



Høgskolen i Telemark

Telemark University College

Department of Electrical Engineering, Information Technology and Cybernetics

Tutorial

# Introduction to MATLAB

HANS-PETTER HALVORSEN, 2011.06.06

```
Command Window
To get started, select MATLAB Help or Demo from the Help menu.

This is a Classroom License for instructional use only.
Research and commercial use is prohibited.
>> A=[1 2;0 3]

A =

     1     2
     0     3

>>
```

```
Command History
end
plot(t,Y), grid
xlabel('Time 0 \leq t \leq 4')
title('Response in the output y(t) after unit step
-- 29.09.09 10:06 --
-- 29.09.09 11:04 --
-- 30.09.09 08:29 --
-- 09.10.09 15:34 --
A=[1 2;0 3]
-- 13.10.09 10:09 --
toolbox
help control system
help linear algebra
-- 14.10.09 09:17 --
A=[1 2;0 3]
```



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

MATLAB is a tool for technical computing, computation and visualization in an integrated environment. This document explains the basic concepts in MATLAB.

MATLAB is an abbreviation for MATrix LABoratory, so it is well suited for matrix manipulation and problem solving related to Linear Algebra.

MATLAB offers lots of additional Toolboxes for different areas such as Control Design, Image Processing, Digital Signal Processing, etc.

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

# Table of Contents

Preface.....	2
Table of Contents .....	iii
1 Introduction.....	1
1.1 Help .....	2
2 Start using MATLAB .....	4
2.1 The MATLAB Environment .....	4
2.1.1 Command Window.....	4
2.1.2 Command History.....	5
2.2 Variables .....	5
2.2.1 Workspace.....	6
2.2.2 Current Directory.....	7
2.3 Useful commands.....	8
3 Matrices and Vectors.....	9
3.1 Useful commands.....	11
4 Scripts and functions – M Files .....	12
4.1 Scripts .....	12
4.2 Functions .....	13
5 Flow Control.....	15
5.1 If-else Statement .....	15
5.2 Switch and Case Statement.....	15
5.3 For loop .....	16
5.4 While loop .....	16
6 Plotting.....	17

---

7	Linear Algebra.....	19
7.1	Vectors.....	19
7.2	Matrices.....	20
7.2.1	Transpose .....	20
7.2.2	Diagonal.....	20
7.2.3	Triangular.....	21
7.2.4	Matrix Multiplication.....	21
7.2.5	Matrix Addition.....	22
7.2.6	Determinant .....	22
7.2.7	Inverse Matrices .....	23
7.3	Eigenvalues.....	24
7.4	Solving Linear Equations.....	24
7.5	LU factorization .....	26
7.6	The Singular Value Decomposition (SVD).....	27
7.7	Commands.....	27
8	Toolboxes.....	28
9	Whats Next? .....	29
	Quick Reference .....	31
9.1	General .....	31
9.2	Matrices.....	31
9.3	Linear Algebra.....	31

---

# 1 Introduction

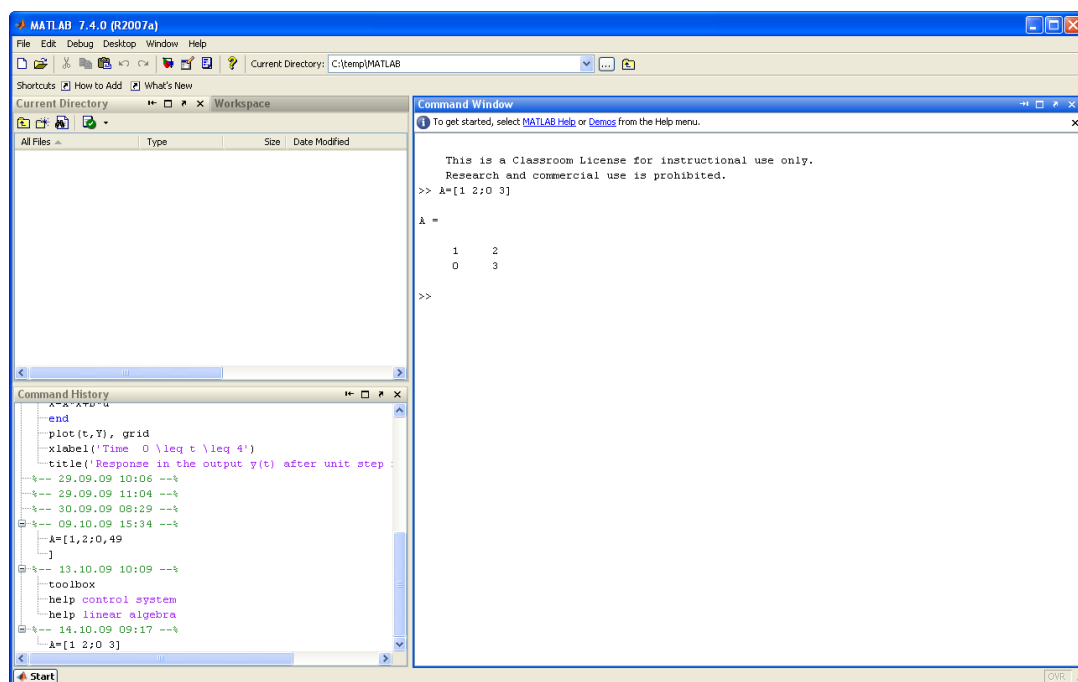
MATLAB is a tool for technical computing, computation and visualization in an integrated environment, e.g.,

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is developed by The MathWorks. MATLAB is a short-term for MATrix LABoratory. MATLAB is in use world-wide by researchers and universities.

