Welcome to BME1901 Introductory Computer Sciences 2019-2020 Fall

- Instructor: Dr. Görkem SERBES (C317)
  gserbes@yildiz.edu.tr
  https://avesis.yildiz.edu.tr/gserbes/

- Lab Assistants:
  Doğan Onur ARISOY - *arisoy@yildiz.edu.tr*
  Nihat AKKAN - *nakkan@yildiz.edu.tr*
  Yeliz ERŞAN - *yelize@yildiz.edu.tr*

- Grading:

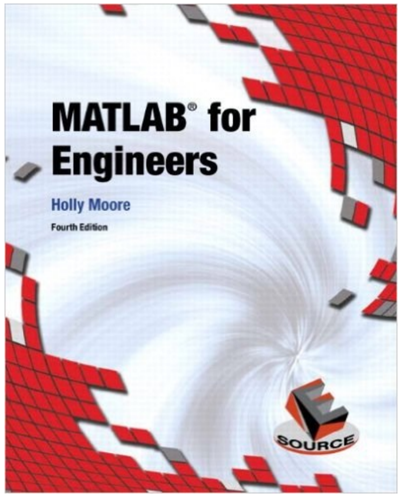
Midterm exams & Assignments & Quizzes	60%
Final exam	40%

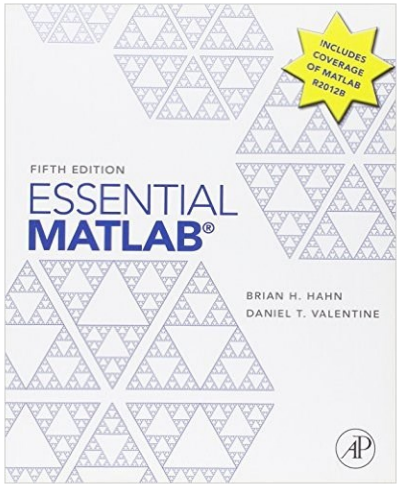
only individual submissions allowed!

Textbooks

- **H. Moore, MATLAB for Engineers, 4th edition, Pearson, 2011.**

- B. H. Hahn, D. T. Valentine, Essential MATLAB for Engineers and Scientists, 5th edition, Academic Press, MA, 2013.
- E. B. Magrab, S. Azarm, B. Balachandran, J. H. Duncan, K. E. Herold, G. C. Waish, An Engineer's Guide to MATLAB with Applications from Mechanical, Aerospace, Electrical, Civil and Biological Systems Engineering, 3rd edition, Printice Hall, 2011.
- A. Gilat, MATLAB, An Introduction with Applications, 4th edition, John Wiley & Sons, 2011.





Topics

- Introduction to MATLAB and MATLAB environment, Mathematical and logical operators, Vectors and matrices, Mathematical operations
- Built-in mathematical functions in MATLAB
- Program design and algorithm development
- Selection structures (“if” statement, “switch” statement)
- Repetition structures (“for” loop, “while” loop)
- m-files and m-functions
- Graphics in MATLAB
- Data Types and array types in MATLAB
- Reading and writing data from files, Symbolic programming
- Graphical user interface (GUI) in MATLAB
- Introduction to Simulink

