MATLAB and SIMULINK Tutorial for ELG3311

TAs

Peng He and Saeed Salehi

What is MATLAB?

- It stands for MATrix LABoratory
- It is developed by The Mathworks, Inc.
  - (*http://www.mathworks.com*)
- It is an interactive, integrated, environment
  - for numerical computations
  - for symbolic computations
  - for scientific visualizations
- It is a high-level programming language
  - Program (or script, actually) runs in interpreted, as opposed to compiled, mode
- Many application-specific toolboxes (functions) available

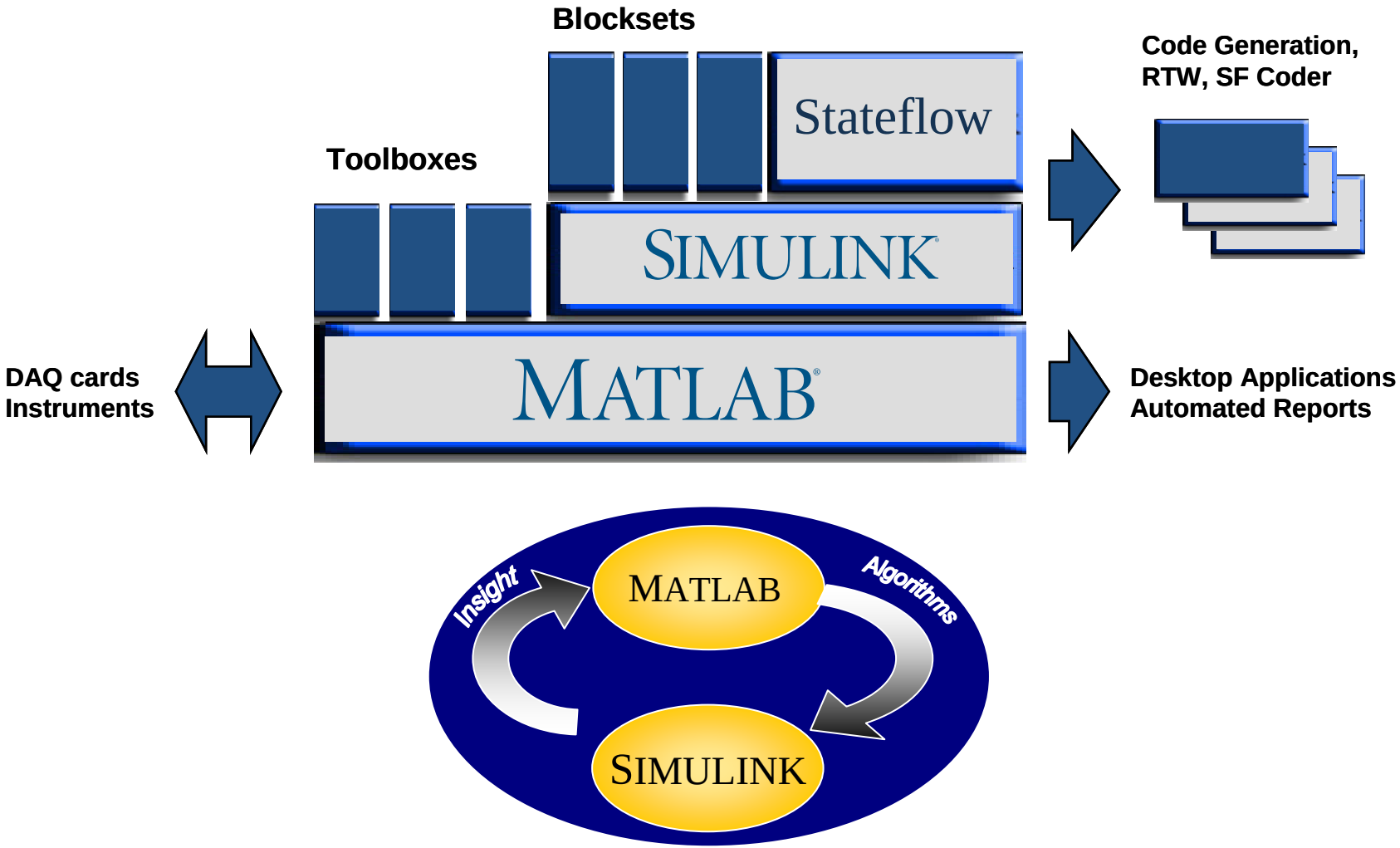
Strengths of MATLAB

- MATLAB is relatively easy to learn
- MATLAB code is optimized to be relatively quick when performing matrix operations
- MATLAB may behave like a calculator or as a programming language
- MATLAB is interpreted, errors are easier to fix
- Although primarily procedural, MATLAB does have some object-oriented elements

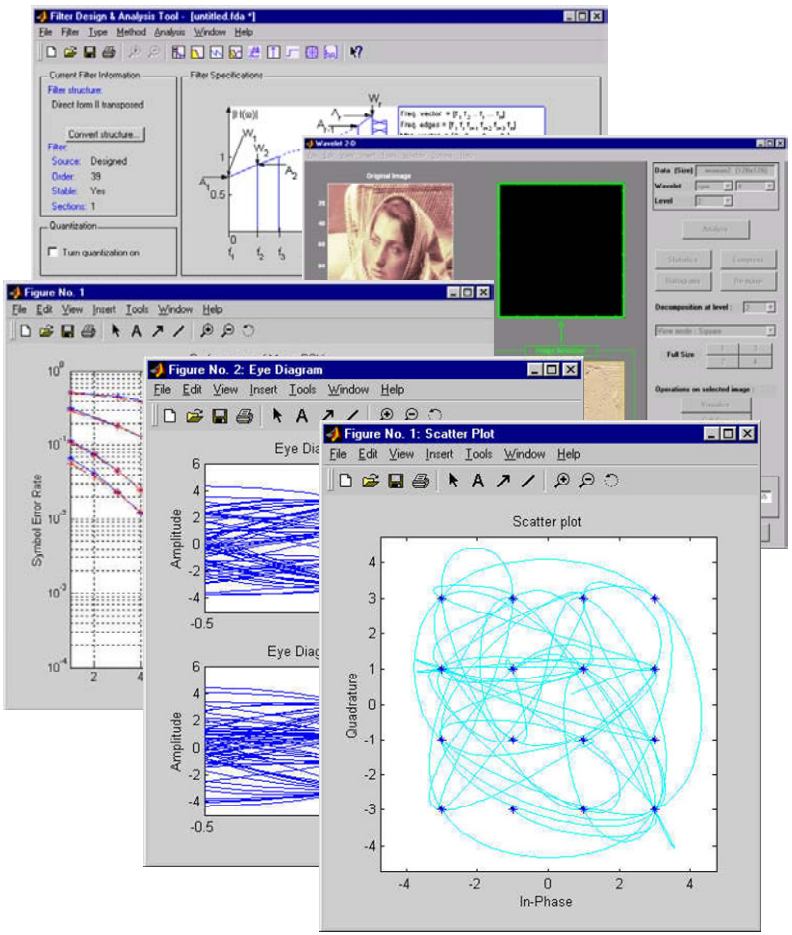
Weaknesses of MATLAB

- MATLAB is NOT a general purpose programming language
- MATLAB is an interpreted language (making it for the most part slower than a compiled language such as C++)
- MATLAB is designed for scientific computation and is not suitable for some things (such as parsing text)

The MathWorks Product Family



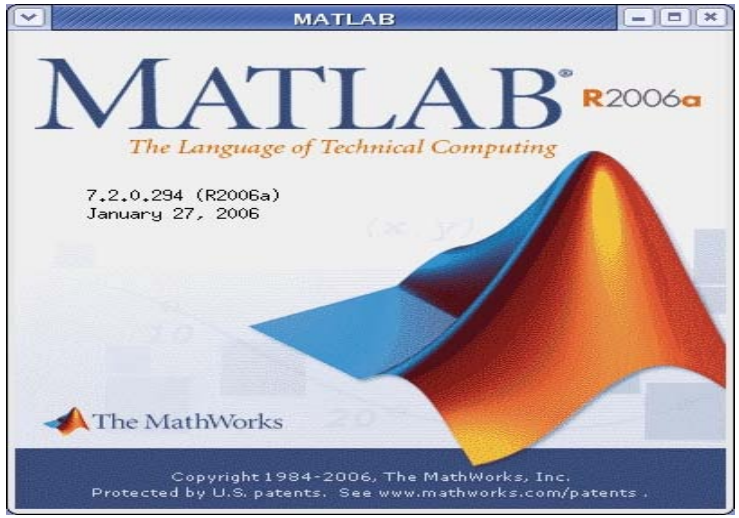
Toolboxes



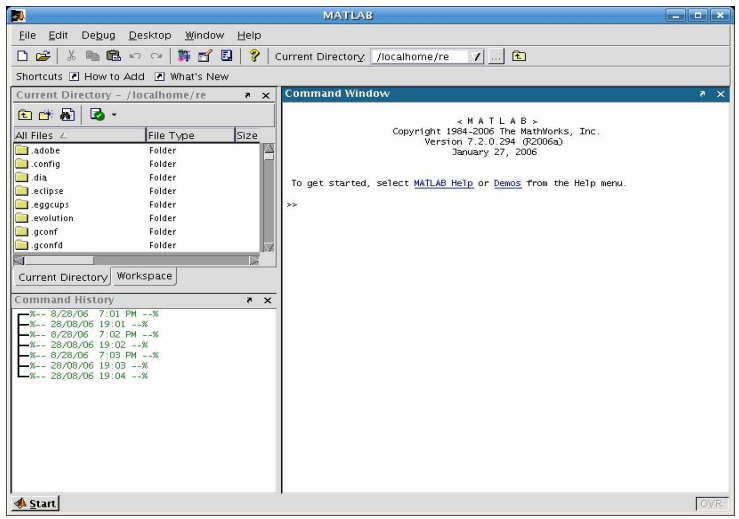
- Signal Processing
- Communications
- Filter Design
- Wavelet Analysis
- Statistics
- Optimization
- Image Processing
- Others…

Starting MATLAB

- When starting up the MATLAB program, MATLAB loads, checks that a license is available and throws up a splash screen.