For more information, see [www.mathworks.com](http://www.mathworks.com)

Below we see the MATLAB Environment:



MATLAB has the following windows:

- Command Window
- Command History

- Workspace
- Current Directory

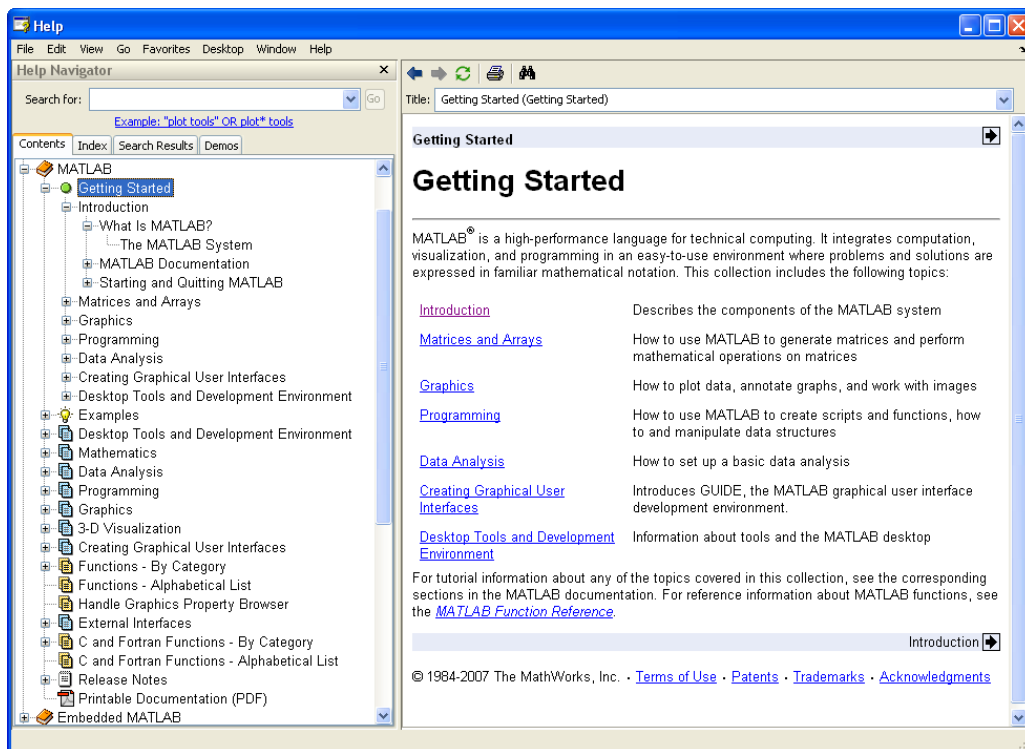
The **Command** window is the main window. Use the Command Window to enter variables and to run functions and M-files scripts (more about m-files later).

Watch the following “Getting Started with MATLAB” video:

<http://www.mathworks.com/demos/matlab/getting-started-with-matlab-video-tutorial.html>

## 1.1 Help

MATLAB has a comprehensive Help system.



You may also type help in your command window

```
>>help
```

Or more specific, e.g.,

```
>>help plot
```

I advise you to test all the examples in this text in MATLAB in order to get familiar with the program and its syntax. All examples in the text are outlined in a frame like this:

```
>>  
...
```

---

# 2 Start using MATLAB

This chapter explains the basic concepts in MATLAB.

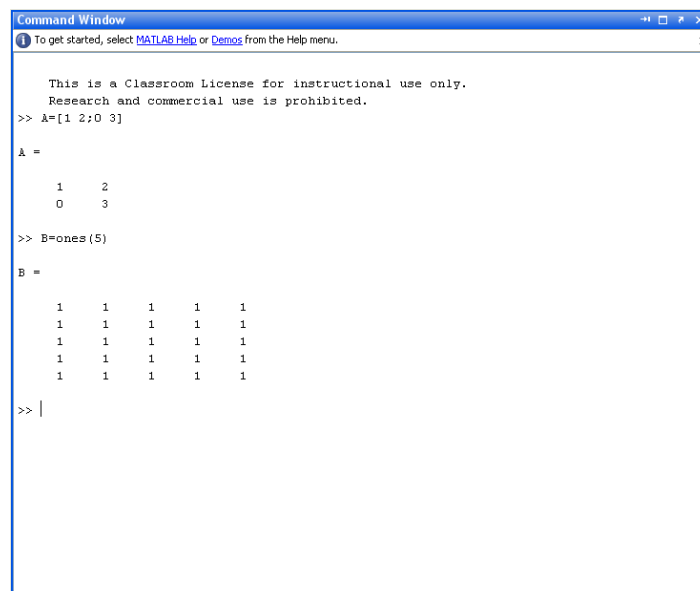
The topics are as follows:

- The MATLAB Environment
  - Command Window
  - Command History
  - Workspace
  - Current Directory
- Variables

## 2.1 The MATLAB Environment

### 2.1.1 Command Window

The **Command** window is the main window. Use the Command Window to enter variables and to run functions and M-files scripts (more about m-files later).



```
Command Window
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.
>> A=[1 2;0 3]

A =

     1     2
     0     3

>> B=ones(5)

B =

     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1

>> |
```

You type all your commands after the command Prompt “>>”, e.g., defining the following matrix

