**Telemark University College** Department of Electrical Engineering, Information Technology and Cybernetics



## Introduction to Simulink

HANS-PETTER HALVORSEN, 2011.06.06



Faculty of Technology, Postboks 203, Kjølnes ring 56, N-3901 Porsgrunn, Norway. Tel: +47 35 57 50 00 Fax: +47 35 57 54 01

## Preface

Simulink, developed by The MathWorks, is a commercial tool for modeling, simulating and analyzing dynamic systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries. It offers tight integration with the rest of the MATLAB environment and can either drive MATLAB or be scripted from it. Simulink is widely used in control theory and digital signal processing for simulation and design.

This training will give you the basic knowledge of Simulink and how you can use it together with MATLAB.

For more information about MATLAB and Simulink, see my Blog: <u>http://home.hit.no/~hansha</u>

## **Table of Contents**

Pı	eface	2
Та	able o	f Contentsiii
1	In	troduction to Simulink1
2	St	art using Simulink
	2.1	Block Libraries
	2.2	Create a new Model5
	2.3	Wiring techniques6
	2.4	Help window7
	2.5	Configuration9
	2.6	Examples
3	U	seful Features
	3.1	Comments/Labels
	3.2	Align and Distribute Blocks19
	3.3	Flip Blocks
	3.4	Hide Names
4	D	ata-driven Modelling23
	4.1	Command window23
	4.2	m-file
	4.3	Simulation Commands27
5	H	ybrid Systems (continuous and discrete) 29
6	E>	kample: Mass-Spring-Damper System
	6.1	Model
	6.2	Simulink

6	5.3	m-File	33
6	5.4	Results	33
7	Emb	pedded Algorithms	36
8	Sub	systems	40
9	Mo	del Explorer	44
10	Exe	rcises	45

## **1Introduction to Simulink**

Simulink is an environment for simulation and model-based design for dynamic and embedded systems. It provides an interactive graphical environment and a customizable set of block libraries that let you design, simulate, implement, and test a variety of time-varying systems, including communications, controls, signal processing, video processing, and image processing.

Simulink offers:

- A quick way of develop your model in contrast to text based-programming language such as e.g., C.
- Simulink has integrated **solvers**. In text based-programming language such as e.g., C you need to write your own solver.

## 2Start using Simulink

You start Simulink from the MATLAB IDE:

Open MATLAB and select the Simulink icon in the Toolbar:

# MATLAB 7.4.0 (R	2007a)					
Ale Edit Debug Desk	cop Window Help	Start Si	mulink			
日本人生産。	a e ( 🖌 🗗 🖬	Current Directory inc	nyenpynnaðit 💌 🗔 🕤			
Shortcuts [ How to Add	el What smulnk					
Current Directory ••	D * × Comm	and History	Command Window	H D P X	Workspace	XEDI
10 m & 10 -			To get started, select <u>MATLAB Help</u> or <u>Denos</u> from the Help menu.	×	'n 🖬 🌌 🖩	· 6
An ter ba Us - Al Fiera Straabulion m Straabulion m Straabulion m Straatu Test Case m Test Case m Test U.U.m Test U.U.m Test U.U.m	Type M-file M-file M-file M-file M-file	Size         Date Head           1 KB         15.10 JB           1 KB         22.10 JB           1 KB         15.10 R           1 KB         15.10 R	To yet started, seld <u>MELARITER</u> or <u>Carror</u> From the Helpmanu.     This is a Classeroom License for instructional use only.     Reenearch and momentally use in prohibited,     Sits Added paths for Samulation Interface Toolkit Version 2009     Starting the SIT Server on port 6011     SIT Server started     >>      )		10 m (20 Toone -	Voke
<		2			× 0	>

Or type "simulink" in the Command window, like this:

Command Window	Ite	8	×
To get started, select MATLAB Help or Demos from the Help menu.			×
This is a Classroom License for instructional use only. Research and commercial use is prohibited. SIT: Added paths for Simulation Interface Toolkit Version 2009 Starting the SIT Server on port 6011 SIT Server started >> simulink			