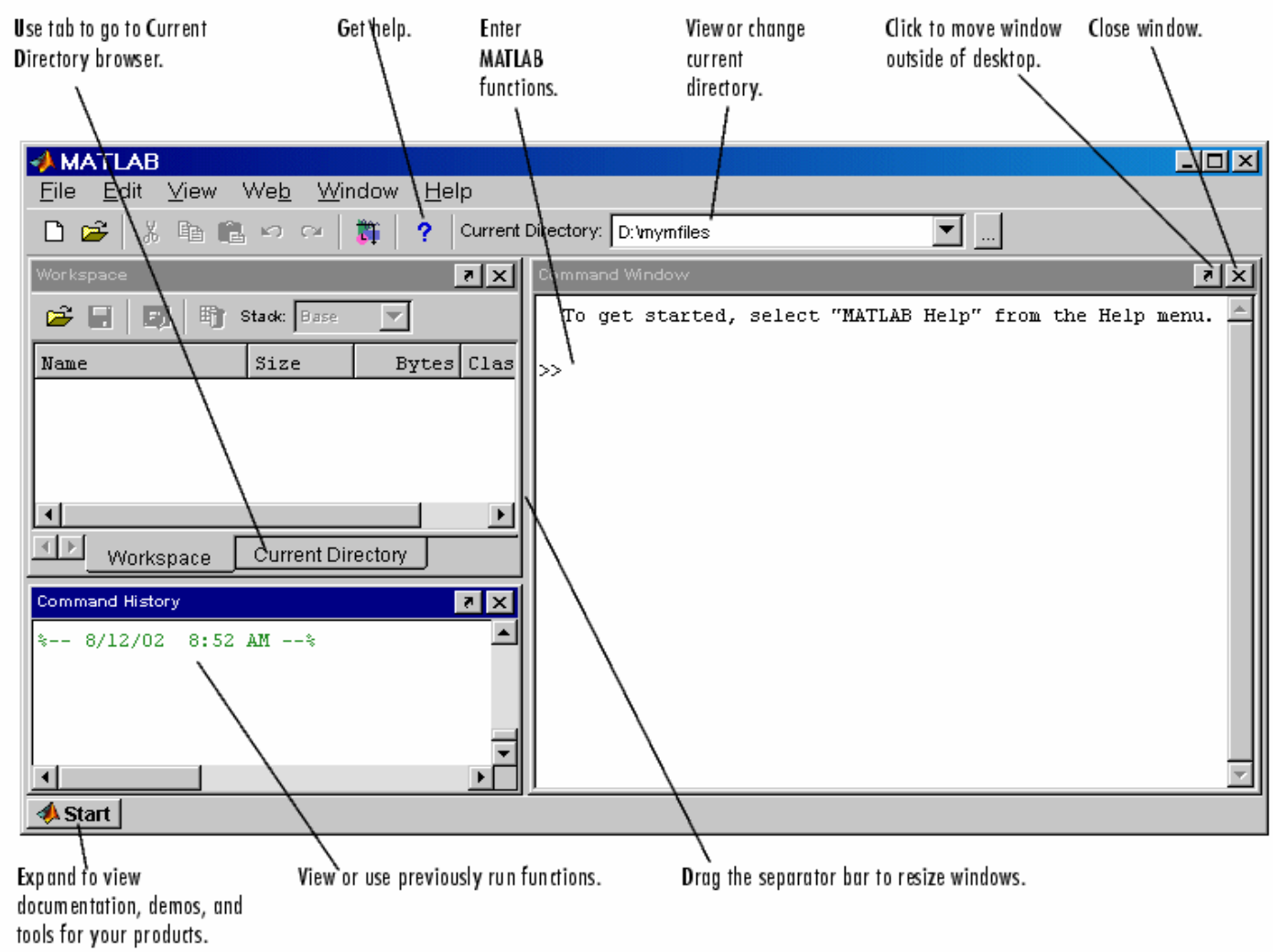


- Next MATLAB's desktop appears.



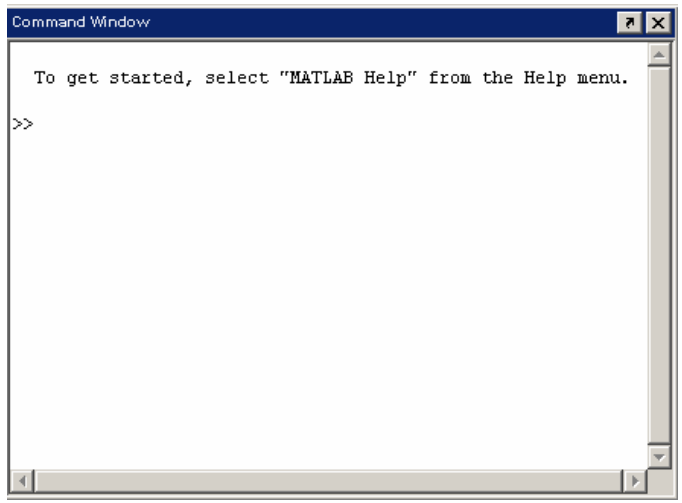
I. MATLAB Desktop

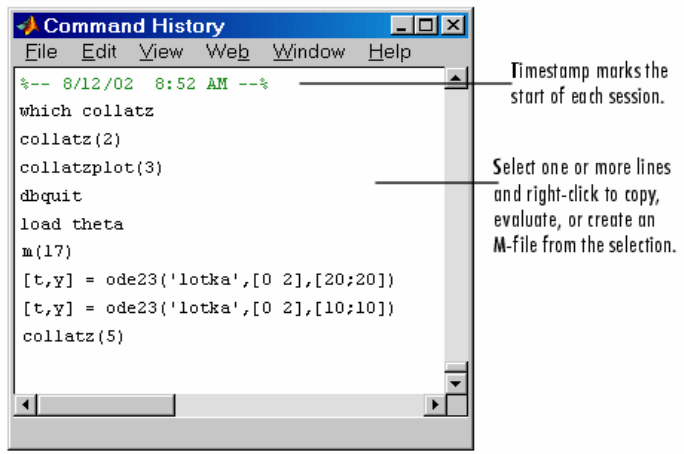
MATLAB Desktop



MATLAB Desktop: Command Window

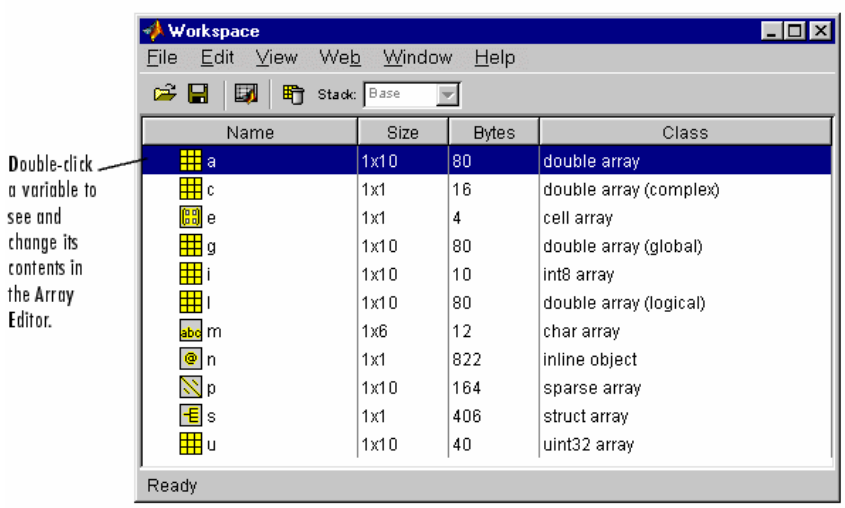
- Use the command window to enter variables and run functions and M-files.
  - For example:
    - a = 2.5;
    - b = ones(5,5);
- Command History
  - Statements you enter in the Command window are logged in the Command History.