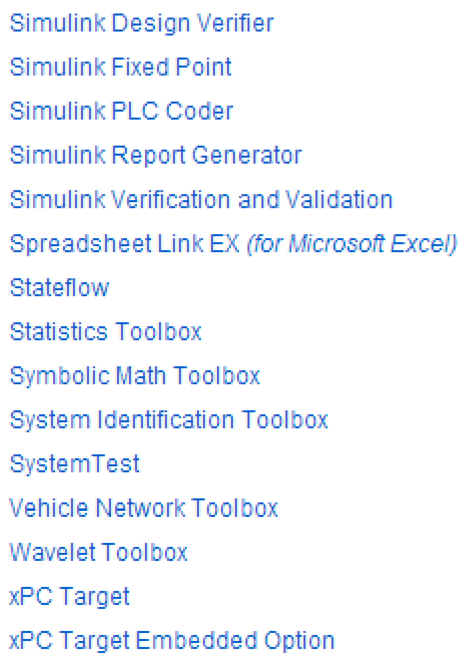
Introduction to MATLAB

- A computer is useful because it can do numerous computations quickly, so operating on large numerical data sets listed in tables as arrays or matrices of rows and columns is quite efficient.
- MATLAB is a powerful technical computing system for handling scientific and engineering calculations,
- The name MATLAB stands for « Matrix Laboratory »,
- One of the main properties of MATLAB is that it is used *interactively*.

MATLAB Products - 1

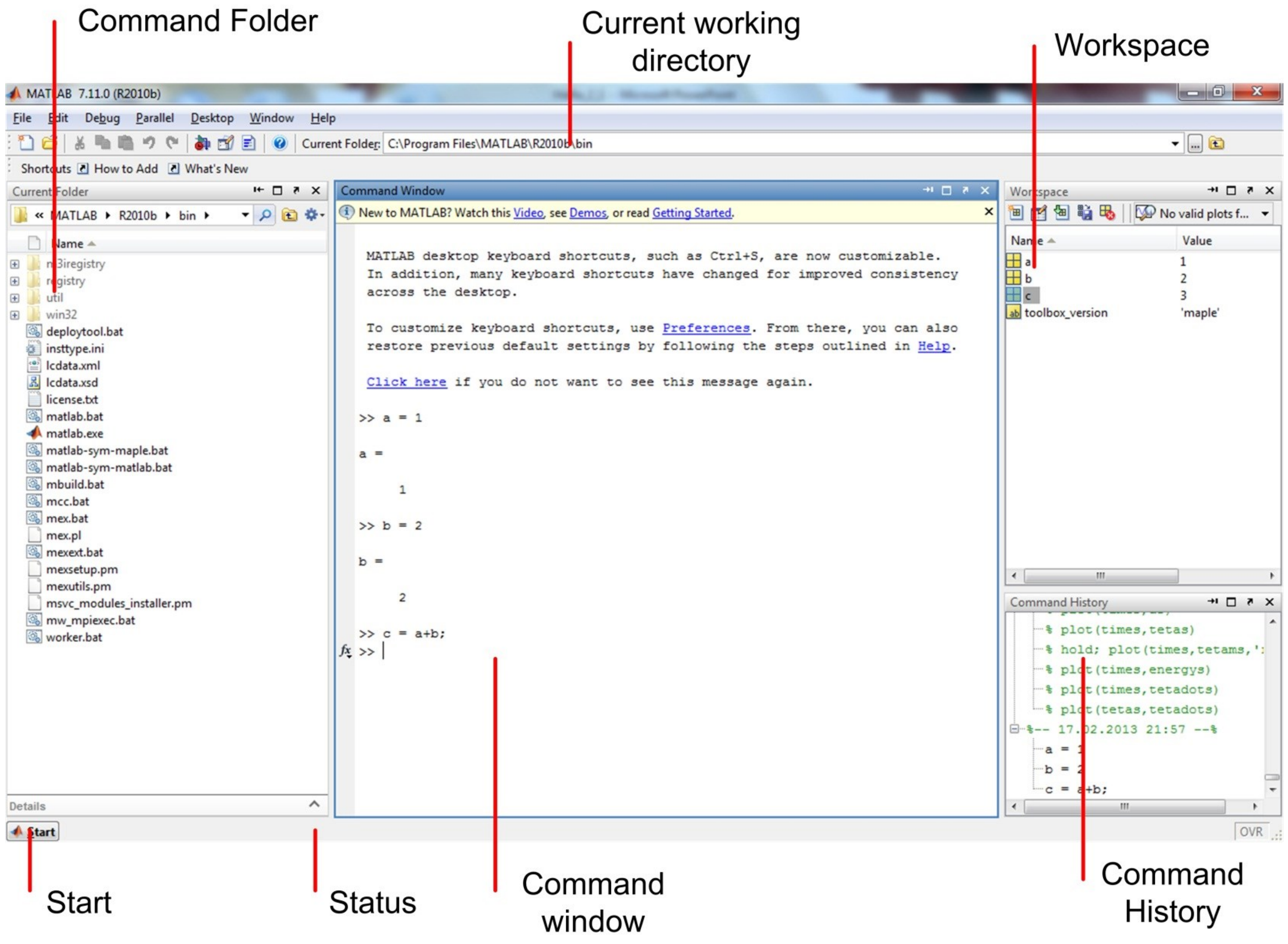
- Aerospace Blockset
- Aerospace Toolbox
- Bioinformatics Toolbox
- Communications System Toolbox
- Computer Vision System Toolbox
- Control System Toolbox
- Curve Fitting Toolbox
- Data Acquisition Toolbox
- Database Toolbox
- Datafeed Toolbox
- DO Qualification Kit *(for DO-178)*
- DSP System Toolbox
- Econometrics Toolbox
- Embedded Coder
- Filter Design HDL Coder
- Financial Instruments Toolbox
- Financial Toolbox
- Fixed-Point Toolbox
- Fuzzy Logic Toolbox
- Gauges Blockset
- Global Optimization Toolbox
- HDL Coder
- HDL Verifier
- IEC Certification Kit *(for ISO 26262 and IEC 61508)*
- Image Acquisition Toolbox
- Image Processing Toolbox
- Instrument Control Toolbox
- Mapping Toolbox
- MATLAB
- MATLAB Builder EX *(for Microsoft Excel)*
- MATLAB Builder JA *(for Java language)*
- MATLAB Builder NE *(for Microsoft .NET Framework)*
- MATLAB Coder
- MATLAB Compiler
- MATLAB Distributed Computing Server
- MATLAB Production Server
- MATLAB Report Generator
- Model Predictive Control Toolbox
- Model-Based Calibration Toolbox
- Neural Network Toolbox
- OPC Toolbox
- Optimization Toolbox
- Parallel Computing Toolbox
- Partial Differential Equation Toolbox
- Phased Array System Toolbox
- Polyspace Client for Ada
- Polyspace Client for C/C++
- Polyspace Model Link SL *(for Simulink)*
- Polyspace Model Link TL *(for dSPACE TargetLink)*
- Polyspace Server for Ada
- Polyspace Server for C/C++
- Polyspace UML Link RH *(for IBM Rational Rhapsody)*
- Real-Time Windows Target
- RF Toolbox
- Robust Control Toolbox
- Signal Processing Toolbox
- SimBiology
- SimDriveline
- SimElectronics
- SimEvents
- SimHydraulics
- SimMechanics
- SimPowerSystems
- SimRF
- Simscape
- Simulink
- Simulink 3D Animation
- Simulink Code Inspector
- Simulink Coder
- Simulink Control Design
- Simulink Design Optimization

