas. Para ello basta con seleccionar con el ratón todos los bloques de interés. (Nota: el bloque `Gain` lo hemos arrastrado desde la librería `Math Operations` o `Math`, según la versión)

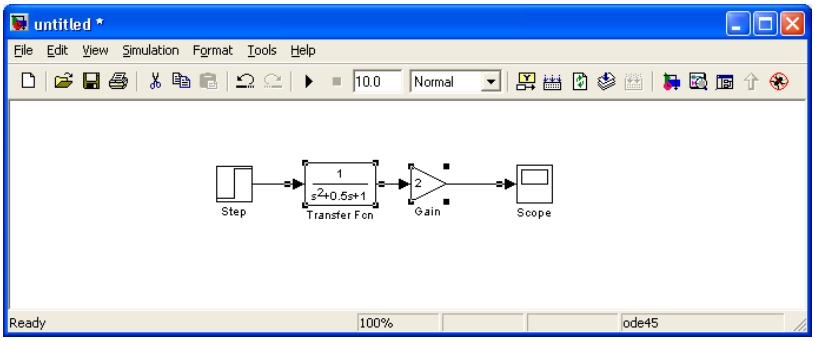


Fig. 12. Selección de bloques…

Una vez seleccionados todos los bloques que formarán el subsistema, ir a `Edit` → `Create Subsystem`:

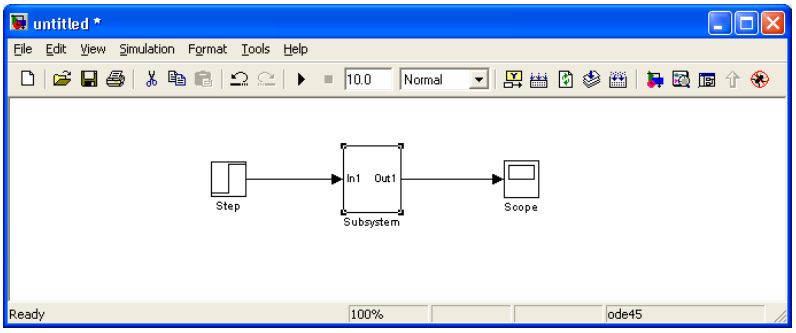


Fig. 13. …para crear un subsistema

Notar que automáticamente se crean, rotulan y numeran los puertos de entrada y salida del submodelo y que, al hacer doble clic sobre el bloque subsistema, se abre una nueva ventana de modelo que lo contiene:

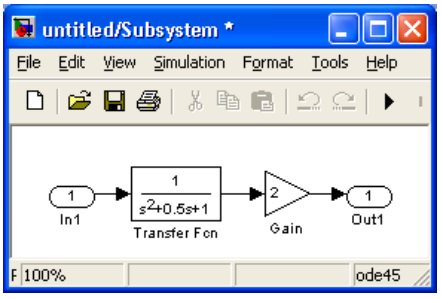


Fig. 14. Ventana del subsistema

Máscaras: Una vez creado un subsistema es posible personalizarlo. El editor de máscaras que se abre al hacer `Edit` → `Mask Subsystem…` permite personalizar el icono del subsistema, poner unos comentarios de ayuda e introducir, en su caso (normalmente en Funciones S), los valores de los parámetros necesarios.

Por ejemplo, en la v7, poner como `Drawing commands` el siguiente comando: `image(imread('b747.jpg'))`, tiene como resultado:

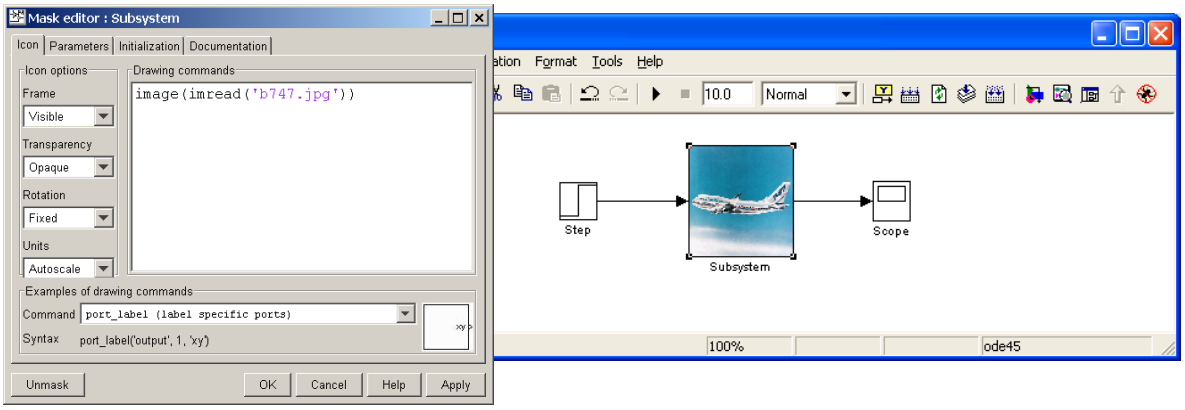


Fig. 15. Máscaras

Una vez creada la máscara, ésta se puede modificar haciendo `Edit` → `Edit Mask...`.

2.5 Funciones S

Las funciones S son bloques SIMULINK programables por el usuario y que sirven para describir modelos dinámicos complicados (parámetros variantes con el tiempo, sistemas no lineales, con diferentes periodos de muestreo, etc.).

La comunicación con el exterior (*hardware-in-the-loop*) también se realiza con funciones S. En este último caso las funciones S están escritas en C (contienen los comandos para comunicación con las tarjetas de adquisició de datos y DSPs) y han de ser compiladas y lincadas con la utilidad `mex` de Matlab.