MATLAB Desktop: Workspace Browser

- The MATLAB workspace consists of the set of variables (named arrays) built up during a MATLAB session and stored in memory.
- You add variables to the workspace by using functions, running M-files and loading saved workspaces.
- To delete variables from the workspace, select the variable and select Delete from the Edit menu. Alternatively, use the "clear" function.



Scripts and Functions: M-file

- Sequences of MATLAB commands can be written to files with the extension `.m`, appropriately called M-files.
- Entering the name of the file (without the extension!) causes automatic execution of all the statements. In their simplest form, such files are called script files.
- Script files do not take the input arguments or return the output arguments.
- The function files may take input arguments or return output arguments.

An Example of Script Files

- Create a file by the name, say, `mytest.m`.
- Contents of `mytest.m` :

```
x=45*pi/180; % convert degrees to radians
a=sin(x); % compute sine 45 degrees
b=cos(x); % compute cosine 45 degrees
disp('sin(45*pi/180)') % print header disp(a) % print result
```

An Example of Function Files

```
function y = cosgen(x,a,f,p)

%COSGEN Generation of a cosine wave
% y = cosgen(x,a,f,p)
% y - cosine of x
% a - amplitude
% f - frequency [hertz]
% p - phase [radians]

y = a*cos( 2*pi*f*(x + p/(2*pi*f)) );
```

Matlab also has many built-in functions

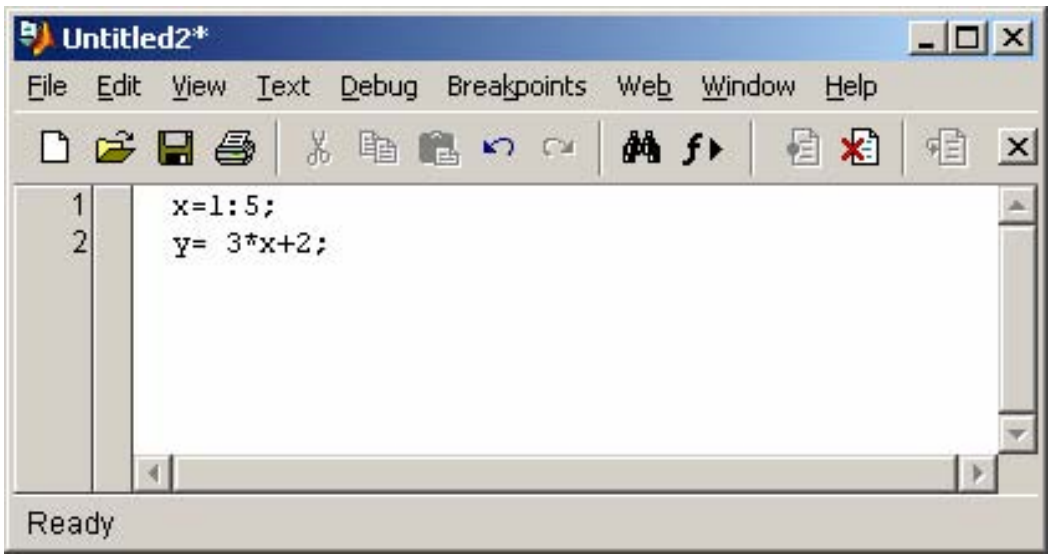
- >> abs(x) % absolute value of x
- >> exp(x) % e to the x-th power
- >> fix(x) % rounds x to integer towards 0
- >> log10(x) % common logarithm of x to the base 10
- >> rem(x,y) % remainder of x/y
- >> sqrt(x) % square root of x
- >> sin(x) % sine of x; x in radians
- >> acoth(x) % inversion hyperbolic cotangent of x
- >> help elfun % get a list of all available elementary functions

```
>> sqrt(64)
ans = 8

>> sin(pi/2)
ans = 1

>> abs(-56)
ans = 56
```

```
>> e = ones(3,3)
e =
1 1 1
1 1 1
1 1 1

>> size(e)
ans =
3 3
```

Editing M-file through Editor Window

- Use the Editor/Debugger to create and debug M-files, which are programs you write to run MATLAB functions.
- The Editor/Debugger provides a GUI for basic text editing, as well as for M-file debugging
  - Create a new M-file: File → New → M-file
  - Open an M-file: File → Open



Rules on Variable and Function Names

- Variable/Function name
  - begins with a LETTER, e.g., A2z.
  - can be a mix of letters, digits, and underscores (e.g., vector_A, but not vector-A (since "-" is a reserved char).
  - is case sensitive, e.g., NAME, Name, name are 3 distinct variables.
  - must not be longer than 31 characters.
  - Suggestion:
    - Since MATLAB distinguishes one function from the next by their file names, name files the same as function names to avoid confusion.
    - Use only lowercase letter to be consistent with MATLAB's convention.
- File name
  - Files that contain MATLAB commands should be named with a suffix of ".m", e.g., something.m.
  - These include, but not restricted to, script m-files and function m-files.
  - Note: To use it, just refer to it by name, without the suffix, e.g.,
  - >> something

Operators

Arithmetic Operators

Operator	Description
+	Addition
–	Subtraction
.*	Multiplication
./	Right division
.\	Left division
+	Unary plus
–	Unary minus
:	Colon operator
.^	Power
.'	Transpose
'	Complex conjugate transpose
*	Matrix multiplication
/	Matrix right division
\	Matrix left division
^	Matrix power

Relational Operators

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

Logical Operators

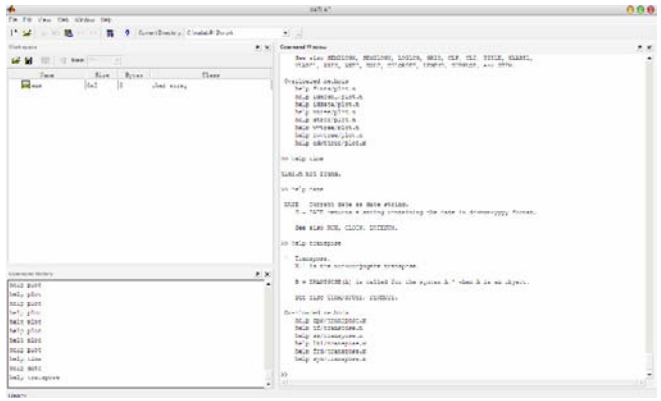
Operator	Description
&	AND
|	OR
~	NOT

Hints for Editing and Run a M-file

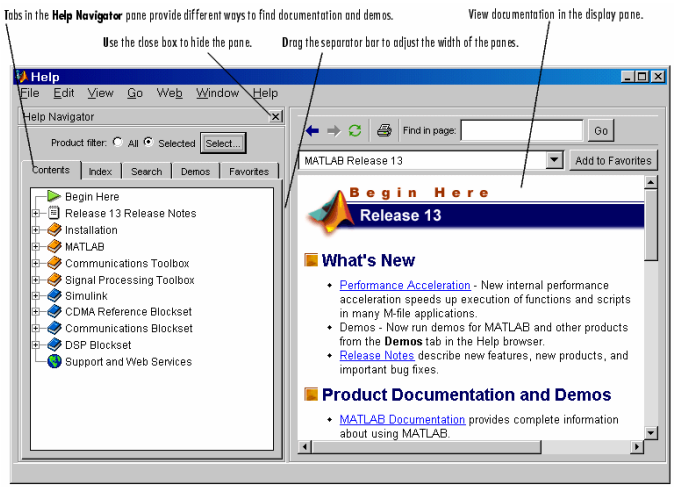
- Hints
  - "%" is for comments.
  - ";" delimits statements; suppresses screen output
  - ":" create lists with fixed step
  - "..." statement continuation, e.g.,
    - `>> x = [ 1 3 5 ...`
    - `7 9]; % x = [1 3 5 7 9] splitted into 2 lines`
  - "," -- command delimiter, e.g.,
    - `>> x = [1:2:9], y = [1:9] % two statements on the same line`
  - Define some variables:
  - x = 1:5; y= 3*x+1;
  - "clear all; close all;" at the top-line of .m file
  - Use MATLAB functions and programming language.
- Run the M-file: (if the M-file is under the current directory)
  - In the command window, input the name of the M-file and then ENTER.
  - In the editor window, press F5.