MATLAB Products - 2



- Simulink Design Verifier
- Simulink Fixed Point
- Simulink PLC Coder
- Simulink Report Generator
- Simulink Verification and Validation
- Spreadsheet Link EX *(for Microsoft Excel)*
- Stateflow
- Statistics Toolbox
- Symbolic Math Toolbox
- System Identification Toolbox
- SystemTest
- Vehicle Network Toolbox
- Wavelet Toolbox
- xPC Target
- xPC Target Embedded Option

MATLAB Environment



```
MATLAB desktop keyboard shortcuts, such as Ctrl+S, are now customizable.
In addition, many keyboard shortcuts have changed for improved consistency
across the desktop.

To customize keyboard shortcuts, use Preferences. From there, you can also
restore previous default settings by following the steps outlined in Help.

Click here if you do not want to see this message again.

>> a = 1

a =

     1

>> b = 2

b =

     2

>> c = a+b;
>> 
```

Name	Value
a	1
b	2
c	3
toolbox_version	'maple'

```
% plot(times,tetas)
% hold; plot(times,tetams,':
% plot(times,energys)
% plot(times,tetadots)
% plot(tetas,tetadots)
%-- 17.02.2013 21:57 --%
a = 1
b = 2
c = a+b;
```

An example in MATLAB and some commands

Example: Assign *a = 3, b = 5, c = a*b* in the command window.

save command: Saves the workspace as «matlab.mat» to the command folder.