Then the following window appears (Simulink Library Browser):



The **Simulink Library Browser** is the library where you find all the blocks you may use in Simulink. Simulink software includes an extensive library of functions commonly used in modeling a system. These include:

- Continuous and discrete dynamics blocks, such as Integration, Transfer functions, Transport Delay, etc.
- Math blocks, such as Sum, Product, Add, etc
- Sources, such as Ramp, Random Generator, Step, etc

## 2.1 Block Libraries

Here are some common used **Continuous** Blocks:

3



Here are some common used Math Operations Blocks:



Add





Product



Sum

Here are some common used **<u>Signal Routing</u>** Blocks:



Demux

Here are some common used <u>Sinks</u> Blocks:



Scope



XY Graph

Here are some common used **Sources** Blocks:



Step



Signal Generator

	Ramp
M	Random Number
1	Constant

In addition there are lots of block in different Toolboxes:



### 2.2 Create a new Model

Click the New icon on the Toolbar in order to create a new Simulink model:



The following window appears:



You may now drag the blocks you want to use from the Simulink Library Browser to the model surface (or right-click on a block and select "Add to...").

#### Example:

In this example we place (drag and drop) to blocks, a Sine Wave and a Scope, on the model surface:



### 2.3 Wiring techniques

Use the mouse to wire the **inputs** and **outputs** of the different blocks. Inputs are located on the left side of the blocks, while outputs are located on the right side of the blocks.



When holding the mouse over an input or an output the mouse changes to the following symbol.

Use the mouse, while holding the left button down, to drag wires from the input to the output.

#### **Automatic Block Connection:**

Another wiring technique is to select the source block, then hold down the **Ctrl** key while left-clicking on the destination block.

Try the different techniques on the example above.

#### Connection from a wire to another block

If wire a connection from a wire to another block, like the example below, you need to hold down the **Ctrl** key while left-clicking on the wire and then to the input of the desired block.



### 2.4 Help window

In order to see detailed information about the different blocks, use the built-in Help system.

🙀 Function Block Parameters: Integrator	X
_ Integrator	
Continuous-time integration of the input signal.	
∠ Parameters	Ę
Evternal reset: none	
Initial condition:	
Limit euteut	
Link output	
Lower saturation limit:	
-inf	
Show saturation port	
Show state port	
Absolute tolerance:	
auto	
Ignore limit and reset when linearizing	
Enable zero crossing detection	
State Name: (e.g., 'position')	
П	
OK Cancel Help Apply	ה ר

All standard blocks in Simulink have detailed Help. Click the Help button in the Block Parameter window for the specific block in order to get detailed help for that block.

The Help Window then appears with detailed information about the selected block:



## 2.5 Configuration

There are lots of parameters you may want to configure regarding your simulation. Select "Configuration Parameters..." in the Simulation menu.



The following window appears:

Select:	Simulation time					
- Solver Data Import/Export	Start time: 🕕			Stop time: 10.0		
Diagnostics	Solver options	27. 				
Sample Time	Туре:	Variable-step	*	Solver:	ode45 (Dormand-Prince)	*
Data Validity	Max step size:	auto		Relative tolerance:	1e-3	
- Connectivity	Min step size:	auto		Absolute tolerance:	auto	
Compatibility	Initial step size:	auto		]		
Model Referencing	Zero crossing control:	Use local settings	*	]		
	Automatically handle data transfers between tasks					
- Real-Time Workshop Comments Symbols Symbols	- Solver diagnostic con Number of consecutiv	trols e min step size violations allowed:	1 10×129	×ana		
	Lonsecutive zero crossings relative tolerance:		1000			
Interface	Number of consecutive zero crossings allowed:			[1000.		
54-69/17/03/0						

Here you set important parameters such as:

- Start and Stop time for the simulation
- What kind of Solver to be used (ode45, ode23 etc.)
- Fixed-step/Variable-step

**Note!** Each of the controls on the Configuration Parameters dialog box corresponds to a configuration parameter that you can set via the "**sim**" and "**simset**" commands. You will learn more about these commands later.