Funciones S para sistemas dinámicos: Para la simulación de sistemas dinámicos, existen unos ficheros plantilla (sistemas continuos `csfunc.m`, sistemas discretos `dsfunc.m` y caso general `sfuntmpl.m`) cuyo propósito es ser copiados en el directorio de trabajo y particularizados con las instrucciones y parámetros del usuario.

```
>> which csfunc.m
C:\Archivos de programa\MATLAB704\toolbox\simulink\blocks\csfunc.m
```

Estos ficheros están estructurados en una serie de rutinas básicas que SIMULINK ejecuta de manera secuencial:

- Inicialización (`mdlInitializeSizes`), `flag=0`,
- cálculo de la derivada en sistemas continuos (`mdlDerivatives`), `flag=1`,
- cálculo de la diferencia finita en sistemas discretos (`mdlUpdate`), `flag=2`,
- cálculo de la salida del sistema (`mdlOutputs`), `flag=3`, y
- tareas de terminación (`mdlTerminate`), `flag=9`.

Notar que esta estructura lo que hace es simular (muestra a muestra) las ecuaciones de estado del sistema. Primero, simula la ecuación de estado propiamente dicha (`mdlDerivatives`), $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$, y, después, la ecuación de salida (`mdlOutputs`), $\mathbf{y} = \mathbf{Cx} + \mathbf{Du}$. En el caso de sistemas discretos en el tiempo la ecuación de estado es una ecuación en diferencias (`mdlUpdate`) es $\mathbf{x}[n+1] = \mathbf{Ax}[n] + \mathbf{Bu}[n]$.

Función csfunc: El contenido del modelo de muestra `csfunc.m` es el siguiente. Notar que se trata de un sistema de dos estados (la matriz **A** es 2×2), dos entradas (la matriz **B** tiene dos columnas) y dos salidas (la matriz **C** tiene dos filas).

```
>> type csfunc

function [sys,x0,str,ts] = csfunc(t,x,u,flag)
%CSFUNC An example M-file S-function for defining a continuous system.
%   Example M-file S-function implementing continuous equations:
%      x' = Ax + Bu
%      y  = Cx + Du
%
%   See sfuntmpl.m for a general S-function template.
%
%   See also SFUNTMPL.

%   Copyright 1990-2002 The MathWorks, Inc.
%   $Revision: 1.9 $

A=[-0.09   -0.01
    1          0];

B=[ 1   -7
    0   -2];

C=[ 0    2
    1   -5];

D=[-3    0
    1    0];

switch flag,
  %%%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D);
  %%%%%%%%%%%%%%%
  % Derivatives %
  %%%%%%%%%%%%%%%
  case 1,
    sys=mdlDerivatives(t,x,u,A,B,C,D);
  %%%%%%%%%%
  % Outputs %
  %%%%%%%%%%
  case 3,
    sys=mdlOutputs(t,x,u,A,B,C,D);
  %%%%%%%%%%%%%%%%%%%
  % Unhandled flags %
  %%%%%%%%%%%%%%%%%%%
  case { 2, 4, 9 },
    sys = [];
  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
% end csfunc

%
%=============================================================================
```

```
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================================
%
function [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D)
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = zeros(2,1);
str = [];
ts  = [0 0];
% end mdlInitializeSizes
%
%=============================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================================
%
function sys=mdlDerivatives(t,x,u,A,B,C,D)
sys = A*x + B*u;
% end mdlDerivatives
%
%=============================================================================
% mdlOutputs
% Return the block outputs.
%=============================================================================
%
function sys=mdlOutputs(t,x,u,A,B,C,D)
sys = C*x + D*u;
% end mdlOutputs
```

Para simular una función S hay que arrastrar el bloque correspondiente a nuestra ventana de modelos. El bloque `S-Function` está en la librería `Simulik` → `User-Defined Functions`. Hacer doble clic para editar el bloque y poner "S-function name" igual a "csfunc".

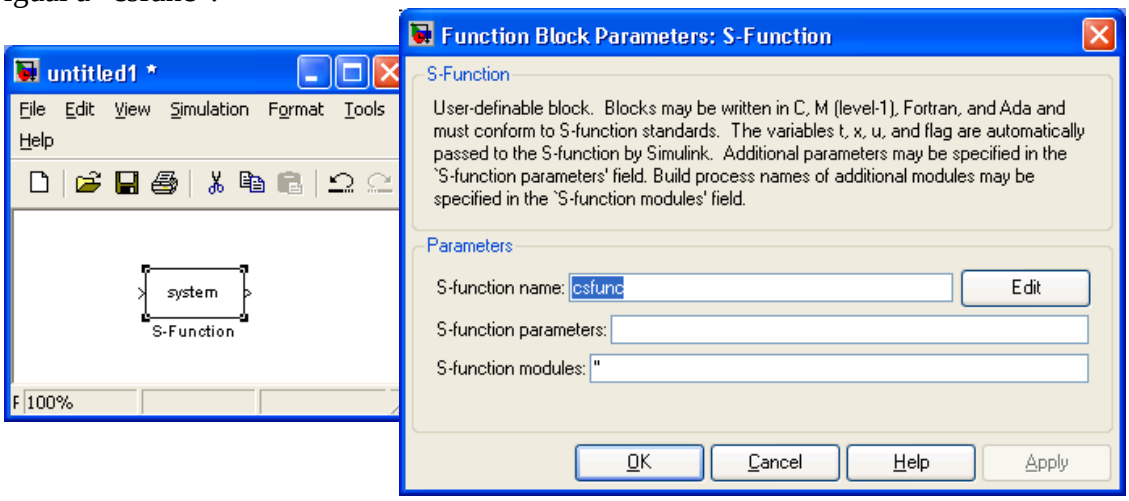


Fig. 16. Edición de una Función-S

Acabar de construir el modelo con una excitación (seno) y un `Scope`. Simular. Notar que da error puesto que estamos intentando excitar con una única señal un sistema que tiene dos entradas:

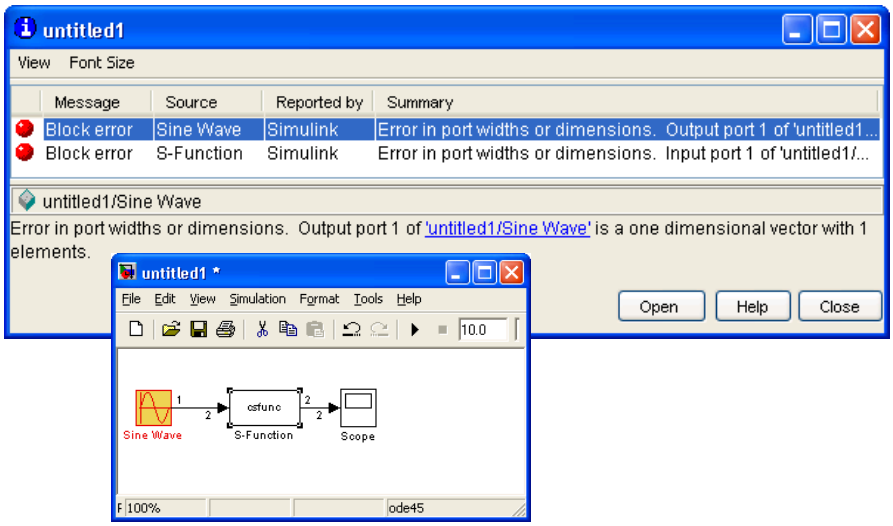


Fig. 17. Error debido a las dimensiones

Para solucionarlo, arrastrar un bloque `Demux` y volver a simular. El resultado ahora es:

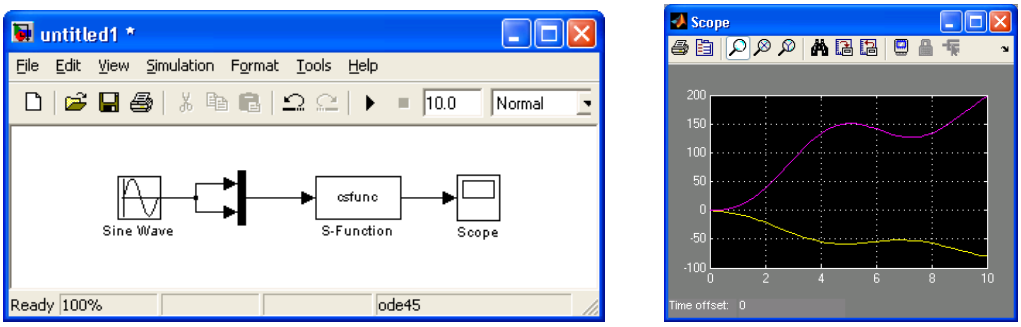


Fig. 18. Simulación de un sistema con 2 entradas y 2 salidas

Debugger: Simulink también tiene su propio *debugger*. Se abre al clicar en el icono . Si hubiéramos ejecutado el *debugger* antes de simular el modelo de la Fig. 17, ya hubiéramos visto que había un conflicto con las dimensiones:

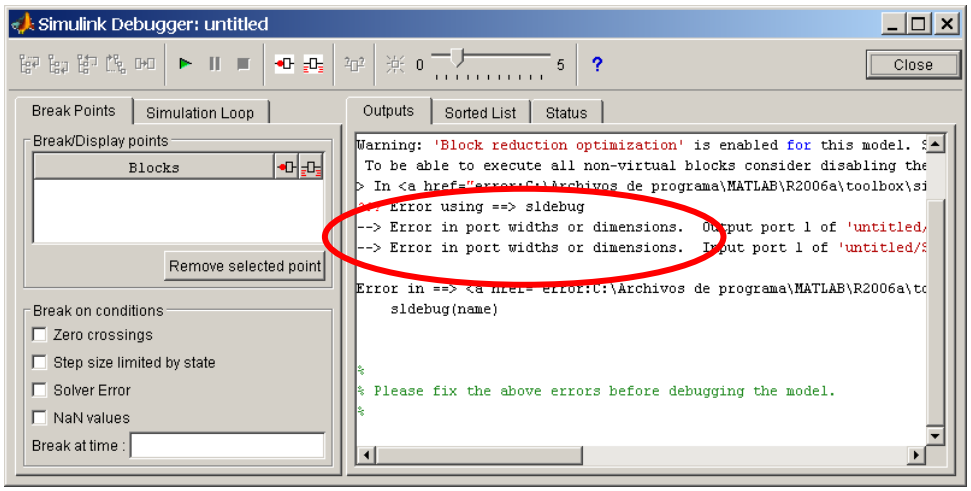


Fig. 19. *Debugger* de Simulink

3. Stateflow

Inicio: Para abrir el STATEFLOW hay que abrir la librería del SIMULINK y, desde allí, arrastrar el bloque `Chart` a una ventana de nuevo modelo, para, a continuación, abrir el editor de máquinas de estado haciendo doble clic en `Chart`.