- save(‘name’): Save the workspace as «name.mat» to the command folder.
- save(‘name’,’x’,’y’): Save the variables x and y in the workspace as «name.mat» to the command folder.

load(‘name’) command: Load «name.mat» from command folder to the workspace.

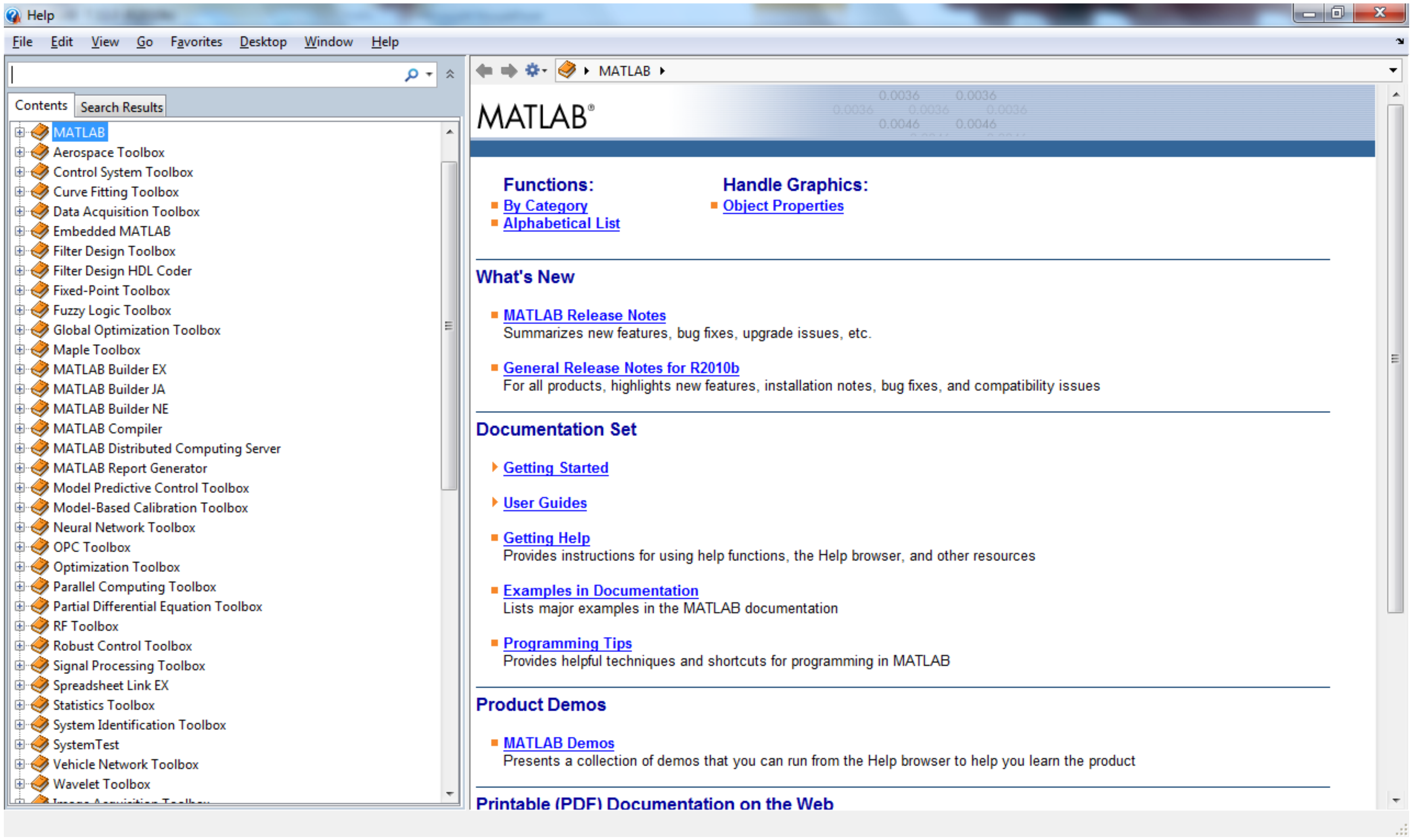
who command: List current variables.