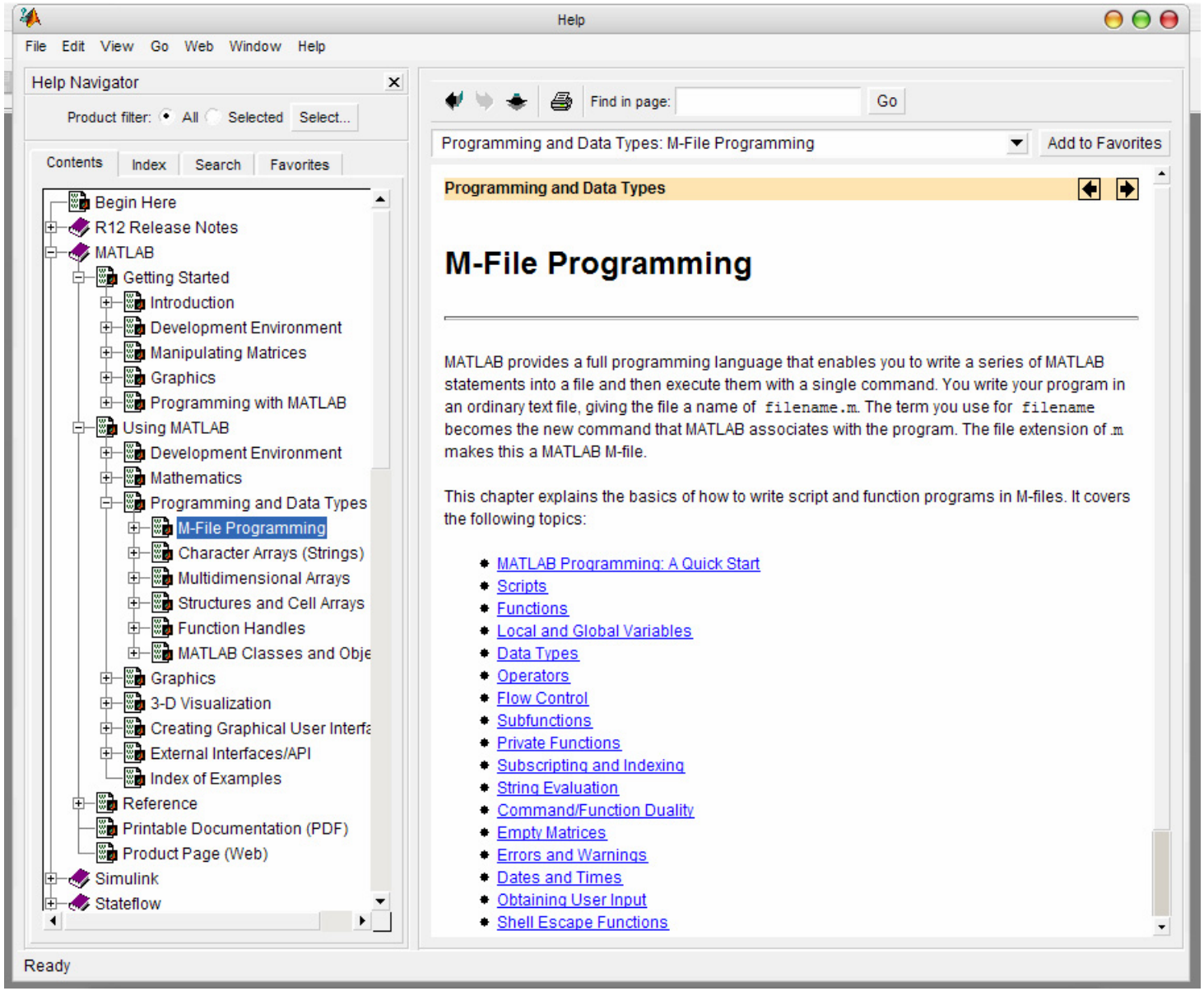
How to find help in MATLAB?

- "On-line" help
  - To find out about the syntax for any of the commands you can type
    >>help <commandname >
    inside of MATLAB.



- Help browser
  - Use the Help browser to search and view documentation and demos for all your MathWorks products.
  - To open the Help browser, click the help button in the toolbar,
  - or type "helpbrowser" in the Command Window.




Help
File Edit View Go Web Window Help
Help Navigator
Product filter: All Selected Select...
Contents Index Search Favorites
Begin Here
R12 Release Notes
MATLAB
Getting Started
Introduction
Development Environment
Manipulating Matrices
Graphics
Programming with MATLAB
Using MATLAB
Development Environment
Mathematics
Programming and Data Types
M-File Programming
Character Arrays (Strings)
Multidimensional Arrays
Structures and Cell Arrays
Function Handles
MATLAB Classes and Obje
Graphics
3-D Visualization
Creating Graphical User Interfa
External Interfaces/API
Index of Examples
Reference
Printable Documentation (PDF)
Product Page (Web)
Simulink
Stateflow
Find in page:
Go
Programming and Data Types: M-File Programming
Add to Favorites
Programming and Data Types
M-File Programming
MATLAB provides a full programming language that enables you to write a series of MATLAB statements into a file and then execute them with a single command. You write your program in an ordinary text file, giving the file a name of filename.m. The term you use for filename becomes the new command that MATLAB associates with the program. The file extension of .m makes this a MATLAB M-file.
This chapter explains the basics of how to write script and function programs in M-files. It covers the following topics:
MATLAB Programming: A Quick Start
Scripts
Functions
Local and Global Variables
Data Types
Operators
Flow Control
Subfunctions
Private Functions
Subscripting and Indexing
String Evaluation
Command/Function Duality
Empty Matrices
Errors and Warnings
Dates and Times
Obtaining User Input
Shell Escape Functions
Ready

II. MATLAB BASICS

Introduction to Vectors in Matlab

- MATLAB is designed to work with matrices, but you can also input scalars and vectors since they can be considered as matrices with dimension 1x1 (scalars) and 1xn or nx1 (vectors).

- Defining a vector
- Accessing elements within a vector
- Basic operations on vectors
- Go to link:
  - http://www.cyclismo.org/tutorial/matlab/vector.html
  - http://www.cyclismo.org/tutorial/matlab/operations.html

Introduction to Matrices in Matlab

- Defining Matrices
- Matrix Functions
- Matrix Operations
- Go to link
  - http://www.cyclismo.org/tutorial/matlab/matrix.html
- The colon operator
  - 1:10 is a row vector containing the integers from 1 to 10: 1 2 3 4 5 6 7 8 9 10
  - To obtain nonunit spacing, specify an increment.
  - For example, 100:-7:50 is 100 93 86 79 72 65 58 51
  - Subscript expressions involving colons refer to portions of a matrix.
  - A(1:k,j) is the first k elements of the jth column of A.
  - A(:,j) is the jth column of A.
  - A(i,:) is the ith row of the A.

Plotting

- MATLAB supports a number of plot types with the ability to create custom graphs.
- The simplest x-y type plot can be created using ***plot(x,y)*** where x and y are vectors with a list of values stored in them.
- Other plot commands including:
  - ***loglog, semilogx, semilogy, bar, stem, polar, plot3, contour, mesh*** **and** ***surf***.
- To find out about the syntax for any of the plot commands you can type **help <commandname >** inside of MATLAB.
- The figure plotted will be shown in the *Figure Window*.

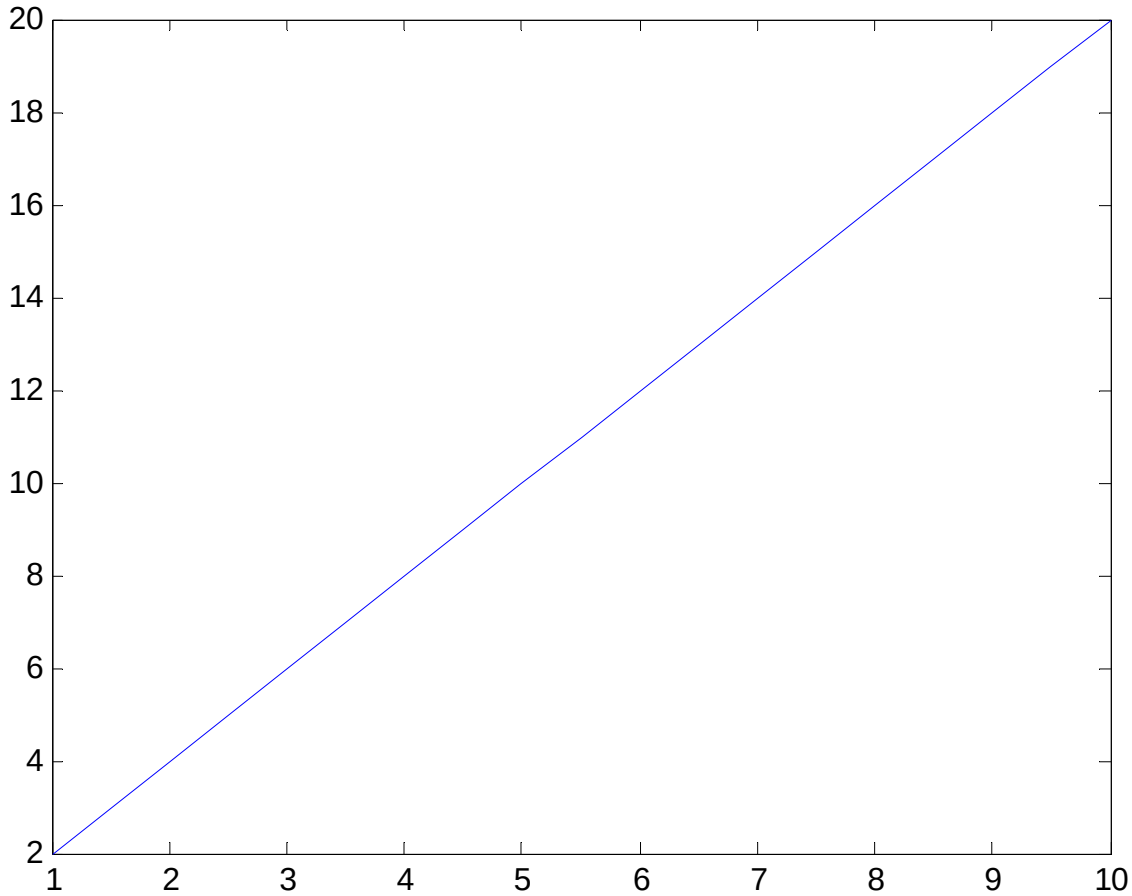
Plotting (2)

- Example 1: Using the colon to create lists and generating a plot
  - **`>>x=[1:.5:10]`**

    Defines a vector x that goes from 1 to 10 by 0.5. The colon is used to both create a list using the syntax [lower:increment:upper] and to select specific portions of a matrix.
  - **`>> y=2*x`**

    Defines a vector y with values equal to twice the values in x.
  - **`>>plot(x,y)`**

    Creates a simple x-y plot of the data stored in the vectors x and y.