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}$$



The MATLAB syntax is as follows:

```
>> A = [1 2;0 3]
```

Or

```
>> A = [1,2;0,3]
```

If you, for an example, want to find the answer to

$$a + b, \text{ where } a = 4, b = 3$$

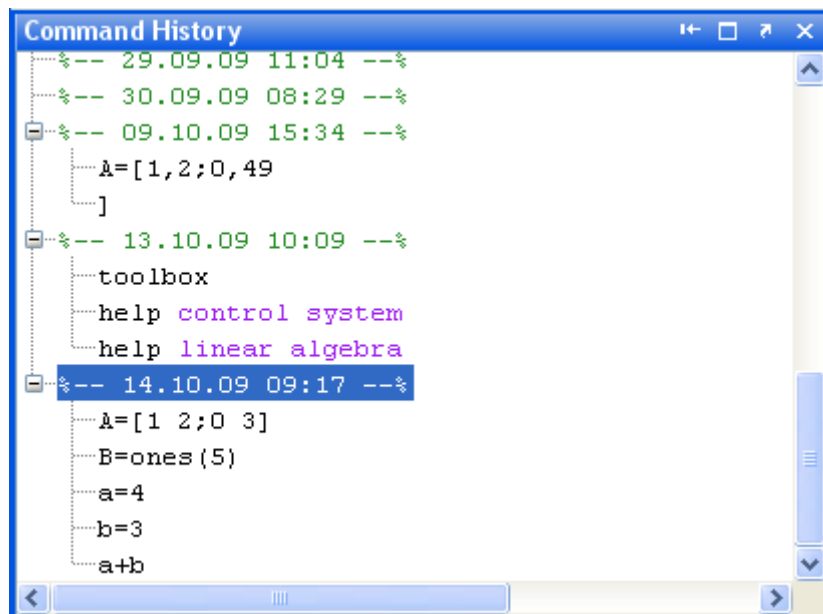
```
>>a=4  
>>b=3  
>>a+b
```

MATLAB then responds:

```
ans =  
      7
```

## 2.1.2 Command History

Statements you enter in the Command Window are logged in the Command History. From the Command History, you can view and search for previously run statements, as well as copy and execute selected statements. You can also create an M-file from selected statements.



## 2.2 Variables

Variables are defined with the assignment operator, “=”. MATLAB is dynamically typed, meaning that variables can be assigned without declaring their type, and that their type can change. Values can come from constants, from computation involving values of other variables, or from the output of a function. For example:

```
>> x = 17
x =
    17
>> x = 'hat'
x =
    hat
>> x = [3*4, pi/2]
x =
    12.0000    1.5708
>> y = 3*sin(x)
y =
   -1.6097    3.0000
```

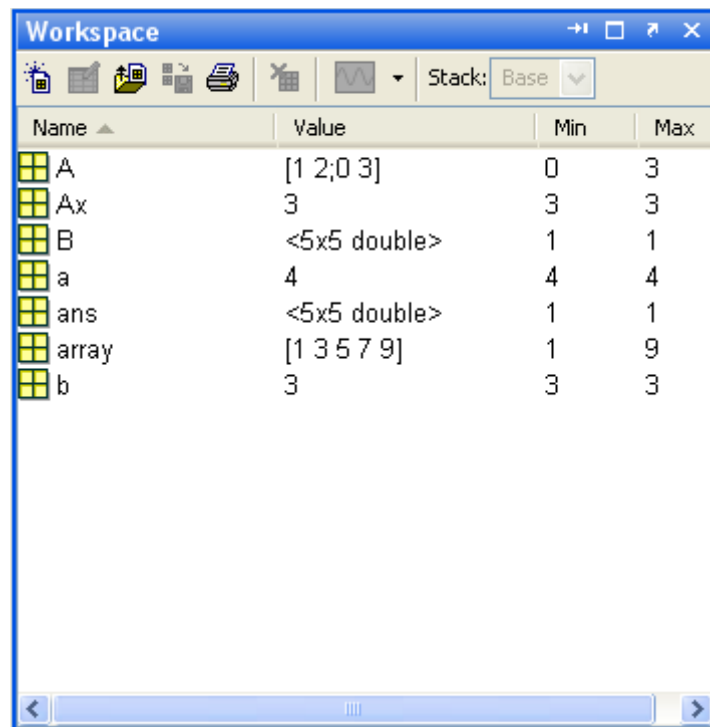
Unlike many other languages, where the semicolon is used to terminate commands, in MATLAB the semicolon serves to suppress the output of the line that it concludes.

```
>> a=5
a =
     5
>> a=6;
>>
```

As you see, when you type a semicolon (;) after the command, MATLAB will not respond.

## 2.2.1 Workspace

The Workspace window list all your variables used as long you have MATLAB opened.



You could also use the following command

```
>>who
```

This command list all the commands used

or

```
>>whos
```

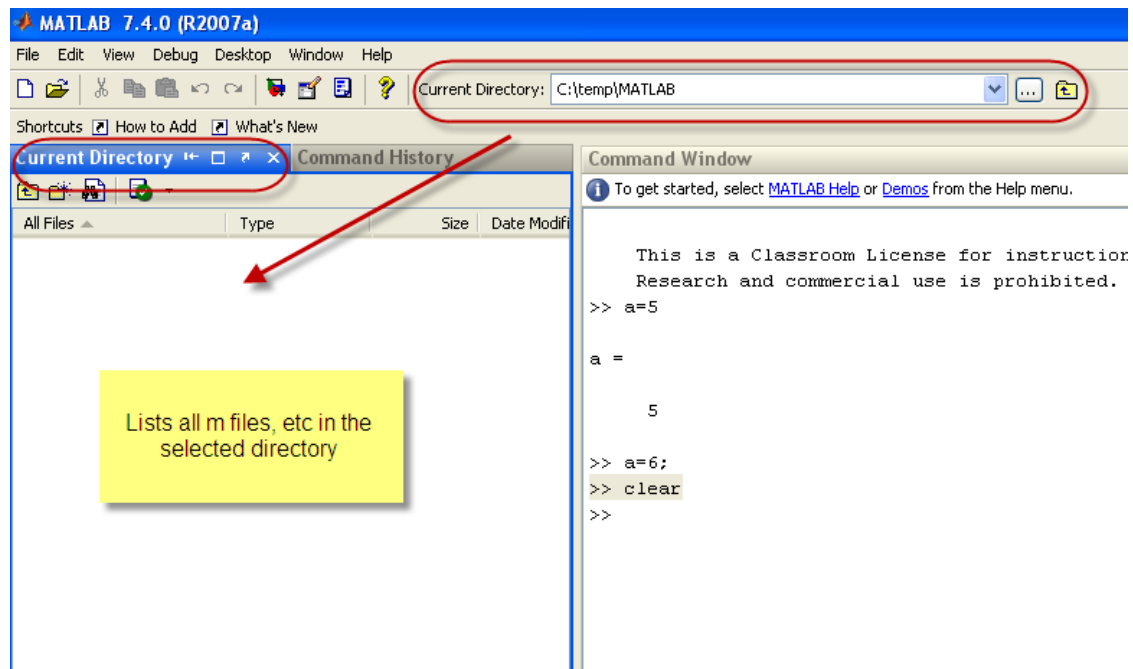
This command lists all the command with the current values, dimensions, etc.