whos command : List current variables command folder with their types.

clear command : Clear variables and functions from workspace.

clc command : Clear command window.

MATLAB Help window



MATLAB
Aerospace Toolbox
Control System Toolbox
Curve Fitting Toolbox
Data Acquisition Toolbox
Embedded MATLAB
Filter Design Toolbox
Filter Design HDL Coder
Fixed-Point Toolbox
Fuzzy Logic Toolbox
Global Optimization Toolbox
Maple Toolbox
MATLAB Builder EX
MATLAB Builder JA
MATLAB Builder NE
MATLAB Compiler
MATLAB Distributed Computing Server
MATLAB Report Generator
Model Predictive Control Toolbox
Model-Based Calibration Toolbox
Neural Network Toolbox
OPC Toolbox
Optimization Toolbox
Parallel Computing Toolbox
Partial Differential Equation Toolbox
RF Toolbox
Robust Control Toolbox
Signal Processing Toolbox
Spreadsheet Link EX
Statistics Toolbox
System Identification Toolbox
SystemTest
Vehicle Network Toolbox
Wavelet Toolbox

MATLAB®

Functions:
- By Category
- Alphabetical List

Handle Graphics:
- Object Properties

What's New

- MATLAB Release Notes
  Summarizes new features, bug fixes, upgrade issues, etc.
- General Release Notes for R2010b
  For all products, highlights new features, installation notes, bug fixes, and compatibility issues

Documentation Set

- Getting Started
- User Guides
- Getting Help
  Provides instructions for using help functions, the Help browser, and other resources
- Examples in Documentation
  Lists major examples in the MATLAB documentation
- Programming Tips
  Provides helpful techniques and shortcuts for programming in MATLAB

Product Demos

- MATLAB Demos
  Presents a collection of demos that you can run from the Help browser to help you learn the product

Printable (PDF) Documentation on the Web

RELATIONAL & LOGICAL OPERATORS

Introduction

- In MATLAB, logical expressions can be constructed not only from the six relational operators, but also from the four *logical operators,* namely «*NOT*», «*AND*», «XOR» and «*OR*».
- MATLAB generates ‘0’ for false and ‘1’ for true for logic operations.
- As a value, only the zero corresponds to logic ‘0’. All the other real numbers return ‘1’ to a logical operation.

Relational Operators - 1

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

Operation	Result
5==6	0
'A'<'Z'	1
5>6	0
5>=6	0
5<6	1
6<=6	1

The operators <, >, <=, and >= use only the real part of their operands for the comparison. The operators == and ~= test real and imaginary parts.

Relational Operators - 2

An array can be compared to a scalar. For instance, if, $a = \begin{bmatrix} 3 & 0 \\ 5 & 4 \end{bmatrix}$ and *b = 0*, *a>b* generates $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Two matrices with the same dimension can also be compared. In this case, each entry of he first matrix is compared to the corresponding entry of the second matrix. Functions defined for relational operators in MATLAB are given below.

Function	Equivalent	Examples	
eq	==	*eq(3,3) = 1*	*eq(3,5) = 0*
ne	~=	*ne(3,5) = 1*	*ne(3,3) = 0*
gt	>	*gt(3,5) = 0*	*gt(5,3) = 1*
ge	>=	*ge(3,5) = 0*	*ge(5,3) = 1*
lt	<	*lt(3,5) = 1*	*lt(5,3) = 0*
le	<=	*le(3,3) = 1*	*le(3,5) = 0*

Logical Operators - 1

There are four logical operators in MATLAB whicah are “& (AND)”, “| (OR)”, “xor (XOR)” ve “~ (NOT)”.