20
18
16
14
12
10
8
6
4
2
1
2
3
4
5
6
7
8
9
10

Plotting (3)

- Adding text, setting scales and styles
  - **`>> axis([xmin xmax ymin ymax])`**

    Sets the x and y scales to the values you specify
  - **`>> title('title text')`**

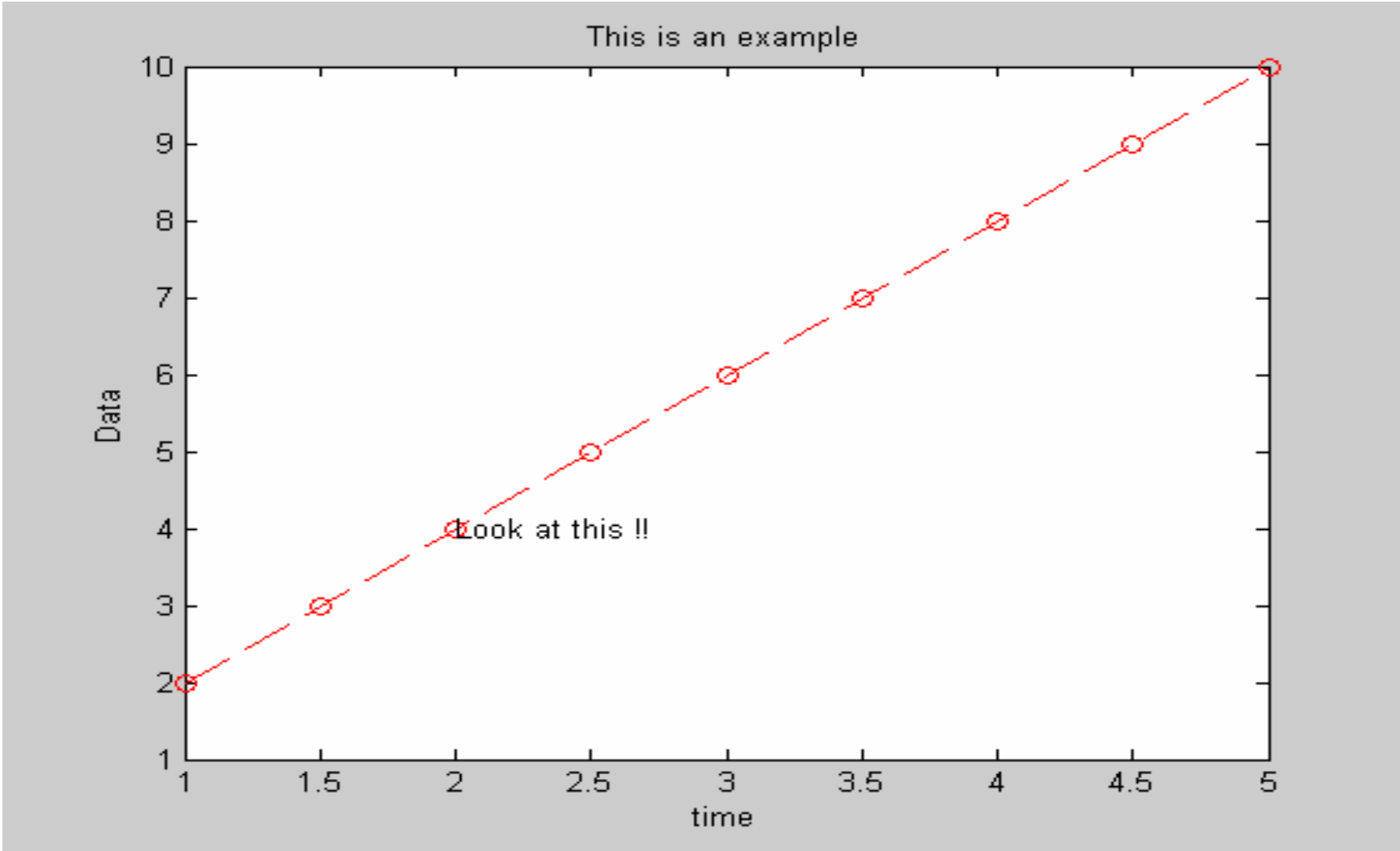
    Places a title above the plot. The commands xlabel('xtitle text') and ylabel('ytitle text') place a titles along the x and y- axes, respectively.
  - **`>>text(x,y,'your text')`**

    Places any text string at the graph coordinates (x, y)

Plotting (4)

- Styles including color and line type can be specified by using a 2 or 3 character string inside the plot command immediately following the **y** variable. For example, the command **`plot(x,y,'r--')`** will produce a red dashed line. The available color and line type variables are given below:
  - **For color:** **`y`** yellow;**`m`** magenta;**`c`** cyan;**`r`** red;**`g`** green;**`b`** blue;**`w`** white;**`b`** black
  - **For line type:** **`.`** point;**`o`** circle;**`x`** x-mark;**`+`** plus;**`*`** star; **`-`** solid;**`:`** dotted;**`-.`** dashdot;**`--`** dashed
- Plotting more than one data set on an a single axis can be accomplished by using the command **hold** on and then plotting the additional data followed by a hold off command.

Plotting (5)

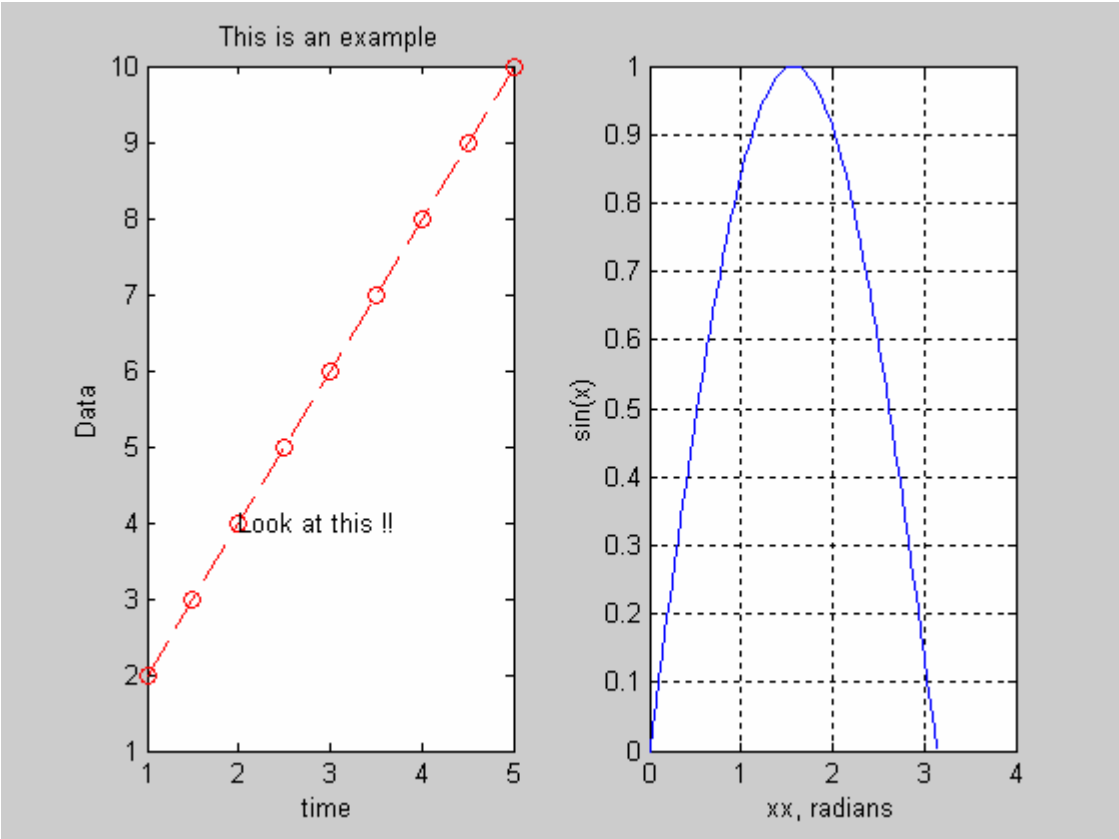

This is an example
Look at this !!
Data
time
10
9
8
7
6
5
4
3
2
1
1
1.5
2
2.5
3
3.5
4
4.5
5