Solvers are numerical integration algorithms that compute the system dynamics over time using information contained in the model. Simulink provides solvers to support the simulation of a broad range of systems, including continuous-time (analog), discrete-time (digital), hybrid (mixed-signal), and multirate systems of any size.

### 2.6 Examples

Below we will go through some examples in order to illustrate how to create block diagrams and related functionality.

**Example: Integrator with initial value** 

Create the following model (an integrator) and run the simulation:



#### Step1: Place the blocks on the model surface

This example use the following blocks:

	1	
l	s	
l	s	

Integrator



Constant



#### Step 2: Configuration

Scope



Integrator

Double-click on the Integrator block. The Parameter window for the Integrator block appears:

🗑 Function Block Parameters: Integrator
/ Integrator
Continuous-time integration of the input signal.
Parameters
External reset: none
Initial condition source external
Limit output
Upper saturation limit:
inl
Lower saturation limit:
-inf
Show saturation port
Show state port
Absolute tolerance:
auto
Ignore limit and reset when linearizing
Enable zero crossing detection
State Name: (e.g., 'position')
II

Select "Initial condition source=external". The Integrator block now looks like this:

I	1
I	×os

1
---

Constant

Double-click on the Constant block. The Parameter window for the Constant block appears:

🖬 Source Block Parameters: Constant
Constant Output the constant specified by the 'Constant value' parameter. If 'Constant value' is a vector and 'Interpret vector parameters as 1-D' is on, treat the constant value as a 1-D array. Otherwise, output a matrix with the same dimensions as the constant value.
Main Signal Data Types
Constant value:
Interpret vector parameters as 1-D
Sampling mode: Sample based 🛛 😪
Sample time:
inf
OK Cancel Help

In the Constant value field we type in the initial value for the integrator, e.g., type the value 1.

#### Step 3: Wiring

Use the mouse to wire the inputs and outputs of the different blocks.



When holding the mouse over an input or an output the mouse change to the following symbol.

#### 

Draw a wire between the output on the Constant block to the lower input in the Integrator block, like this:



You could also do like this:



Wire the rest of the blocks together and you will get the following diagram:



#### Step 4: Simulation

Start the simulation by clicking the "Start Simulation" icon in the Toolbar:



#### Step 5: The Results

Double-click in the Scope block in order to see the simulated result:



#### **Example: Sine Wave**

Create the block diagram as shown below:

🙀 Example	
File Edit View Simulation Format Tools Help	
D  ❷ ▋ ❹   X ħ €   ← → 수   Ω ⊆   ▶	<b>10.0</b>
Sine Wave Integrator Scope	
Ready 100% ode45	

Set the following parameter for the Integrator block:

🐱 Function Block Parameters: Integrator	×
Continuous-time integration of the input signal.	
Parameters	
External reset: none	
Initial condition source: internal	
Initial condition:	
Limit output	
Upper saturation limit:	
inf	
Lower saturation limit:	
-inf	
Show saturation port	
Show state port	
Absolute tolerance:	
auto	
Ignore limit and reset when linearizing	
Enable zero crossing detection	
State Name: (e.g., 'position')	
n	
UK Cancel Help Apply	

The result should be like this:

15



#### **Example: Using vectors**

Create the following block diagram:



For the Gain block, type the following parameters:

🗟 Function Block Parameters: Gain
Gain
Element-wise gain [ $y = K$ ."u] or matrix gain [ $y = K$ "u or $y = u$ "K].
Main Signal Data Types Parameter Data Types
Gain:
3:1:10
Multiplication: Element-wise(K.*u)
Sample time (-1 for inherited):
-1
OK Cancel Help Apply

As you see, we can use standard MATLAB syntax to create a vector.