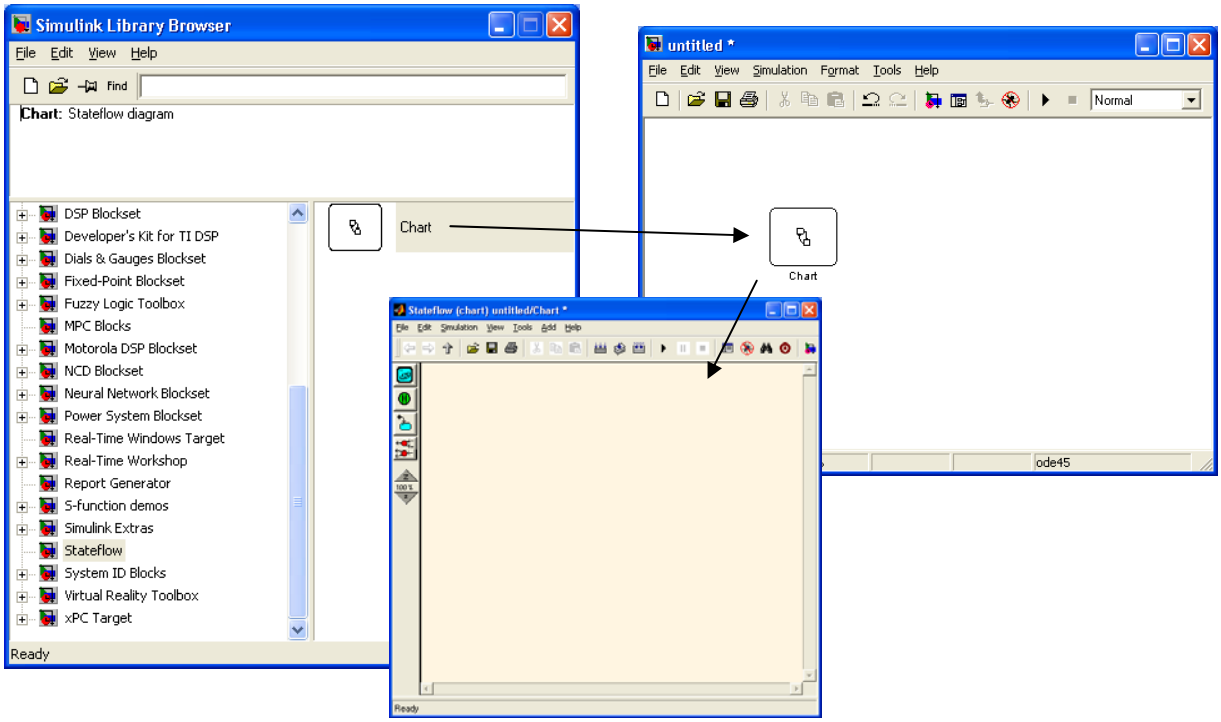


Fig. 20. Inicio de Stateflow

Componentes: Para crear la máquina de estados hay que arrastrar los componentes que se encuentran en la parte lateral de esta última ventana: *Estados*  , *Histórico*  , *Transición por defecto*  y *Punto de conexión/decisión*  . Las versiones más recientes incluyen *Tabla de verdad*  , *Función*  y *Función Matlab*  .

Las *transiciones* entre estados se crean arrastrando el ratón entre bloques y, para editarlas, se clica en ellas y se escribe sobre el interrogante que aparece.

Ejemplo 2. Encendido/apagado

Suponer que se quieren simular los dos estados de un dispositivo (encendido y apagado). El dispositivo se enciende cuando la variable externa "hora" es mayor que 12 y se apaga cuando es menor o igual que 12. De entrada está apagado.

En primer lugar se arrastran los dos estados y se edita su nombre: "apagado" y "encendido". Indicar que el estado "apagado" es el estado por defecto insertando sobre él una transición por defecto.

Las transiciones entre los dos estados se trazan con ayuda del ratón y, para editarlas, se clica sobre ellas y se escribe sobre el signo "?". Las condiciones booleanas, como es el caso del ejemplo, se introducen entre corchetes [ ], mientras que las acciones a realizar durante una transición, si las hubiera, se indican entre llaves { }.

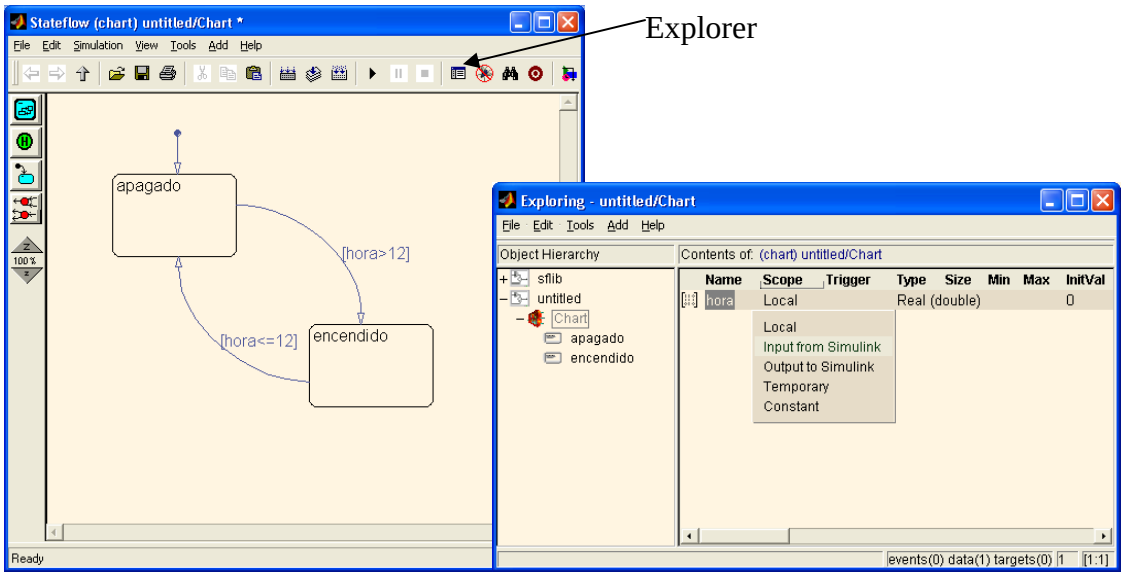


Fig. 21. Construcción de un modelo Stateflow (matlab v6)

Para indicar que la variable `hora` es externa (vendrá de SIMULINK), hay que abrir el explorador de modelos (clicar en  , o bien seleccionar `View → Model Explorer`). En el explorador, seleccionar `Add → Data` y ajustar los siguientes parámetros: `Name=hora`, `Scope=Input from Simulink` (o `Input`, según la versión). Así, en el modelo SIMULINK, en el bloque `Chart` aparecerá la entrada `hora`.