Plotting (6)

- Example2: simple plots
  - The following sequence of commands plots the graph of the sine function between **0** and π, provided that the two arrays have the same number of elements.

```
>> xx = 0:pi/90: pi;
>> yy = sin(xx);
>> plot(xx, yy)
>> grid on
>> xlabel('xx, radians')
>> ylabel('sin(xx)')
```

- Example3: subplot


This is an example
Look at this !!
Data
time
sin(x)
xx, radians

MATLAB Programming

- Loops
  - For Loops
  - While Loops
  - Go to link:
    - http://www.cyclismo.org/tutorial/matlab/control.html
- If
  - Go to link:
    - http://www.cyclismo.org/tutorial/matlab/if.html
- Subroutines (optional)
  - Go to link:
    - http://www.cyclismo.org/tutorial/matlab/subroutine.html

What is SIMULINK?

- **SIMULINK** is an extension to **MATLAB** which uses a icon-driven interface for the construction of a block diagram representation of a process.
- Simulink encourages you to try things out.
- A block diagram is simply a graphical representation of a process (which is composed of an input, the system, and an output).

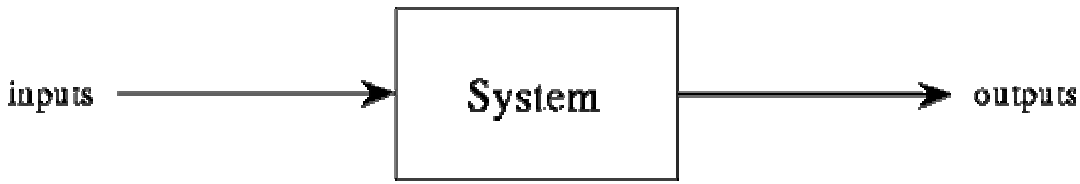


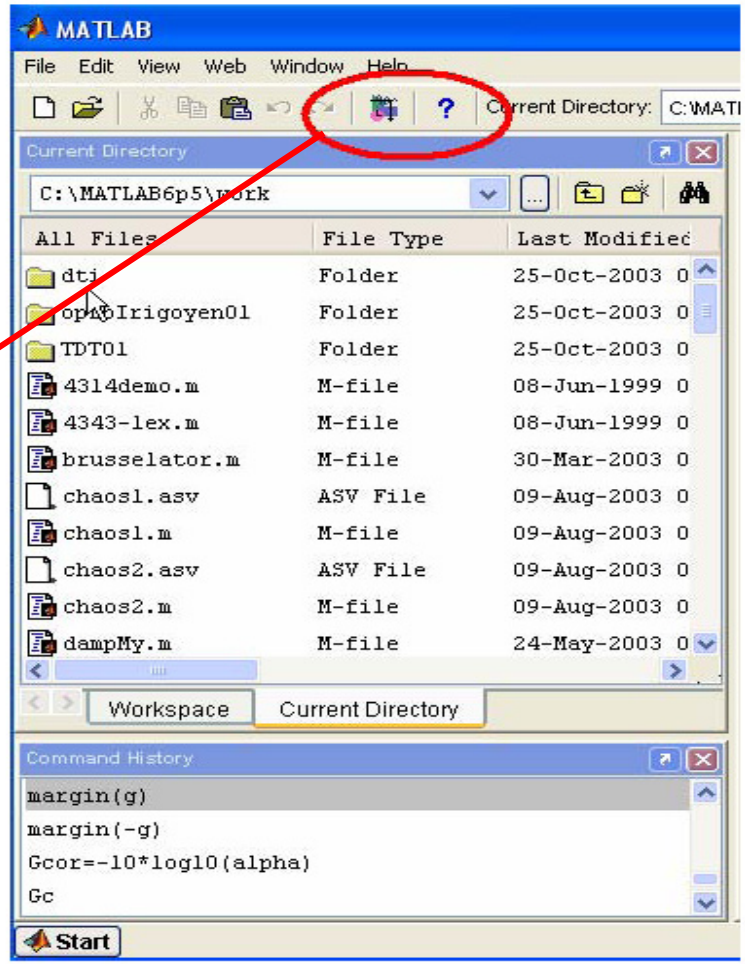
Figure 1: A VERY simple block diagram of a process.

About SIMULINK

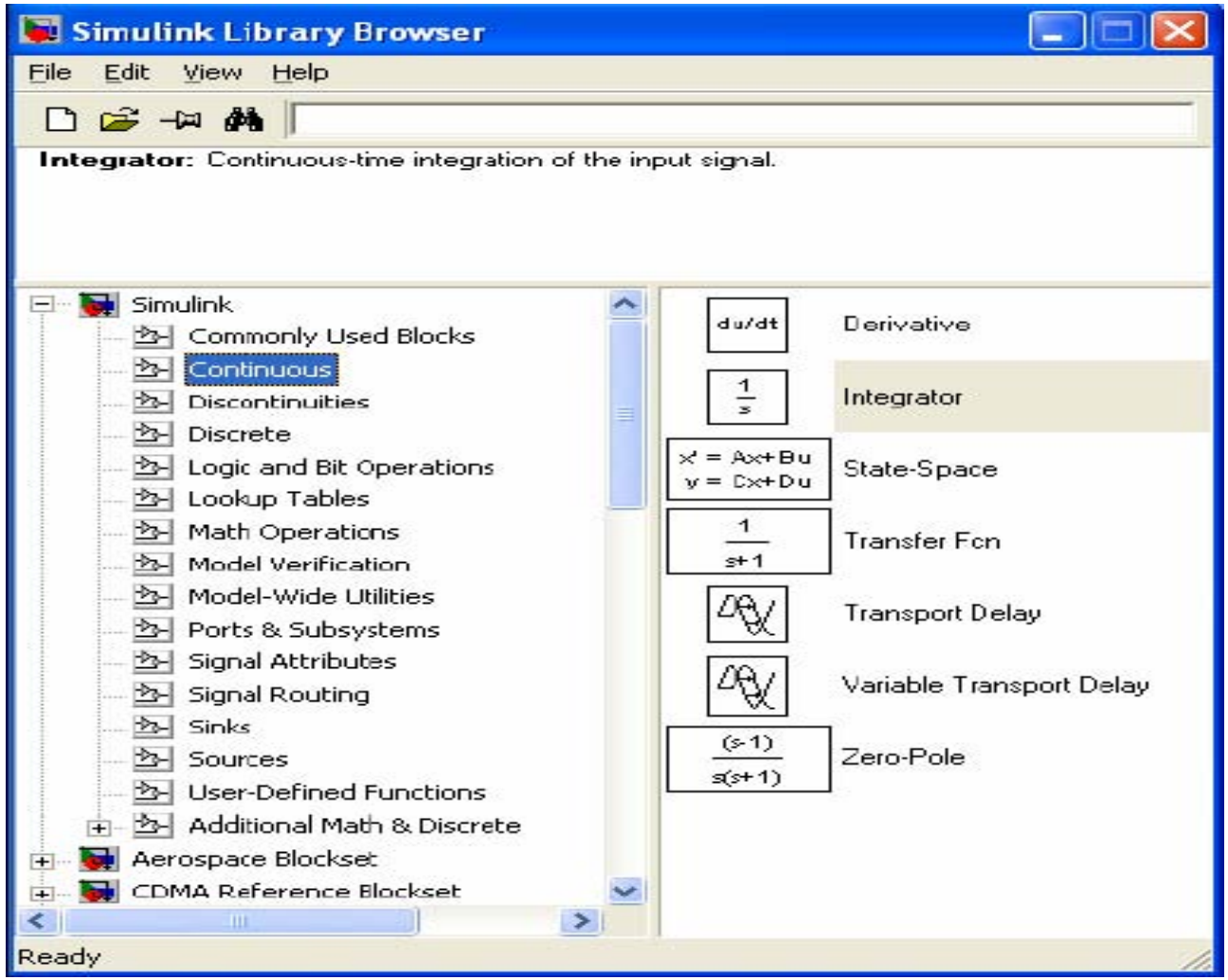
- SIMULINK uses a graphical user interface (GUI) for solving process simulations.
- Instead of writing MATLAB code, we simply connect the necessary ``icons'' together so as to construct the block diagram.
- The ``icons'' represent possible inputs to the system, parts of the systems, or outputs of the system.
- SIMULINK allows the user to easily simulate systems of *linear* and *nonlinear* ordinary differential equations.