If you want to see the signal dimensions, select "Signal Dimensions" and "Wide Nonscalar Lines" as shown here:

°				
tion	Format Tools Help			
, E	Font Text Alignment Enable TeX Commands		Þ	■ 10.0 Normal ▼
	Flip Name Flip Block Rotate Block	Ctrl+I Ctrl+R		
<u>А</u> 1е W.	Hide Name Show Drop Shadow Show Port Labels		►	n Saturation
	Foreground Color Background Color Screen Color		* * *	
	Port/Signal Displays		Þ	Sample Time Colors
	Block Displays Library Link Display		) }	<ul> <li>Linearization Indicators</li> <li>Port Data Types</li> </ul>
				Storage Class
		100	9%	Testpoint Indicators     Viewer Indicators     Wide Nonscalar Lines

The block diagram should now look like this:



The thick lines indicate vectors, while the number (8) is the size of the vector.

Let's change the Saturation block:

🐱 Function Block Parameters: Saturation	×
∽ Saturation	
Limit input signal to the upper and lower saturation values.	
Parameters	
Upper limit:	
rand(8,1)*10	
Lower limit:	
rand(8,1)*-10	
Treat as gain when linearizing	
Enable zero crossing detection	
Sample time (-1 for inherited):	
-1	
OK Cancel Help Apply	

As you see you may use standard MATLAB functions and syntax.

Run the simulation and see the results in the Scope block.

## **3Useful Features**

You should know about these features in Simulink in order to take full advantage of Simulink.

## 3.1 Comments/Labels

Double-click on your surface in order to write Labels or Comments in your model block diagram.



## 3.2 Align and Distribute Blocks

You may align your blocks:



### 3.3 Flip Blocks

Normally the inputs are on the left and the outputs on the right, but in some cases it is convenient to have the inputs on the right and output on the left side.



In order to "flip" the input and outputs right-click on the block and select "Flip Block".



### 3.4 Hide Names

Hide default labels that appear on the diagram, such as Constant1, Integrator1, etc.

Select "Hide Names" on the "Format" menu:



## 4Data-driven Modelling

You may use Simulink together with MATLAB in order to specify data and parameters to your Simulink model. You may specify commands in the MATLAB Command Window or as commands in an m-file. This is called data-driven modeling.

### 4.1 Command window

Example:



Note! In order to get 3 inputs on the Scope block:

Double-click on the Scope and select the Parameters icon in the Toolbar:



Then select Number of Axes=3:

General	Data history	Tip: try right clicking on ax
Number	ofaxes:	floating scope
Time rai	nge: auto	
Tick lab	els: bottom axis on	ly 🖌
Samplin	ig	
Decima	tion 🔽 1	

Configure the zero-order hold blocks like this:

🐱 Function Block Parameters: Zero-Order Hold 🛛 🛛 🔁
CZero-Order Hold
Zero-order hold.
Parameters
Sample time (-1 for inherited):
Ts1
OK Cancel Help Apply

🙀 Function Block Parameters: Zero-Order Hold1	
CZero-Order Hold	
Zero-order hold.	
Parameters	
Sample time (-1 for inherited):	
12	
OK Cancel Help A	pply

Write the following in the **Command window** in MATLAB:

Command Window	. H □ ₹ X
To get started, select <u>MATLAB Help</u> or <u>Demos</u> from the Help menu.	×
>> Ts1=0.2	
Ts1 =	
0.2000	
>> Ts2=0.5	
Ts2 =	
0.5000	
>>	

#### Run the Simulink model from the Simulink:



We then get the following results:



### 4.2 m-file

It is good practice to build your in Simulink and configure and run the simulation from a MATLAB m-file.

A Typical m-file could look like this:



You use the **simset** command to configure your simulation parameters and the **sim** command to run the simulation.

The variables you refer to in the m-file is set in the Constant value field in the Parameter window for each block.

Source Block Parameters: Constant1
Constant Output the constant specified by the 'Constant value' parameter. If 'Constant value' is a vector and 'Interpret vector parameters as 1-D' is on, treat the constant value as a 1-D array. Otherwise, output a matrix with the same dimensions as the constant value.
Main Signal Data Types Constant value: dxdt_init V Interpret vector parameters as 1-D
Sampling mode: Sample based
OK Cancel Help

#### 4.3 Simulation Commands

The most used command is:

- simset
- sim

Use these commands if you configure and run your Simulink model from a m-file.