Inputs (Logical)		**AND**	**OR**	**XOR**	**NOT**
a	**b**	**a&b**	**a|b**	**xor(a,b)**	**~a**
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

The functions of these operators are given below.

Operator	**Symbols**	**Example**	
and	&	and(1,0)=0	and(1,1)=1
or	|	or(0,0)=0	or(1,0)=1
not	~	not(5)=0	not(0)=1
xor	xor	xor(5,1)=0	xor(0,3)=1

Logical Operators - 2

In the case of scalar or vector inputs, results of the logical operations are given in the table below. Where ‘*S*’ stands for scalar, ‘*A*’ stands for vector and ‘*R*’ stands for result.

Operation	**Result**
S1 & S2	R = S1 & S2
S & A	R(1) = S & A(1); R(2) = S & A(2); ...
A1 & A2	R(1) = A1(1) & A2(1); R(2) = A1(2) & A2(2); ...
S1 | S2	R = S1 | S2
S | A	R(1) = S | A(1); R(2) = S | A(2); ...
A1 | A2	R(1) = A1(1) | A2(1); R(2) = A1(2) | A2(2); ...
~S	R = ~S
~A	R(1) = ~A(1); R(2) = ~A(2), ...

Arithmetic Operations for Scalars

Operation	**in MATLAB**
Addition	`a + b`
Substraction	`a - b`
Multiplication	`a * b`
Right division	`a / b`
Left division	`b \ a`
Power	`a^b`

Precedence of Operators

Several operations may be combined in one expression, and MATLAB has strict rules about which operations are performed first in such cases

1. First arithmetic operations are resulted:
    - a. Parentheses,
    - b. Power, left to right,
    - c. Multiplication and division, left to right,
    - d. Addition and substraction, left to right.
2. Then, relational operations, <, <=, >, >=, ==, ~=, are performed.
3. In logical operations, «NOT» (~) is precedence.
4. Finally, logical operations (&, | ve xor) are resulted.

<u>Example</u>: Let a = 1, b = -5 and c = 0.

~a = 0,   a|c=1,   a&c=0,   a&b|c=1,   a&(b|c)=1,   ~(a&c)=1,

a>b&c=0,   a>(b&c)=1.

Some Special Cases and Functions

`&&` (`a&&b`) and `||` (`a||b`) operators are also used for logical operations. The difference from & and | is that, if the result can be generated with the first variable, the result is given what ever the value of the second variable.

<u>Example</u>: Let a=0, then `a&&b` generates ‘0’, similarly, if a=1, then `a||b` generates ‘1’.

<u>‘`all`’ and ‘`any`’ functions</u>:

‘`all`’ and ‘`any`’ functions are defined also for vectors and they run same as `&` and `|`. For instance, the result of `all(dizi)` is true if all elements of a vector are nonzero, and the result of `any(dizi)` is true if any element of a vector is a nonzero number or is logical ‘1’.

Logical functions

MATLAB has the logical functions that result according to the tested conditions. These functions can be used with the other operators. Some of these functions are described in the table below.

Function	**Result**
`ischar(a)`	Logical ‘1’, if «a» is a character array (string), ‘0’ otherwise.
`isnumeric(a)`	Logical ‘1’, if «a» is a numeric array, ‘0’ otherwise.
`isempty(a)`	Logical ‘1’, if «a» is an empty array, ‘0’ otherwise.
`isinf(a)`	Logical ‘1’, if «a» is an infinite element, ‘0’ otherwise.
`isnan(a)`	Logical ‘1’, if «a» is not-a-number, ‘0’ otherwise.

VARIABLES, VECTORS and MATRICES in MATLAB*

*** Some of the contents are adopted from**
Essential MATLAB for Engineers and Scientists, B. H. Hahn, D. T. Valentine, 5th Ed., Academic Press, MA, 2013.