Starting SIMULINK

- To start Simulink, on the MATLAB command prompt, type
- `>>simulink`
- Or
- Click here



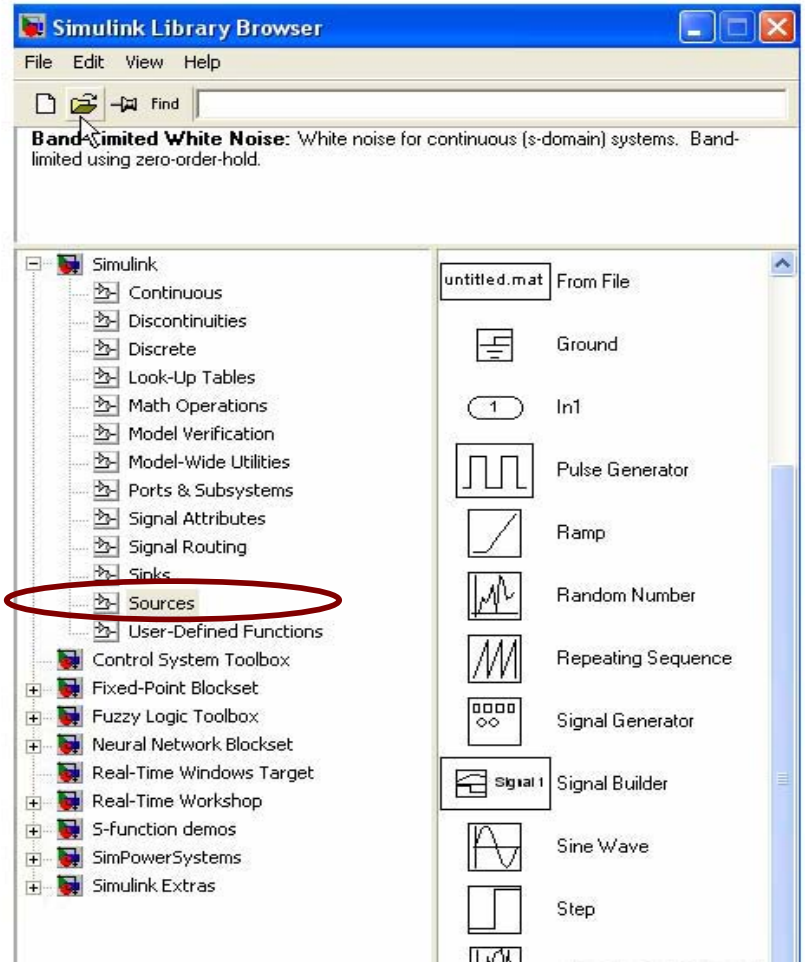
“Simulink Library Browser” will open.



Block Diagram Construction

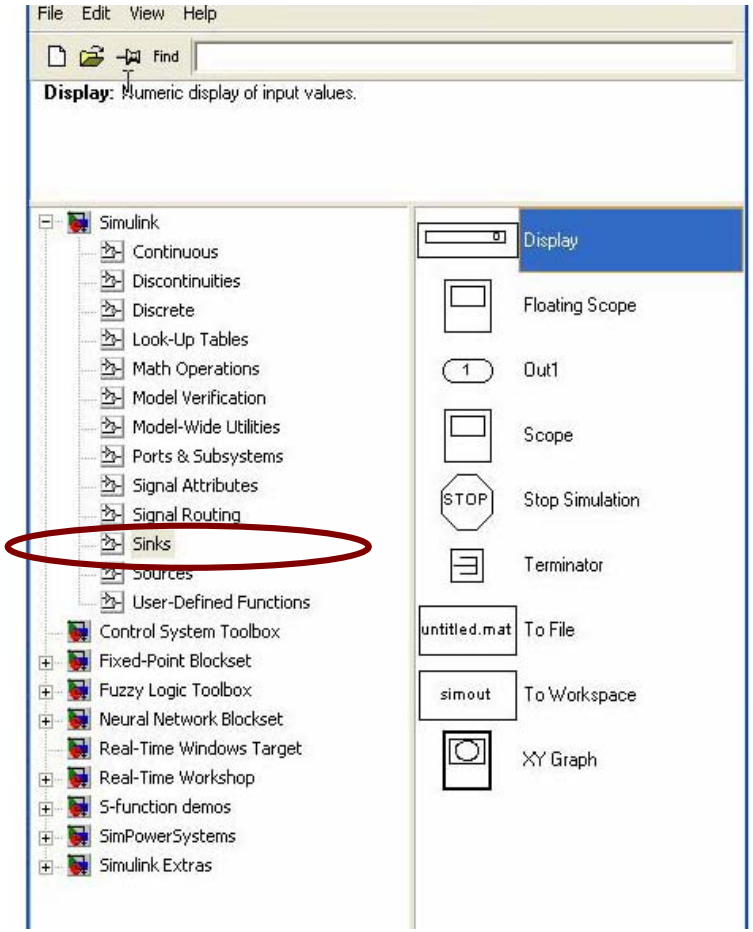
- Basically, one has to specify the ***model*** of the system (state space, discrete, transfer functions, nonlinear ODE's, etc), the ***input (source)*** to the system, and where ***the output (sink)*** of the simulation of the system will go.
- Open up the Sources, Sinks, and Linear windows by clicking on the appropriate icons.
  - Note the different types of sources (step function, sinusoidal, white noise, etc.), sinks (scope, file, workspace), and linear systems (transfer function, state space model, etc.).
- The next step is to connect these icons together by drawing lines connecting the icons using the left-most mouse button (hold the button down and drag the mouse to draw a line).

Sources



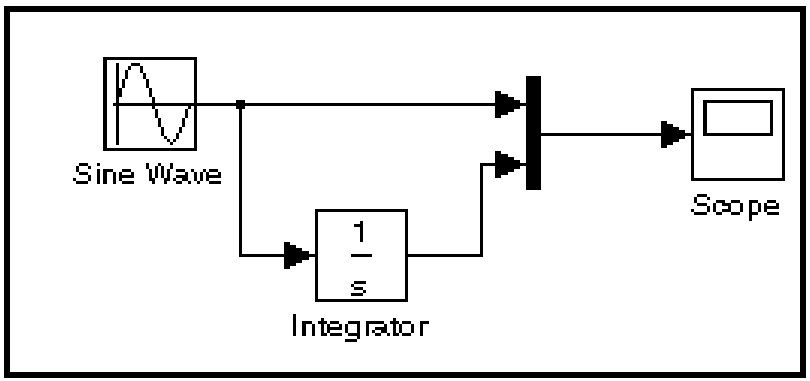
- Sources library contains the sources of data signals to be used in the dynamic system simulation.
- E.g. Constant signal, signal generator, sinusoidal waves, step input, repeating sequences like pulse trains and ramps etc.

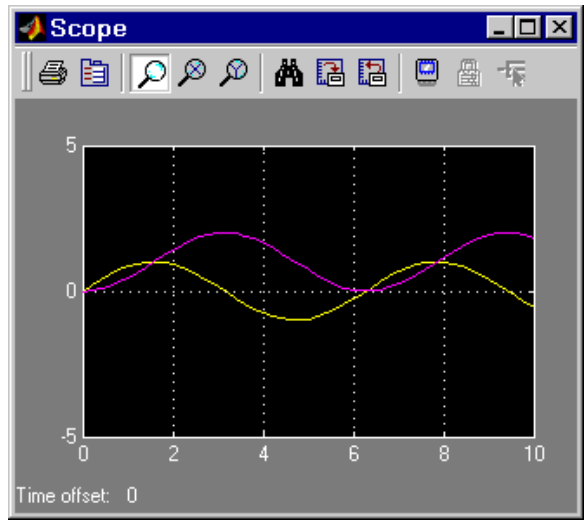
Sink



- Sinks library contains blocks where the signal terminates.
- You may store data in a file, display it.
- Use the terminator block to terminate unused signals.
- STOP block is used to stop the simulation if the input to the block is non-zero.