Example:

```
%Simulator Settings
t_stop=100; %[s]
T_s=t_stop/1000; %[s]
options=simset('solver', 'ode5', 'fixedstep', T_s);
%Starting simulation
sim('mass_spring_damper', t_stop, options);
```

# 5Hybrid Systems (continuous and discrete)

You may mix continuous blocks and discrete blocks in the same system, so-called Hybrid systems.

#### **Example: Hybrid System**

Create the following block diagram:



The Block diagram now looks like this:



The black color is the continuous system while the colored part (red and green) is the discrete part of the system.

# 6Example: Mass-Spring-Damper System

#### 6.1 Model

In this example we will create a mass-spring-damper model in Simulink and configure and run the simulation from a MATLAB m-file.

In this exercise you will construct a simulation diagram that represents the behavior of a dynamic system. You will simulate a spring-mass damper system.

$$F(t) - c\dot{x}(t) - kx(t) = m\ddot{x}(t)$$

where t is the simulation time, F(t) is an external force applied to the system, c is the damping constant of the spring, k is the stiffness of the spring, m is a mass, and x(t) is the position of the mass.  $\dot{x}$  is the first derivative of the position, which equals the velocity of the mass.  $\ddot{x}$  is the second derivative of the position, which equals the acceleration of the mass.

The following figure shows this dynamic system.



The goal is to view the position x(t) of the mass m with respect to time t. You can calculate the position by integrating the velocity of the mass. You can calculate the velocity by integrating the

acceleration of the mass. If you know the force and mass, you can calculate this acceleration by using Newton's Second Law of Motion, given by the following equation:

Force = Mass × Acceleration

Therefore,

Acceleration = Force / Mass

Substituting terms from the differential equation above yields the following equation:

$$\ddot{x} = \frac{1}{m}(F - c\dot{x} - kx)$$

You will construct a simulation diagram that iterates the following steps over a period of time.

### 6.2 Simulink

Create the block diagram for the mass-spring-damper model above.

Instead of hard-coding the model parameters in the blocks you should refer to them as variables set in an m-file.

🐱 Source Block Parameters: Constant 🛛 🛛 🔀
Constant Output the constant specified by the 'Constant value' parameter. If 'Constant value' is a vector and 'Interpret vector parameters as 1-D' is on, treat the constant value as a 1-D array. Otherwise, output a matrix with the same dimensions as the constant value.
Main Signal Data Types
Sampling mode: Sample based
Sample time: inf
OK Cancel Help

These variables should be configured:

• x\_init

- dxdt\_init
- m=
- c=
- k
- t\_step\_F
- F\_O
- F\_1

### 6.3 m-File

The following variables should then be set in the m-file:

```
x_init=4; %[m]. Initial position.
dxdt_init=0; %[m/s]. Initial Speed.
m=20; %[kg]
c=4; %[N/(m/s)]
k=2; %[N/m]
t_step_F=50; %[s]
F_0=0; %[N]
F 1=4; %[N]
```

### 6.4 Results

The Block Diagram should look something like this:



The m-File should look something like this:

2	ditor - M:\Work\Training\MATLAB\An Introduction to Simulink\Code\Mass-Spring-Dam 🔳 🗖	$\mathbf{X}$
File	Edit Text Go Cell Tools Debug Desktop Window Help 🏾 🍟	e x
۵		~
0	*= C= = - 1.0 + ÷ 1.1 × %* %* 0	-
1	Script of mass-spring-damper simulator.	
2	%Hans-Petter Halvorsen. 20.11.2009	
3		
4	%Nodell Parameters	
5	- x_init=4; %[m]. Initial position.	
6	- dxdt_init=0; %[m/s]. Initial Speed.	
7	- m=20; %[kg]	
8	-c=4; [N/(m/s)]	
9	- k=2; *[N/m]	
10	- t_step_F=50; %[s]	
11	- F_O=U; %[N]	
12	- F_1=4; %[N]	
1.4	Scinulator Cattings	
15	- t ston=100. %[s]	
16	- T s=t ston/1000: %[s]	
17	<pre>- ontions=simset('solver', 'ode5', 'fixedsten', T s):</pre>	
18		
19	%Starting simulation	
20	<pre>sim('mass_spring_damper', t_stop, options);</pre>	
	script In 16 Col 22 OV	R