Notar que el `Model Explorer` tiene muchas opciones disponibles, como por ejemplo, el `Model Advisor`. Se recomienda echar un vistazo a todas las opciones del menú y la barra de herramientas.

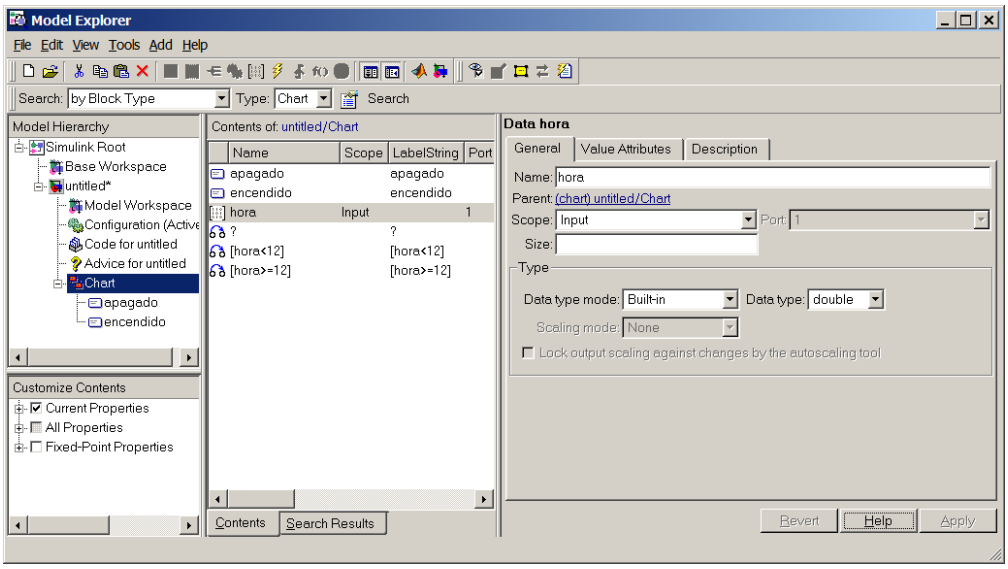


Fig. 22. Model explorer en la v7

Ahora basta con generar en Simulink una variable que oscile de 0 a 24 y excitar el bloque `Chart` con ella. En el ejemplo "hora" es una sinusoide de amplitud 12, frecuencia 0.5Hz y *offset* 12. El tiempo final de simulación es 10s y el tiempo de muestreo es fijo e igual a 0.001.

Para iniciar la simulación basta con clicar en el botón run (o seleccionar `Simulation` → `Start`) del modelo Simulink. En la pantalla de comandos de Matlab aparecerán una serie de mensajes relacionados con la creación de la máquina de estados.

A medida que la simulación se lleva a cabo, podemos ver cuál de los estados está activado cada vez.