An Example

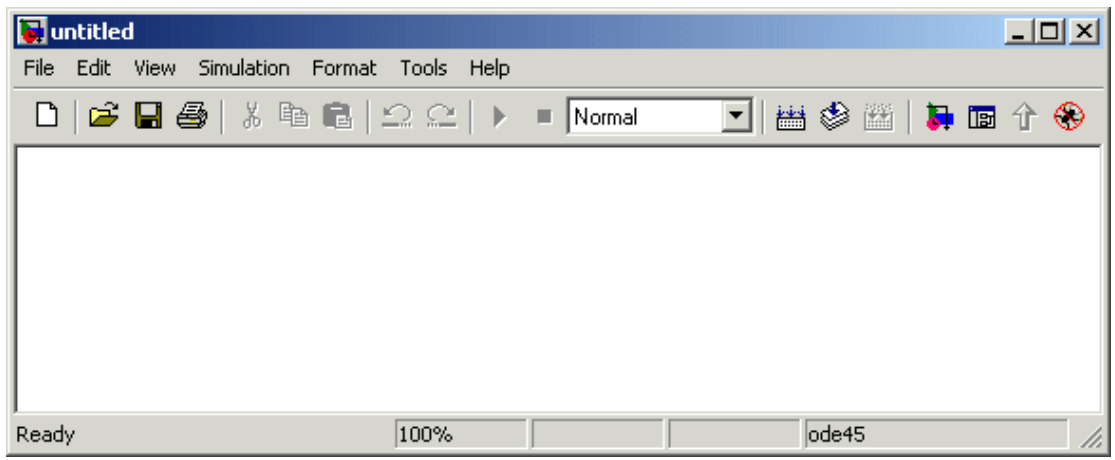

Sine Wave
1
s
Integrator
Scope


Scope
5
0
-5
0
2
4
6
8
10
Time offset: 0

Building this Model: Creating an Empty Model

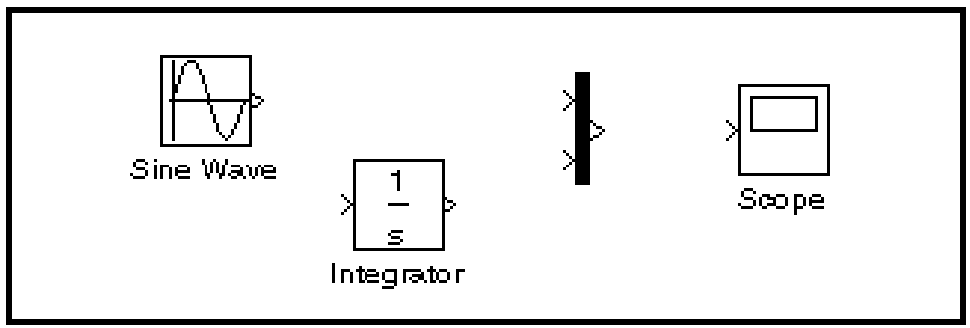
- click the **New Model** button on the Library Browser's toolbar.

- Simulink opens a new model window



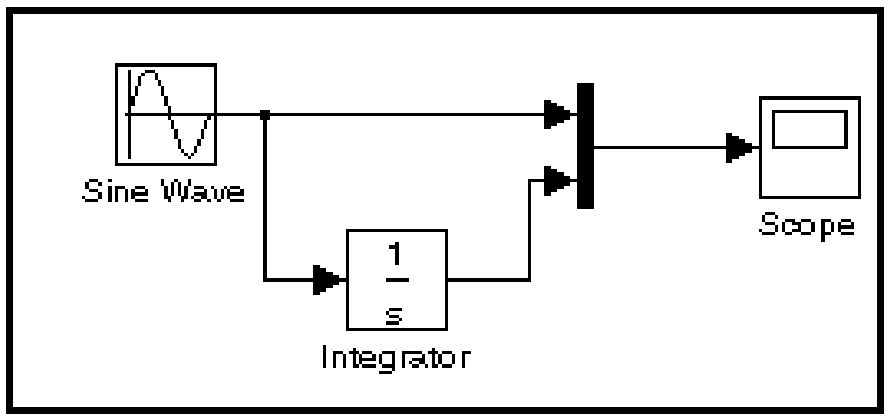
Building this Model: Adding Blocks

- To create this model, you need to copy blocks into the model from the following Simulink block libraries:
  - Sources library (the Sine Wave block)
  - Sinks library (the Scope block)
  - Continuous library (the Integrator block)
  - Signal & System library (the Mux block)



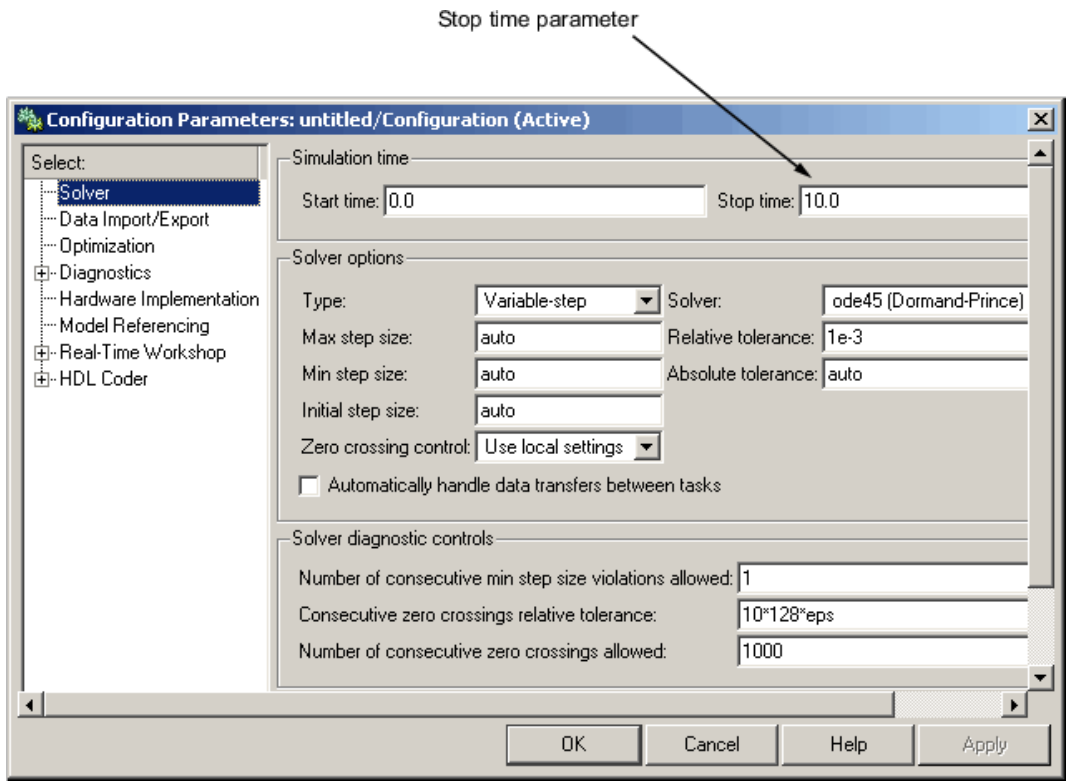
Building this Model: Connecting the Blocks

- First, position the pointer on the line between the Sine Wave and the Mux block.
- Press and hold down the Ctrl key (or click the right mouse button). Press the mouse button, then drag the pointer to the Integrator block's input port or over the Integrator block itself.
- Release the mouse button. Simulink draws a line between the starting point and the Integrator block's input port.
- Finish making block connections.



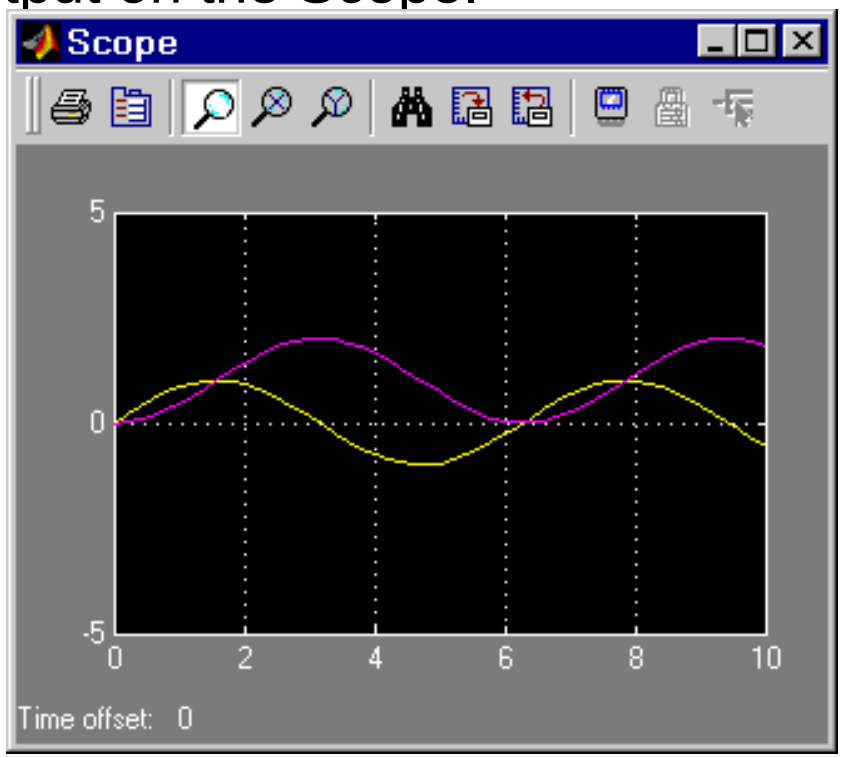
Building this Model: Configuring the Model

- Now set up Simulink to run the simulation for 10 seconds. (Simulation Parameters)



Building this Model: Running the Model

- Now **double-click** the Scope block to open its display window.
- Finally, choose Start from the Simulation menu and watch the simulation output on the Scope.



Reference: See the help of SIMULINK

Good Luck!!!!