UNIT 5

Simulink

1. Introduction ............................................................................................................... 1
2. Simulink ...................................................................................................................... 2
   2.1 Starting Simulink ...................................................................................................... 2
   2.2 Model building .......................................................................................................... 3
   2.3 Simulation parameters and Scope block ................................................................. 5
   2.4 Subsystems and masks ............................................................................................. 11
   2.5 S Functions ............................................................................................................. 13
3. Stateflow ................................................................................................................... 17
4. Animation effects with Simulink ............................................................................ 20

1. Introduction

SIMULINK is a special toolbox for MATLAB that simulates the dynamic systems behaviour. SIMULINK can simulate linear systems and nonlinear systems, continuous time models and discrete time models, combinations of all of them, and hybrid systems. SIMULINK is a graphical environment where each model is built using click-and-drag mouse operations. SIMULINK models are saved in *.mdl files (versions up to v7) and *.slx files (version v8).

In the newest versions, SIMULINK has extended its possibilities and block libraries (blocksets). In particular, the STATEFLOW package allows the simulation of state machines.

![Fig. 1. Matlab, Simulink, and Stateflow hierarchy](image-url)
Other interesting blocksets are, for instance, the communication ones (Communications Blockset, CDMA Reference Blockset) which include blocks to simulate devices such as PLLs or stations for emitting/receiving mobile signals; blocksets that allow the data exchange between SIMULINK and DSPs (Motorola, Texas Instruments) and between SIMULINK and different data acquisition cards (Real-Time Workshop); utility blocks (Dials & Gauges Blockset, Virtual Reality); and other advanced applications (Power Systems, Neural Network, Fuzzy Logic,...).

Type `>>ver` in the MATLAB command window to see which Simulink version is installed and which blocksets are available.

## 2. Simulink

### 2.1 Starting Simulink

To open the SIMULINK library browser, type `>>simulink` in the MATLAB command window, or click on the SIMULINK icon on the MATLAB toolbar (in v8).

![SIMULINK](image)

Fig. 2. Starting SIMULINK (in v6.1)

**Note:** if MATLAB has already been started, to edit an existing SIMULINK model, `modell.mdl`, you can enter `>>modell` in the MATLAB command window. If not,
you can double-click on the file name `model1.mdl` or `model1.slx` (this will start MATLAB and then open a window `model1` containing the model).

To create a new model select `File → New → Model` or click on the icon in the Simulink library (in v8).

Fig. 3. Starting SIMULINK (in v8)

### 2.2 Model building

**Model window:** Each model (or submodel) is built in a different window. To create a new model, choose `New` from the `File` menu or click on the corresponding icon (see Fig. 2). This opens a new model window, `untitled`. Then you can drag the blocks from the SIMULINK library browser to the model window. It is advisable to take a look to the options in the menu bar and in the toolbar of the model window.

**Block interconnection:** To interconnect blocks you can drag the mouse between the input and output ports of such blocks. An alternative is to select a block and, while pressing the `<ctrl>` key, select the other block. You can put labels everywhere (simply click on the desired location), change the block names and use different colours (`Format → Foreground Color`). You can also rotate the blocks (`Format → Flip block, Rotate block`), etc.
Example 1. Simple model building

Next figure shows how to build a simple model in order to simulate the step response of a continuous time second order system \( H(s) = \frac{1}{s^2 + 0.5s + 1} \).

The excitation block \textit{Step} has been dragged from the \textit{Sources} library, the block \textit{Scope} can be found in the \textit{Sinks} library and the continuous time system \textit{Transfer Fcn} can be found in the \textit{Continuous} library.

To change the default parameters and put ours, it is necessary to double-click the block \textit{Transfer Fcn} in the model window, and type the example numerator and denominator.