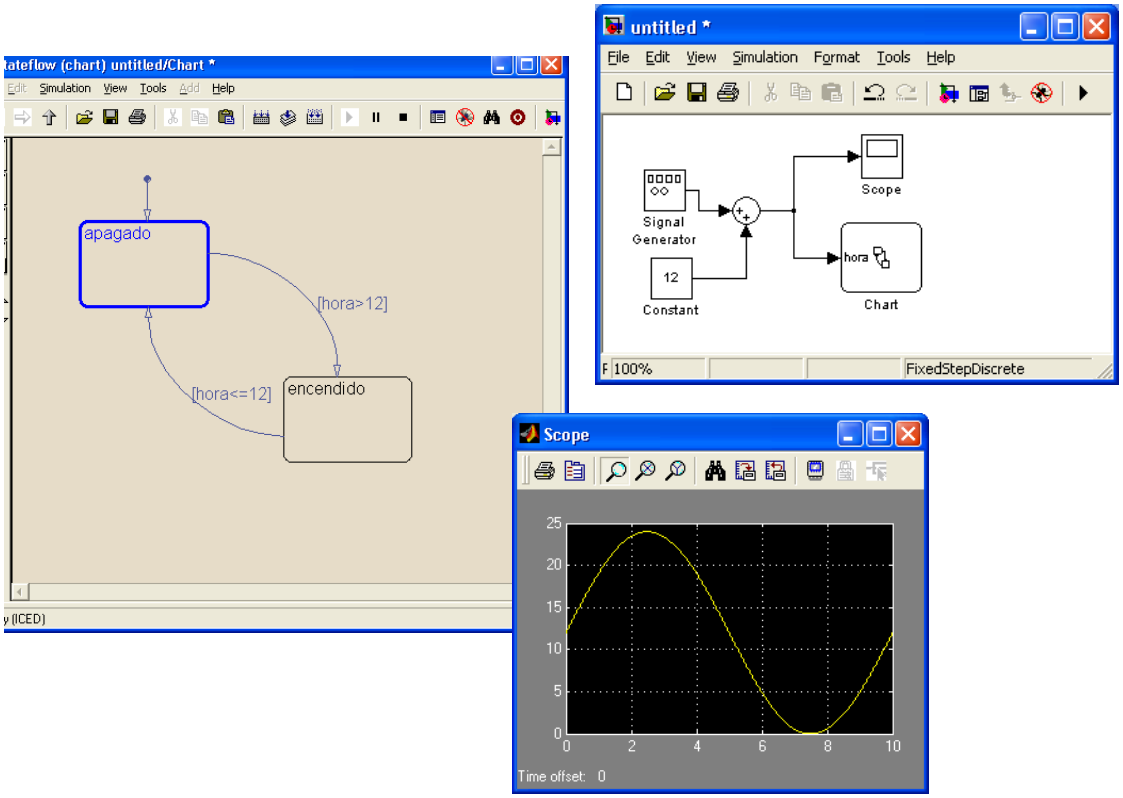


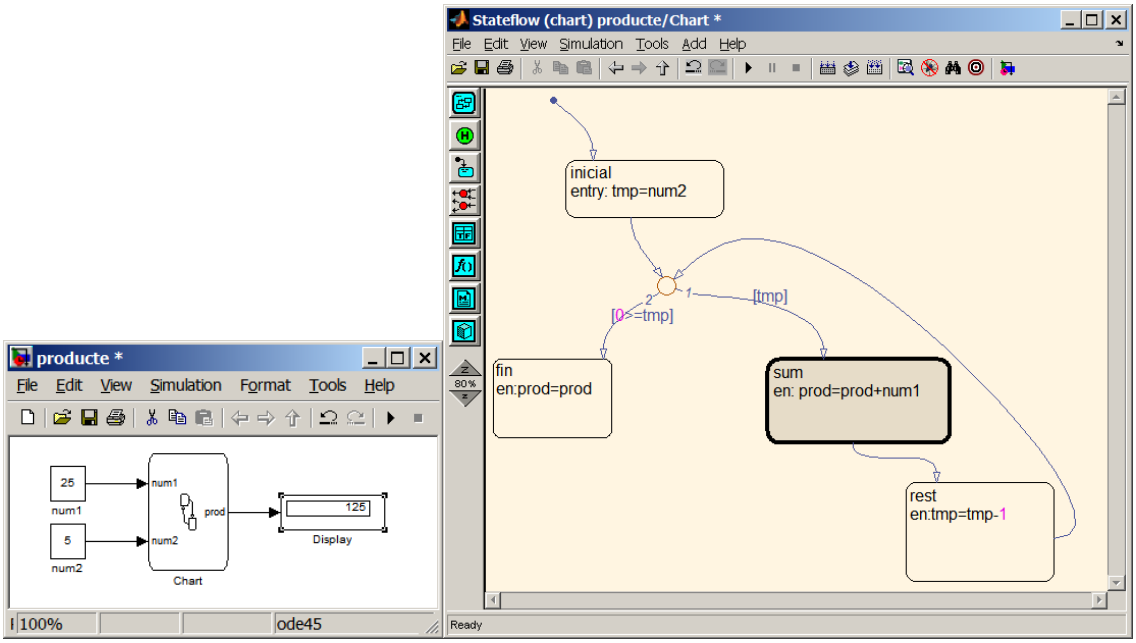
Fig. 23. Ejecución del modelo Stateflow

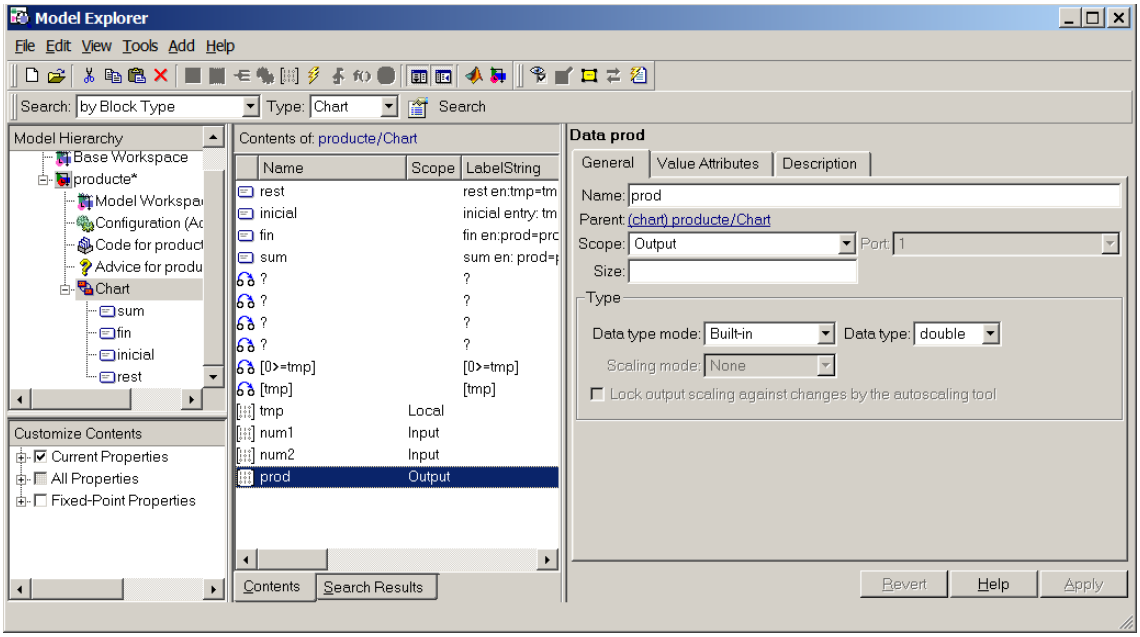
Acciones durante los estados: Pueden ser de entrada al estado, de salida del estado o a realizar mientras se está en el estado. Las acciones se indican mediante las etiquetas `entry:`, `during:`, `exit:` (o sus abreviaturas `en:`, `du:`…).