The command `clear`, will clear all the variables.

```
>>clear
```

## 2.2.2 Current Directory

The Current Directory window list al m files, etc.



## 2.3 Useful commands

Here are some useful commands:

Command	Description
<code>help</code>	Help
<code>help x</code>	Gives you help on subject "x"
<code>who, whos</code>	Get list of variables
<code>clear</code>	Clears all variables in the Workspace
<code>clear x</code>	Clears the variable x
<code>what</code>	List all m files in the Working Folder

# 3 Matrices and Vectors

This chapter explains the basic concepts of using vectors and matrices in MATLAB.

Topics:

- Matrices and Vectors syntax
- Matrices functions

MATLAB is a "Matrix Laboratory", and as such it provides many convenient ways for creating vectors, matrices, and multi-dimensional arrays. In the MATLAB, a vector refers to a one dimensional ( $1 \times N$  or  $N \times 1$ ) matrix, commonly referred to as an array in other programming languages. A matrix generally refers to a 2-dimensional array, i.e. an  $m \times n$  array where  $m$  and  $n$  are greater than or equal to 1. Arrays with more than two dimensions are referred to as multidimensional arrays.

MATLAB provides a simple way to define simple arrays using the syntax: "**init:increment:terminator**". For instance:

```
>> array = 1:2:9
array =
 1 3 5 7 9
```

defines a variable named array (or assigns a new value to an existing variable with the name array) which is an array consisting of the values 1, 3, 5, 7, and 9. That is, the array starts at 1 (the init value), increments with each step from the previous value by 2 (the increment value), and stops once it reaches (or to avoid exceeding) 9 (the terminator value).

The increment value can actually be left out of this syntax (along with one of the colons), to use a default value of 1.

```
>> ari = 1:5
ari =
 1 2 3 4 5
```

assigns to the variable named ari an array with the values 1, 2, 3, 4, and 5, since the default value of 1 is used as the incrementer.

Note that the indexing is one-based, which is the usual convention for matrices in mathematics. This is atypical for programming languages, whose arrays more often start with zero.

Matrices can be defined by separating the elements of a row with blank space or comma and using a semicolon to terminate each row. The list of elements should be surrounded by square brackets: []. Parentheses: () are used to access elements and subarrays (they are also used to denote a function argument list).

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
>> A(2,3)
ans =
    11
```

Sets of indices can be specified by expressions such as "2:4", which evaluates to [2, 3, 4]. For example, a submatrix taken from rows 2 through 4 and columns 3 through 4 can be written as:

```
>> A(2:4,3:4)
ans =
    11     8
     7    12
    14     1
```

A square identity matrix of size  $n$  can be generated using the function `eye`, and matrices of any size with zeros or ones can be generated with the functions `zeros` and `ones`, respectively.

```
>> eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
>> zeros(2,3)
ans =
     0     0     0
     0     0     0
>> ones(2,3)
ans =
     1     1     1
     1     1     1
```

Most MATLAB functions can accept matrices and will apply themselves to each element. For example, `mod(2*J,n)` will multiply every element in "J" by 2, and then reduce each element modulo "n". MATLAB does include standard "for" and "while" loops, but using MATLAB's vectorized notation often produces code that is easier to read and faster to execute. This code, excerpted from the function `magic.m`, creates a magic square M for odd values of  $n$  (the built-in MATLAB function `meshgrid` is used here to generate square matrices I and J containing 1:n).

```
[J,I] = meshgrid(1:n);
A = mod(I+J-(n+3)/2,n);
B = mod(I+2*J-2,n);
M = n*A + B + 1;
```

## 3.1 Useful commands

Here are some useful commands:

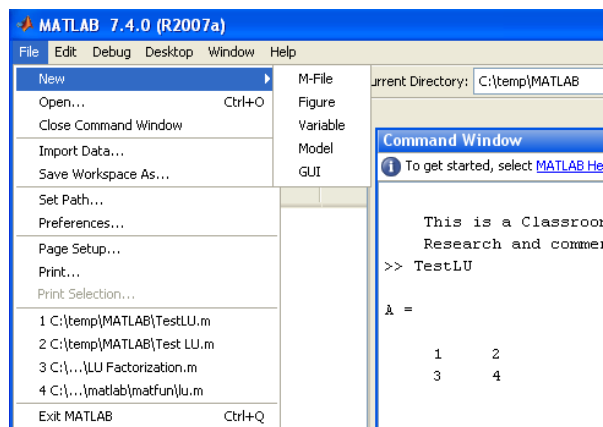
Command	Description
<code>eye(x)</code> , <code>eye(x,y)</code>	Identity matrix of order x
<code>ones(x)</code> , <code>ones(x,y)</code>	A matrix with only ones
<code>zeros(x)</code> , <code>zeros(x,y)</code>	A matrix with only zeros
<code>diag([x y z])</code>	Diagonal matrix
<code>size(A)</code>	Dimension of matrix A
<code>A'</code>	Inverse of matrix A

# 4 Scripts and functions – M Files

M-files are text files containing MATLAB code. Use the MATLAB Editor or another text editor to create a file containing the same statements you would type at the MATLAB command line. Save the file under a name that ends in “.m”.

## 4.1 Scripts

Create a new m-file from the File → New menu or the New button on the Toolbar.



The built-in Editor for creating and modifying m-files:



```

1 - A = [1 2; 3 4]
2 - [L, U, P] = lu(A)
3 - |

```

Running a m-file in the Command window:

Command Window

```

>> TestLU
A =
     1     2
     3     4

L =
  1.0000     0
  0.3333  1.0000

U =
  3.0000  4.0000
         0  0.6667

P =
     0     1
     1     0

>> |

```

Workspace

Name	Value	Min
A	[1 2; 3 4]	1
L	[1 0; 0.3333 1]	0
P	[0 1; 1 0]	0
U	[3 4; 0 0.6667]	0

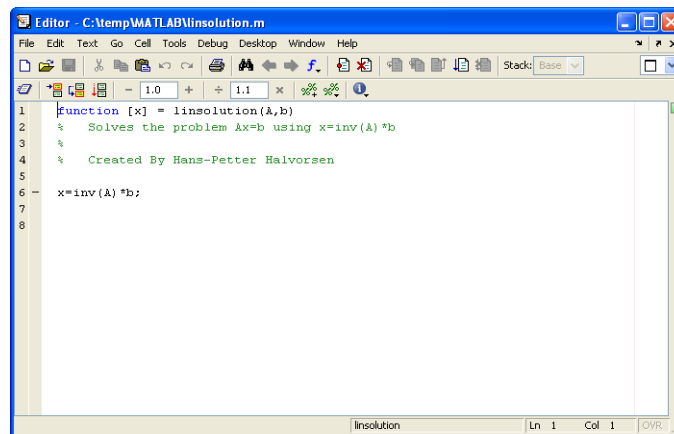
## 4.2 Functions

You may create your own functions and save them as a m-file.

### Example:

Create a function called “linsolution” which solve  $Ax = b \rightarrow x = A^{-1}b$

Below we see how the m-file for this function looks like:



```
Editor - C:\temp\MATLAB\linsolution.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1 function [x] = linsolution(A,b)
2 % Solves the problem Ax=b using x=inv(A)*b
3 %
4 % Created By Hans-Petter Halvorsen
5
6 x=inv(A)*b;
7
8
linsolution Ln 1 Col 1
```

You may define  $A$  and  $b$  in the Command window and then use the function in order to find  $x$ :

```
>> A=[1 2;3 4];
>> b=[5;6];
>> x = linsolution(A,b)
x =
-4.0000
 4.5000
```

After the function declaration (`function [x] = linsolution(A,b)`) in the m-file, you may write a description of the function. This is done with the Comment sign “%” before each line.

From the Command window you can then type “`help <function name>`” in order to read this information:

```
>> help linsolution
Solves the problem Ax=b using x=inv(A)*b
Created By Hans-Petter Halvorsen
```

# 5 Flow Control

This chapter explains the basic concepts of flow control in MATLAB.

The topics are as follows:

- If-else statement
- Switch and case statement
- For loop
- While loop

## 5.1 If-else Statement

The if statement evaluates a logical expression and executes a group of statements when the expression is true. The optional elseif and else keywords provide for the execution of alternate groups of statements. An end keyword, which matches the if, terminates the last group of statements. The groups of statements are delineated by the four keywords—no braces or brackets are involved.

**Example:**

```
n=5
if n > 2
    M = eye(n)
elseif n < 2
    M = zeros(n)
else
    M = ones(n)
end
```

## 5.2 Switch and Case Statement

The switch statement executes groups of statements based on the value of a variable or expression. The keywords case and otherwise delineate the groups. Only the first matching case is executed. There must always be an end to match the switch.

**Example:**

```
n=2
switch(n)
    case 1
```

```
M = eye(n)
case 2
    M = zeros(n)
case 3
    M = ones(n)
end
```

## 5.3 For loop

The for loop repeats a group of statements a fixed, predetermined number of times. A matching end delineates the statements.

### Example:

```
m=5
for n = 1:m
    r(n) = rank(magic(n));
end
r
```

## 5.4 While loop

The while loop repeats a group of statements an indefinite number of times under control of a logical condition. A matching end delineates the statements.

### Example:

```
m=5;
while m > 1
    m = m - 1;
    zeros(m)
end
```

# 6 Plotting

This chapter explains the basic concepts of creating plots in MATLAB.

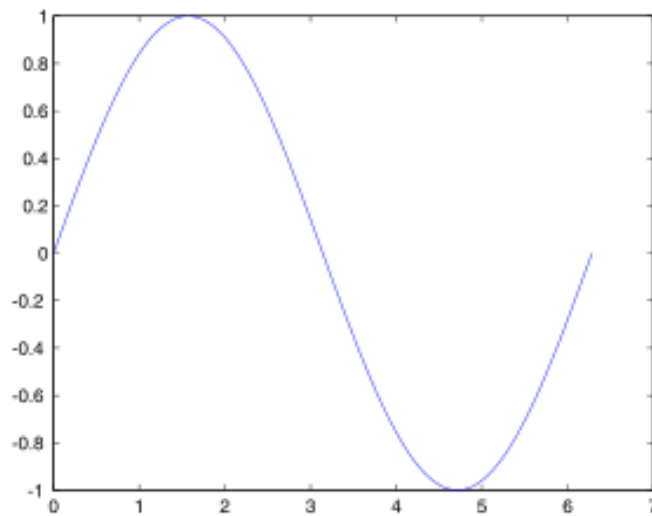
Topics:

- Basic Plot commands

Function plot can be used to produce a graph from two vectors  $x$  and  $y$ . The code:

```
x = 0:pi/100:2*pi;  
y = sin(x);  
plot(x,y)
```

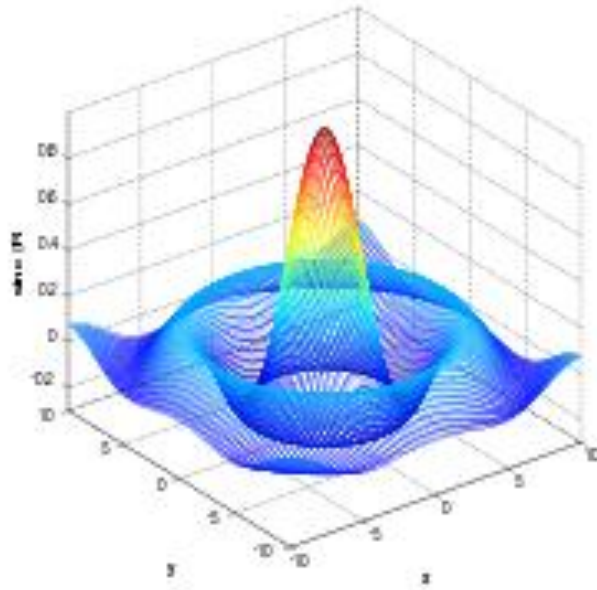
produces the following figure of the sine function:



Three-dimensional graphics can be produced using the functions `surf`, `plot3` or `mesh`.

```
[X,Y] = meshgrid(-10:0.25:10,-10:0.25:10);  
f = sinc(sqrt((X/pi).^2+(Y/pi).^2));  
mesh(X,Y,f);  
axis([-10 10 -10 10 -0.3 1])  
xlabel('\bfx')  
ylabel('\bfy')  
zlabel('\bfsinc ( {\bfR} )')  
hidden off
```

This code produces the following 3D plot:



# 7 Linear Algebra

Linear algebra is a branch of mathematics concerned with the study of matrices, vectors, vector spaces (also called linear spaces), linear maps (also called linear transformations), and systems of linear equations.

MATLAB are well suited for Linear Algebra.

## 7.1 Vectors

Given a vector  $x$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in R^n$$

**Example:**

$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```
>> x=[1; 2; 3]
x =
     1
     2
     3
```

The **Transpose** of vector  $x$ :

$$x^T = [x_1 \ x_2 \ \cdots \ x_n] \in R^{1 \times n}$$

```
>> x'
ans =
     1     2     3
```

The **Length** of vector  $x$ :

$$\|x\| = \sqrt{x^T x} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

**Orthogonality:**

$$x^T y = 0$$

## 7.2 Matrices

Given a matrix  $A$ :

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} \in R^{n \times m}$$

**Example:**

$$A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$$

```
>> A=[0 1;-2 -3]
A =
     0     1
    -2    -3
```

### 7.2.1 Transpose

The **Transpose** of matrix  $A$ :

$$A^T = \begin{bmatrix} a_{11} & \cdots & a_{n1} \\ \vdots & \ddots & \vdots \\ a_{1m} & \cdots & a_{nm} \end{bmatrix} \in R^{m \times n}$$

**Example:**

$$A^T = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}^T = \begin{bmatrix} 0 & -2 \\ 1 & -3 \end{bmatrix}$$

```
>> A'
ans =
     0    -2
     1    -3
```

### 7.2.2 Diagonal

The **Diagonal** elements of matrix  $A$  is the vector

$$\text{diag}(A) = \begin{bmatrix} a_{11} \\ a_{22} \\ \vdots \\ a_{pp} \end{bmatrix} \in R^{p=\min(x,m)}$$

**Example:**



```
>> diag(A)
ans =
     0
    -3
```

The **Diagonal** matrix  $\Lambda$  is given by:

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \in R^{n \times n}$$

Given the **Identity** matrix  $I$ :

$$I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \in R^{n \times m}$$

**Example:**

```
>> eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
```

## 7.2.3 Triangular

Lower Triangular matrix  $L$ :

$$L = \begin{bmatrix} \cdot & 0 & 0 \\ \vdots & \ddots & 0 \\ \cdot & \cdots & \cdot \end{bmatrix}$$

Upper Triangular matrix  $U$ :

$$U = \begin{bmatrix} \cdot & \cdots & \cdot \\ 0 & \ddots & \vdots \\ 0 & 0 & \cdot \end{bmatrix}$$

## 7.2.4 Matrix Multiplication

Given the matrices  $A \in R^{n \times m}$  and  $B \in R^{m \times p}$ , then

$$C = AB \in R^{n \times p}$$

where

$$c_{jk} = \sum_{l=1}^n a_{jl}b_{lk}$$

**Example:**

```
>> A=[0 1;-2 -3]
A =
     0     1
    -2    -3
>> B=[1 0;3 -2]
B =
     1     0
     3    -2
>> A*B
ans =
     3    -2
    -11     6
```