To run the model simulation, click on \( \text{Run} \) or select \textit{Simulation} \( \rightarrow \) \textit{Start}. To change the simulation parameters (stop time, sampling, etc.) select \textit{Simulation} \( \rightarrow \) \textit{Configuration Parameters...} (or \textit{Simulation} \( \rightarrow \) \textit{Simulation Parameters...} depending on the particular version).

To see the simulation result, click on the \textit{Scope} block to open it. Then click on \( \text{Autoscale} \) to auto scale the axes.

Fig. 4. Creating and running a new model
Next figure shows the same example in v8. Here the auto-scale icon is and the run icon is . Note also that the v8 version includes a model browser in the model window.

![Creating and running a new model (v8)](image)

**Fig. 5.** Creating and running a new model (v8)

### 2.3 Simulation parameters and Scope block

**Simulation parameters:** In version 6.x, simulation parameters can be modified from the menu bar: **Simulation → Simulation parameters.**

![Simulation parameters (v6)](image)

**Fig. 6.** Simulation parameters (v6)

In version 7.x, simulation parameters can be modified from the menu bar choosing the options: **Simulation → Configuration parameters.**
In version 8.x, simulation parameters can be modified from the menu bar choosing the options: Simulation → Model configuration parameters.... or clicking on the icon.

Run: To run the simulation click on or select Simulation → Start in the model window (see Example 1).

Function sim: It is also possible to run a Simulink model from the command window or from an M file with sim. In our example:

```
>> 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.

Interaction with the MATLAB workspace: SIMULINK models can access *.m files, *.mat files, and variables in MATLAB workspace by means blocks such as From File, From Workspace (Sources library), To File, To WorkSpace (Sinks library) and MATLAB Fcn (Functions & Tables library).

It is also possible refer to workspace variables inside a Simulink model. For example:
To run the model, these variables must have a value.

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

**Model Explorer:** Click on the Model Explorer icon to open the model explorer (in v8). In particular, the Base Workspace contains the variables available for the Simulink models.

**Scope block:** To visualize a simulation result, click on the Scope block to open the plot window. Optionally, auto-scale the result with (or in v8) or zoom x-axis.
and y-axis using the corresponding icons and in v8 in the scope window toolbar (see Fig. 10).

To include a title, click with the right button on the plot to open the context menu and select Axes properties…:

![Scope Title](image)

**Fig. 10. Scope title**

To modify the Scope parameters click on the Scope. By default, only the last 5000 samples are displayed on the Scope. If we want to see more than 5000 samples we must de-select the option Limit data points to last: in Data history (see Fig. 11).

The result of a Simulink simulation can be stored in the Matlab workspace by means of the Scope window (see Fig. 11).

![Scope Parameters](image)

**Fig. 11. Saving results to workspace**

The option Save data to workspace allows to pass to the workspace the values represented in the Scope. Three formats are available: Structure with time, Structure and Array.

*Format structure with time:* If we choose format Structure with Time, in the workspace we will see two new variables:
The variable ScopeData is a struct array with the following fields:

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

‘time’, ‘signals’ and ‘blockName’ are the three fields that form the structure ScopeData. Note that the field ‘signals’ is a structure, as well. To get more information about these data types, enter `>>help struct` and `>>help cell`.

*Functions fieldnames, isfield, getfield:* To see which fields are in ScopeData we can use fieldnames (this command stores the result in a cell array)

```matlab
>> names=fieldnames(ScopeData)  
names =  
    'time'  
    'signals'  
    'blockName'
```

To see if a specified field belongs to a struct: `isfield(ScopeData, 'time')`.

To access to the fields we can use the dot or the command `getfield`:

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

```matlab
>> ScopeData.signals.dimensions  
an =  
    1
```

```matlab
>> y=getfield(ScopeData,'signals')  
y =  
    values: [53x1 double]  
    dimensions: 1  
    label: ''  
    title: ''  
    plotStyle: 0
```
Note that the following commands yield to the same result:

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

**Array format:** If we have chosen the format *Array*, we can directly access the variables:

```matlab
>> size(ScopeData)
an =
      53     2
>> size(tout)
an =
      53     1
```

The following two commands have the same result:

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

An alternative way to explore the variables is using the workspace window and the variables editor:

![Workspace window](image)

**Fig. 13.** Accessing the simulation results
Structure format: In the previous example, if we had chosen the format Structure, the variables in the Matlab Workspace will be again ScopeData and tout, but now the field time in ScopeData will be empty.

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

2.4 Subsystems and masks

Creating subsystems: It is possible to group several blocks to obtain a subsystem. To do this, select with the mouse the corresponding blocks and then select Edit → Create Subsystem. (Note: the Gain block can be dragged from the Math Operations library or Math library, depending on the particular version.)

![Subsystem creation](image1)

Fig. 14. Selecting blocks…

![Subsystem creation](image2)

Fig. 15. …to create a subsystem.

Notice that, when creating a new subsystem, all input/output ports are created, numbered and labeled. To edit the subsystem, click on it. A new window containing the subsystem will be opened:
In the v8 version, the model browser shows all the subsystems. See Fig. 17.

**Masks:** It is possible to customize subsystem blocks by means of the mask editor (Edit \(\rightarrow\) Mask Subsystem, or from the context menu over the subsystem: Mask \(\rightarrow\) Create mask). You can change the subsystem icon, insert help comments, and introduce additional parameters (this is useful in the case of S functions that will be presented next). For instance, in version 7.x, if you put as Drawing commands the following command: \texttt{image(imread('b747.jpg'))}, the result is:
To modify an already created mask, select Edit → Edit Mask....

### 2.5 S Functions

S functions are SIMULINK blocks that can be programmed by the user (in MATLAB language or in C language). They are used to simulate complex dynamic systems (with time-varying parameters and/or with nonlinear components, different sampling rates, etc.). They are also used to communicate with external devices in *hardware-in-the-loop* applications. In this latter case, the S function contains the C program that allows the real time interaction between the acquisition card and the model input/output ports. In the case the code is written in the C language, it is necessary to compile and link the S functions using the `mex` utility of MATLAB.

**S functions for the simulation of dynamic systems**: There exist several template files (continuous time systems `csfunc.m`, discrete time systems `dsfunc.m`) that can be copied to the work folder to be modified with the particular user system and parameters.

```
>> which csfunc.m
C:\Program files\MATLAB704\toolbox\simulink\blocks\csfunc.m
```

S functions consist in a series of basic subroutines which SIMULINK executes in a sequential way:

- Initialization (`mdlInitializeSizes`), flag=0
- Derivative computation in continuous time systems (`mdlDerivatives`), flag=1
- Difference computation in discrete time systems (`mdlUpdate`), flag=2
- System output computation (`mdlOutputs`), flag=3, and
- Ending tasks (`mdlTerminate`), flag=9.

Note that this structure simulates (sample-to-sample) the system state equations. Firstly, the state equation (`mdlDerivatives`) \( \dot{x} = Ax + Bu \) is executed, and then the output equation (`mdlOutputs`) \( y = Cx + Du \) is executed. In the case of discrete time
systems, the first equation to be executed is the differences equation (mdlUpdate)
\[ x[n+1] = Ax[n] + Bu[n]. \]

**Csfunc function:** The template file `csfunc.m` contain the following statements. Note
that it corresponds to a system with 2 states, 2 inputs, and 2 outputs.