Si se quiere ejecutar una función de matlab dentro de estos bloques hay que usar la función ml, por ejemplo, `en:` `a=ml('sin(x)')` o bien en: `a=ml('sin(%f)',x)` si `x` es una variable local del Stateflow.

Ejemplo 3. Producto

El siguiente modelo implementa el producto de dos números:





4. Efectos de animación con Simulink

El siguiente ejemplo muestra cómo conseguir un efecto de animación por medio de funciones S y las herramientas GUI.

Ejemplo 4. Pelota que bota

En primer lugar, editar el siguiente fichero `bota.m`. Este fichero será introducido en un bloque *S-function* (de ahí la definición de argumentos de entrada y salida).

```
function [sys,x0] = bota(t,x,u,flag)
%
%BOTA  Animació:  jo tinc una pilota que bota, bota, bota...
%

global figH bolaH  %la H indica que aquestes variables són object handles

if not(isempty(flag)) & flag==0  %inicialització
    %
    figH=figure('Name','pilota que bota','numbertitle','off');
    %
    radi=1;phi=linspace(0,2*pi);
    bola=radi*exp(j*phi)+j;
    bolaH=patch(real(bola),imag(bola),'b');
    set(bolaH,'userdata',bola)
    axis([-5 5 0 10]),axis('square')
    %
    sys=[0 0 0 1 0 0];  %dimensions 0-var d'estat 1-entrada 0-sortides
    x0=[];  %condicions inicials no hi ha pq no hi ha [variables d'] estat
    %
end

if not(isempty(flag)) & flag==2  %actualització [posició] bola (posició: centre=u)
    %
    if any(get(0,'children')==figH)
        posic=u;
        bola=get(bolaH,'userdata');
        nova_bola=(bola+j*posic);
        set(bolaH,'xdata',real(nova_bola),'ydata',imag(nova_bola),'erase','normal');
    end
    %
    %sortida.  No torna res
    sys=[];
    x0=[];
    drawnow;
end

if not(isempty(flag)) & flag==9  %terminació
    close
end
```

En este fichero se ha usado la propiedad `UserData` del objeto `patch`. La propiedad `UserData` (de *user-specified data*) permite asociar una matriz de datos al objeto `patch`. En nuestro caso hemos asociado la matriz `bola` (que contiene los puntos que forman el contorno de la pelota y la posición de ésta) a fin de poderlo modificar a lo largo de la simulación con SIMULINK.

En segundo lugar, creamos un modelo SIMULINK (`pilota.mdl`) con los siguientes bloques. Este modelo es el que genera una señal *u* con la posición de la bola en cada instante de la simulación.

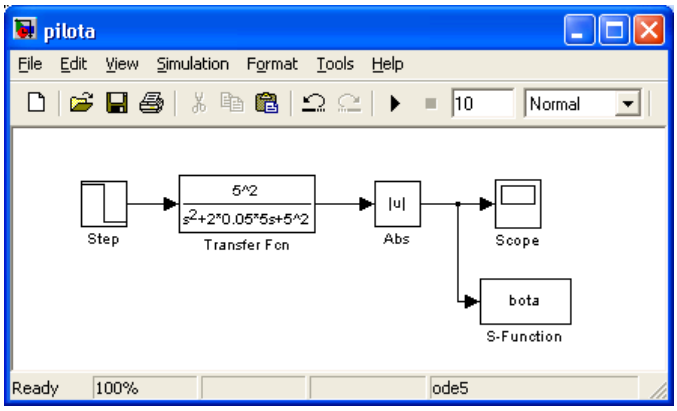


Fig. 24. Generación de la excitación

Los bloques `Step`, `Tranfer Fcn` y `S-Function` contienen los siguientes parámetros:

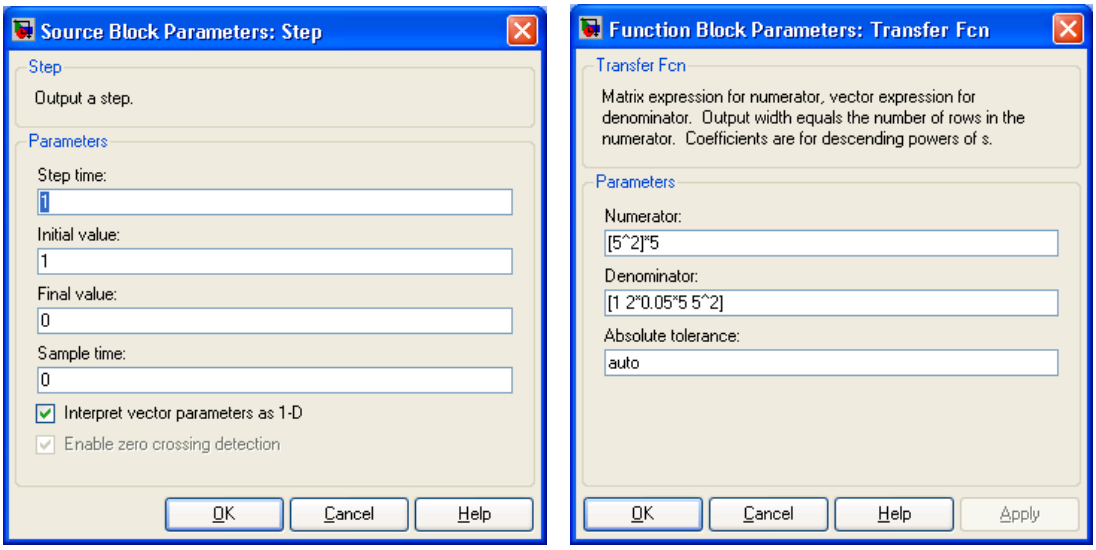


Fig. 25. Parámetros de Step y Transfer Fcn

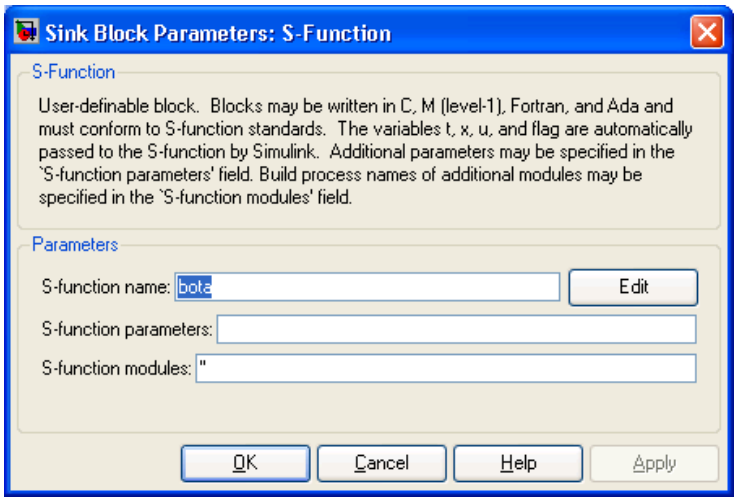
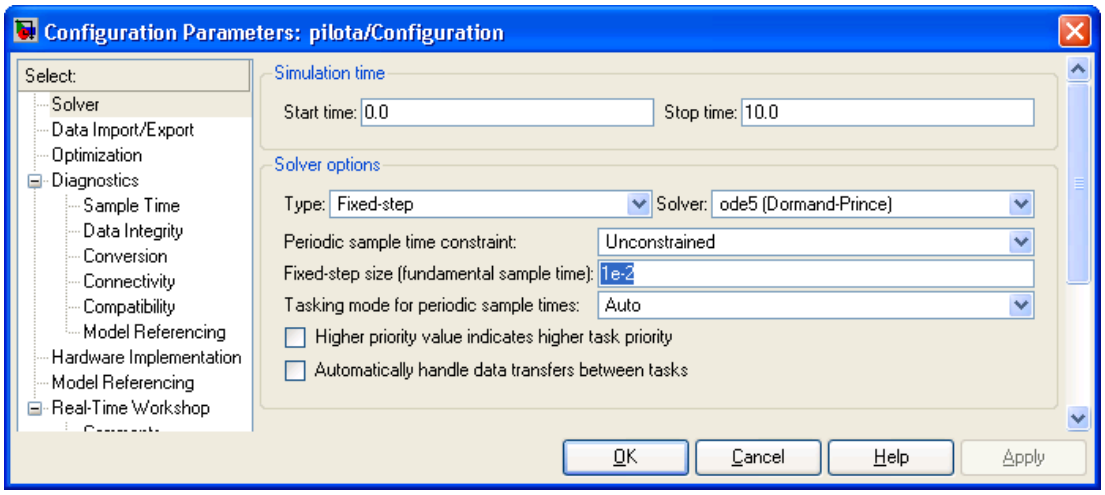


Fig. 26. Parámetros de S-Function

En tercer lugar, ajustar los parámetros de simulación (`Simulation` → `Configuration Parameters`) tal como se indica:



Finalmente, abrir el bloque `Scope` y clicar en el botón de `Run`. Los resultados de la simulación son:

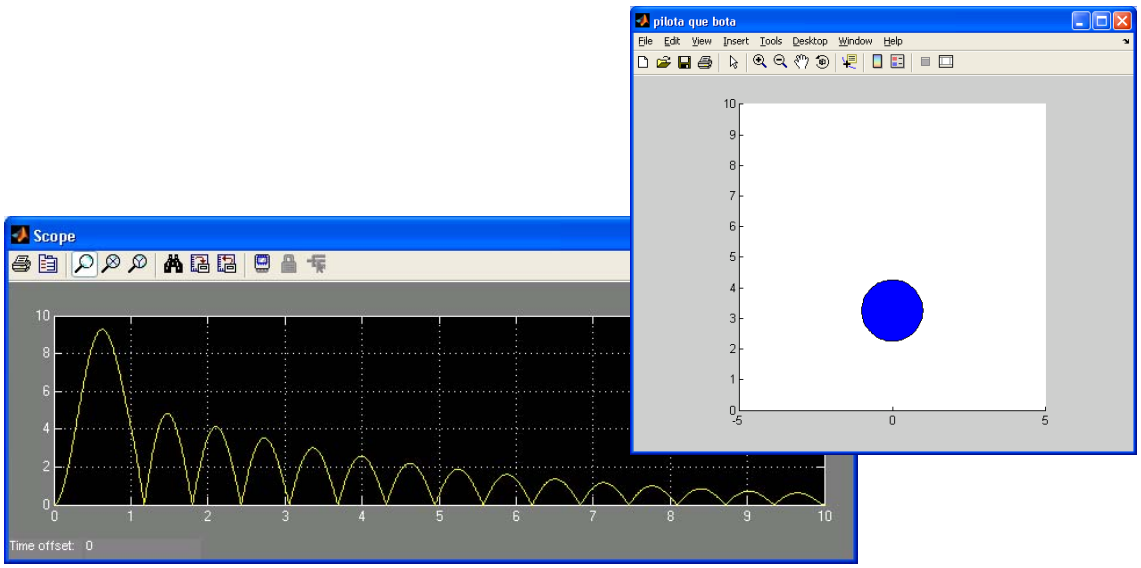


Fig. 27. Pelota que bota (efecto de animación)