**Note!**

$$AB \neq BA$$

$$A(BC) = (AB)C$$

$$(A + B)C = AC + BC$$

$$C(A + B) = CA + CB$$

## 7.2.5 Matrix Addition

Given the matrices  $A \in R^{n \times m}$  and  $B \in R^{n \times m}$ , then

$$C = A + B \in R^{n \times m}$$

**Example:**

```
>> A=[0 1;-2 -3]
>> B=[1 0;3 -2]
>> A+B
ans =
     1     1
     1    -5
```

## 7.2.6 Determinant

Given a matrix  $A \in R^{n \times n}$ , then the **Determinant** is given:

$$\det(A) = |A|$$

Given a 2x2 matrix

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \in R^{2 \times 2}$$

Then

$$\det(A) = |A| = a_{11}a_{22} - a_{21}a_{12}$$

### Example:

```
A =
     0     1
    -2    -3
>> det(A)
ans =
     2
```

Notice that

$$\det(AB) = \det(A) \det(B)$$

and

$$\det(A^T) = \det(A)$$

### Example:

```
>> det(A*B)
ans =
    -4
>> det(A)*det(B)
ans =
    -4
>> det(A')
ans =
     2
>> det(A)
ans =
     2
```

## 7.2.7 Inverse Matrices

The **inverse** of a quadratic matrix  $A \in R^{n \times n}$  is defined by:

$$A^{-1}$$

if

$$AA^{-1} = A^{-1}A = I$$

For a 2x2 matrix we have:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \in R^{2 \times 2}$$

The inverse  $A^{-1}$  is given by

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \in R^{2 \times 2}$$

**Example:**

```
A =
     0     1
    -2    -3
>> inv(A)
ans =
   -1.5000   -0.5000
    1.0000     0
```

Notice that:

$$AA^{-1} = A^{-1}A = I$$

→ Prove this in MATLAB

## 7.3 Eigenvalues

Given  $A \in R^{n \times n}$ , then the Eigenvalues is defined as:

$$\det(\lambda I - A) = 0$$

**Example:**

```
A =
     0     1
    -2    -3
>> eig(A)
ans =
    -1
    -2
```

## 7.4 Solving Linear Equations

Given the linear equation

$$Ax = b$$

with the solution:

$$x = A^{-1}b$$

(Assuming that the inverse of  $A$  exists)

### Example:

The equations

$$\begin{aligned}x_1 + 2x_2 &= 5 \\ 3x_1 + 4x_2 &= 6\end{aligned}$$

may be written

$$Ax = b$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$$

where

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$b = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$$

The solution is:

```
A =
     1     2
     3     4
>> b=[5;6]
b =
     5
     6
>> x=inv(A)*b
x =
    -4.0000
     4.5000
```

In MATLAB you could also write “ $x=A \setminus b$ ”, which should give the same answer. This syntax can also be used when the inverse of  $A$  don't exists.

### Example:

```
>> A=[1 2;3 4;7 8]
>> x=inv(A)*b
??? Error using ==> inv
Matrix must be square.
>> x=A\b
x =
    -3.5000
     4.1786
```

## 7.5 LU factorization

LU factorization of  $A \in R^{n \times m}$  is given by

$$A = LU$$

where

$L$  is a lower triangular matrix

$U$  is a upper triangular matrix

The MATLAB syntax is `[L,U]=lu(A)`

### Example:

```
>> A=[1 2;3 4]
>> [L,U]=lu(A)
L =
    0.3333    1.0000
    1.0000         0
U =
    3.0000    4.0000
         0    0.6667
```

Or sometimes LU factorization of  $A \in R^{n \times m}$  is given by

$$A = LU = LDU$$

where

$D$  is a diagonal matrix

The MATLAB syntax is `[L,U,P]=lu(A)`

### Example:

```
>> A=[1 2;3 4]
A =
     1     2
     3     4
>> [L,U,P]=lu(A)
L =
```

```

1.0000    0
0.3333    1.0000
U =
3.0000    4.0000
      0    0.6667
P =
 0    1
 1    0

```

## 7.6 The Singular Value Decomposition (SVD)

The **Singular value Decomposition** (SVD) of the matrix  $A \in R^{n \times m}$  is given by

$$A = USV^T$$

where

$U$  is a orthogonal matrix

$V$  is a orthogonal matrix

$S$  is a diagonal singular matrix

**Example:**

```

>> A=[1 2;3 4];
>> [U,S,V] = svd(A)
U =
 -0.4046   -0.9145
 -0.9145    0.4046
S =
 5.4650     0
      0    0.3660
V =
 -0.5760    0.8174
 -0.8174   -0.5760