#### Graphs:

Force F



Position x and speed  $\dot{x}$ :



## 7Embedded Algorithms

This chapter explains how you incorporate an existing MATLAB function into your Simulink model.

Make sure your MATLAB function is compiled as an embedded MATLAB function using the #eml directive, e.g.:





Drag in the Embedded MATLAB function:



Double-click on the Embedded MATLAB function give us the standard template for an embedded function:



Modify the template so it calls your MATLAB function:



Wire the system like this:



Run the Simulation:



## 8Subsystems

You create subsystems to create hierarchical systems and hide details in the model.

Select the part of your system from which you want to create a subsystem, right-click and select "Create Subsystem".



Example:



Right-click and select "Create Subsystem":



If we double-click on the subsystem we see the blocks in the subsystem:



Right-click on the block and select "Edit Mask" in order to open the Mask Editor:

🛎 Mask Editor : Subsystem	
Icon Parameters Initialization Documentation	
Icon options       Drawing commands         Frame       Visible         Visible       Visible         Transparency       Opaque         Opaque       Visible         Rotation       Fixed         Units       Autoscale	
Examples of drawing commands Command port_label (label specific ports)	×y>
Unmask OK Cancel Help	Apply

41

	1
	1
Integrator	Scope
Subsystem	
	L
Mask Editor : Subsys	stem
ICOD   Parameters   Toiti	alization   Documentation
Icon options	Drawing commands
Frame	plot (peaks);
Visible	
Transparency	
Transparency	
Opaque	

The Mask Editor allows you to change how the subsystem should look, e.g., the subsystem icon.

Set Parameters for the subsystem:

Prompt	Variable	Тур	е	Evaluate	Tun
Sample Time 1	Ts1	edit	٣	2	
Sample Time 2	Ts2	edit	+	V	3

Double click on the sub system now gives the Parameter window for the subsystem:

Parameters Sample Time 1	ied rate	(mask) signals at a spe	ny custom subsy osystem samples	1		
			ters Time 1 Time 2	Subsystem	$-\frac{1}{s}$	ne Wave

## 9Model Explorer

The Model Explorer allows you to quickly locate, view, and change elements of a Simulink model or Stateflow chart.

To display the Model Explorer, select Model Explorer from the Simulink View menu.



## 10 Exercises

In this chapter we provide more exercises.

#### **Exercise:**

Build the following block diagram:



In this exercise we will model a rubber ball that is thrown in the air with an initial velocity of 15 m/s from a height of 10 m. We will model the dynamics of the ball as it bounces, under the influence of gravity. We will assume that 20% of the energy is lost on each bounce. (That is, after each impact, the ball will travel at 80% of its prior velocity, but in the opposite direction.)

We can model this example by integrating g (g = -9.81m/s<sup>2</sup>) over time with the initial condition set to 15 m/s. We reset the integrator each time the position reaches zero meters and set the new initial condition to -80% of the impact velocity. Position is modeled by integrating the velocity over time with the initial condition set to 10m/s.

The result should be:



Watch this video to see the result:

http://www.mathworks.com/products/demos/simulink/Simulink\_Key\_Features/videos/building.h tml Telemark University College Faculty of Technology Kjølnes Ring 56 N-3914 Porsgrunn, Norway <u>www.hit.no</u>

Hans-Petter Halvorsen, M.Sc.

**Telemark University College** 

Department of Electrical Engineering, Information Technology and Cybernetics

Phone: +47 3557 5158

E-mail: hans.p.halvorsen@hit.no

Blog: <u>http://home.hit.no/~hansha/</u>

Room: B-237a