```matlab
>> 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 = [];
    %%%%%%%%%%%%%%%%%
    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;
```
To simulate an S function, drag the corresponding block to the model window. The S-Function block can be found in Simulink \(\rightarrow\) User-Defined Functions. Double-click on the block to edit it and write “csfunc” in the “S-function name” space.

Complete the Simulink model with an excitation block (sine) and a Scope block. Run the simulation. Note that the simulation gives error because we are trying to excite a two input system with a one-dimensional input signal:

![Error due to the dimensions](image)
To solve the problem drag a Demux block and run the simulation again. The result is now:

\[ \text{Fig. 21. Simulation of a 2 input 2 output system} \]

**Debugger:** Simulink has its own debugger. Click on \( \text{Debugger} \) to open it (go to Simulation → Debugger → Debug model in v8). If we had run the debugger before the simulation we had noticed the conflict in the ports dimension. See Fig. 22.

\[ \text{Fig. 22. Debugger in Simulink} \]
3. Stateflow

Start: To start STATEFLOW it is necessary that SIMULINK is already opened. To create a new Stateflow model, drag the “Chart” block to a new model window and double-click on it.

Fig. 23. Starting Stateflow

Palette: To create the state machine, drag the elements at the side of the Stateflow window. These constitutive elements are: States, Historic, Default transition, Connecting point (or decision point). More recent versions include Truth table, Function and Matlab Function.

Transitions are established by dragging the mouse between states and, to edit them, click on them and write on the “?” symbol.

Example 2. Lamp and timer.

We want to simulate the two states (“encendido” and “apagado”) of a particular device. The device is in “encendido” state if the external variable “hora” is greater than 12. The state is “apagado” if “hora” is smaller or equal to 12. The default state is “apagado”.

Firstly, drag two state blocks to the stateflow window and label them “apagado” and “encendido” respectively. Insert a default transition over “apagado” to indicate that it is
the default state. Draw the transitions between the two states and write the corresponding Boolean conditions. *(Remark: Boolean conditions are between [ ] while actions to execute during a transition are written between { }).*

**Fig. 24. Building a stateflow model (v6)**

To indicate that variable “hora” is external to the stateflow chart, select the explorer (click on 📦 in the toolbar or select View → Model Explorer in the menu bar). In the explorer, select Add → Data and set up the following parameters: Name=hora, Scope=Input from Simulink (or Input, depending on the particular version). Note that input “hora” appears now in the block chart of the Simulink model.

The model explorer presents many interesting options (see, for instance, the model advisor).

**Fig. 25. Model explorer in v7**
Now, generate a Simulink variable varying from 0 to 24 and excite the chart block with it. For instance, “hora” can be a sinus of amplitude 12, frequency 0.5Hz and offset 12. Final simulation time is 10s and sampling period is fixed and equal to 0.001.

To run the simulation, click on the corresponding button or select Simulation \rightarrow Start. In the Matlab command window will appear some messages regarding the building of the state machine

During the simulation, you can see which one of the two states is activated at each time.

**Fig. 26.** Stateflow chart execution

**Actions during the states:** They can be executed when entering the state (entry: or en:), during the state (during: or du:), or when leaving the state (exit: or ex:).

To execute a Matlab function during the states you must use the command ml, for example, en: a=ml('sin(x)') or en: a=ml('sin(%f)',x), if x is a local variable of Stateflow.
4. Animation effects with Simulink

Next example illustrates how to get an animation effect by means S-functions and GUI tools.

Example 3. Bouncing ball.

Firstly, create a Simulink model (pilota.mdl) with the following blocks:

The Step block and the Transfer Function block contain the following parameters:

The animation effect is implemented by the following S-function block:
where bota.m contains the following instructions:

```matlab
function [sys,x0] = bota(t,x,u,flag)
  
  if not(isempty(flag)) & flag==0  %inicialització
    figH=figure('Name','bota 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-ve 1-in 0-out
  x0=[];
  
  if not(isempty(flag)) & flag==2  %actualització bola
    if any(get(0,'children')==figH)
      posi=mu;
      nova_bola=(bola+ posi);
      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
end
```

Adjust the simulation parameters (Simulation → Configuration Parameters) to the following:
Open the Scope block and click on the Play button. The simulation results are:

![Simulation Results](image)

**Fig. 30.** Bouncing ball