```

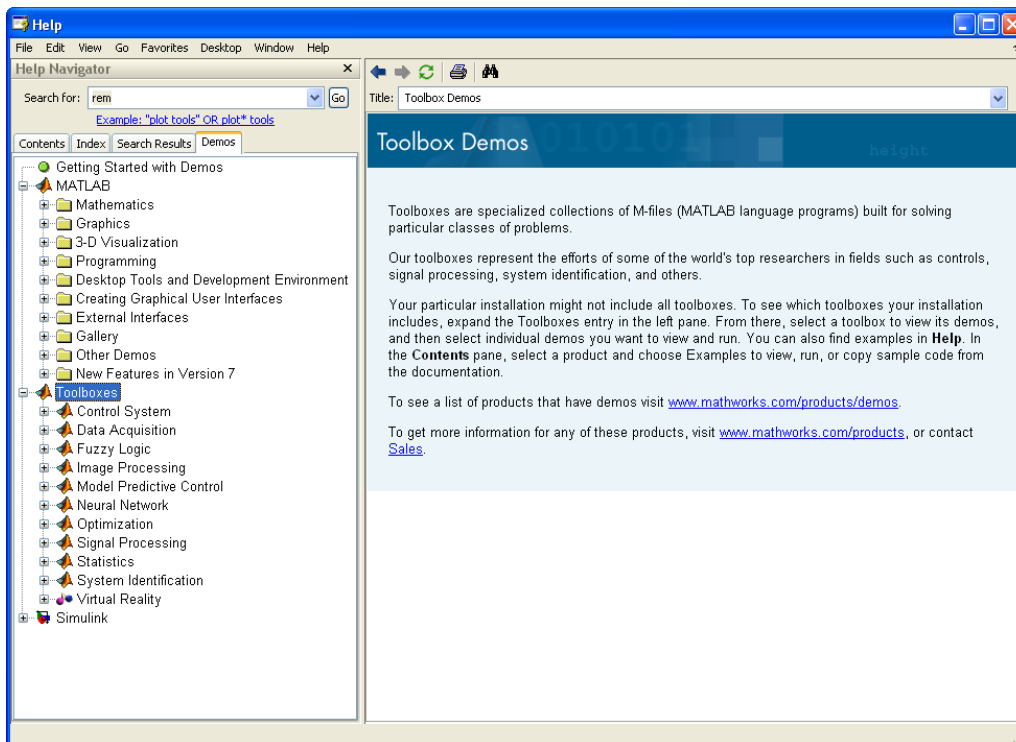
## 7.7 Commands

Command	Description
<code>[L,U]=lu(A)</code> <code>[L,U,P]=lu(A)</code>	LU Factorization
<code>[U,S,V] = svd(A)</code>	Singular Value Decomposition (SVD)

# 8 Toolboxes

Toolboxes are specialized collections of M-files built for solving particular classes of problems, e.g.,

- Control System Toolbox
- Signal Processing Toolbox
- Statistics Toolbox
- System identification Toolbox
- etc.





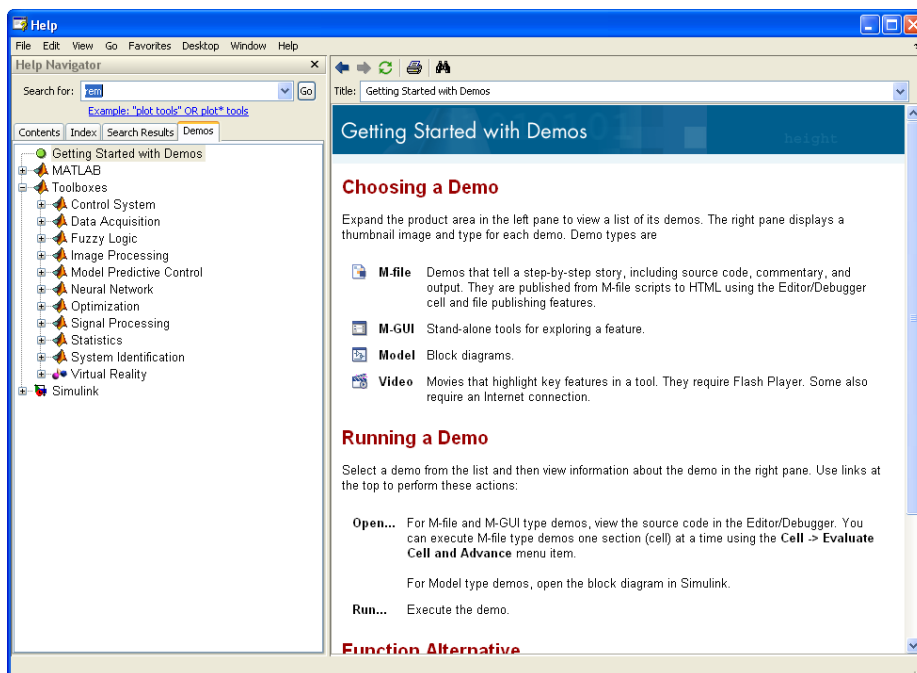
# 9Whats Next?

There are lots of useful resources to dig into if you want to learn more about MATLAB.

Type `demo` in the Command window in MATLAB

```
>>demo
```

In the “Getting Stated with Demos” in the MATLA Help system you get access to tons of Demos in form of M-files, Videos, etc.



For more information about MATLAB, see [www.mathworks.com](http://www.mathworks.com)

At the MathWorks home page there are lots of Documentation, Examples, Videos, Tips and Tricks, etc.

Another resource is the MATLAB documentation (may be downloaded as pdf files from [www.mathworks.com](http://www.mathworks.com)), such as:

- MATLAB Desktop Tools and Development Environment
- MATLAB Getting Started Guide
- MATLAB Function Reference
- MATLAB Mathematics

- MATLAB Graphics
- Etc.

# Quick Reference

## 9.1 General

Command	Description
<code>help</code>	Help
<code>help x</code>	Gives you help on subject "x"
<code>who, whos</code>	Get list of variables
<code>clear</code>	Clears all variables in the Workspace
<code>clear x</code>	Clears the variable x
<code>what</code>	List all m files in the Working Folder

## 9.2 Matrices

Command	Description
<code>eye(x), eye(x,y)</code>	Identity matrix of order x
<code>ones(x), ones(x,y)</code>	A matrix with only ones
<code>zeros(x), zeros(x,y)</code>	A matrix with only zeros
<code>diag([x y z])</code>	Diagonal matrix
<code>size(A)</code>	Dimension of matrix A
<code>A'</code>	Inverse of matrix A

## 9.3 Linear Algebra

Command	Description
<code>[L,U]=lu(A)</code>	LU Factorization

---

<code>[L,U,P]=lu(A)</code>	
<code>[U,S,V] = svd(A)</code>	Singular Value Decomposition (SVD)



**Høgskolen i Telemark**

**Telemark University College**

**Faculty of Technology**

**Kjølnes Ring 56**

**N-3914 Porsgrunn, Norway**

**[www.hit.no](http://www.hit.no)**

---

**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](mailto:hans.p.halvorsen@hit.no)**

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

**Room: B-237a**

---