Variables

- A variable *name* must comply with the following two rules,
  - It may consist only of the letters *a–z*, the digits *0–9*, and the underscore (_),
  - It must start with a letter.

Examples:

1) `a_variable, myvar2` are valid.

2) `a-variable, 2myvar, var$, _var` are invalid.

- A variable is created simply by assigning a value to it at the command line or in a program.

For example,

```
a = 98,
```

assigns `98` to the variable `a`.

Variables ctd.

- MATLAB is *case-sensitive*, which means it distinguishes between upper and lowercase letters.
- All the variables you create during a session remain in the workspace until you `clear` them.
- Values of the variables in the workspace can be used or changed at any stage during the session.
- Remember that, the command `who` lists the names of all the variables in your workspace. And the command `whos` lists the size of each variable as well.

Variables ctd.

- In MATLAB, all variables are referred to as *arrays, whether* they are single-valued (scalars) or multi-valued (vectors or matrices). In other words, a scalar is a 1-by-1 array.

- Each scalar occupies eight *bytes of storage* where a byte is the amount of computer memory required for one character (one byte is the same as eight *bits*).

Examples:

```
value = 10;
var1 = value;
a = 5i-5;
b = a/5;
x = 3, y = 5;
```

Vectors

- A *vector* is a special type of matrix, having only one row or one column. Vectors are called *lists* or *arrays* in some programming languages.

- MATLAB handles vectors and matrices in the same way.

- Elements in the list must be enclosed in square brackets, not parentheses. And Elements in the list must be separated *either by spaces or by commas.*

- To begin, we try the accompanying short exercises on the command line.

Exercises on Vectors

Enter a statement like:

```
v1 = [2 4 -5 0 10]
```

or

```
v1 = [2,4,-5,0,10]
```

- Can you see that you have created a vector (list) with five elements?
- Enter the command `disp(v1)` to see how MATLAB displays a vector.
- Enter the command `whos` (or look in the Workspace browser).

The colon operator

- A vector can also be generated with the *colon operator*

  For example, enter the following statements:

1)	`x = 1:10`	(elements are the integers 1, 2, …, 10),
2)	`x = 1:0.5:4`	(elements are the values 1, 1.5, …, 4 in increments of 0.5.),
3)	`x = 10:-1:1`	(elements are the integers 10, 9, …, 1),
4)	`x = 1:2:6`	(elements are 1, 3, 5).

The colon operator ctd.

The General Structure:

`j:k`	is the same as *[j, j+1, ..., k]*, or empty when *j > k*.
`j:i:k`	is the same as *[j, j+i, j+2i, ..., j+m*i]*, where *m* is the rounded value of *((k-j)/i)* towards zero, for integer values.

`linspace` and `logspace` functions

- The `linspace` function generates linearly spaced vectors. It is similar to the colon operator ":", but gives direct control over the number of points. For example,

```
linspace(0, pi/2, 10)
```

  creates a vector of 10 equally spaced points from *0* to *π/2.*

- The `logspace` function generates logarithmically spaced vectors. Especially useful for creating frequency vectors, it is a logarithmic equivalent of `linspace` and the ":" or colon operator. For example,

```
y = logspace(a,b,n)
```

  generates *n* points between decades *10^a* and *10^b*.

Transposing vectors

- All of the vectors we examined so far are *row vectors*. Each has one row and several columns.

To generate the *column vectors,*

1) ‘;’ can be used, i.e. `x1 = [1;2;3]` generates a 3 row 1 column vector,
2) A row vector can be transposed, i.e. `x2 = [1 2 3]’` also generates a 3 row 1 column vector.

```
              1
x1 = x2 =     2
              3
```

Subscripts

- We can refer to particular elements of a vector by means of *subscripts.*

  For example,

  - Let `v = [0,5,10,15,20,25,30];`
  - `v(3)` gives the third element of *v* which is 10, here the numeral 3 is the subscript.
  - `v(2:4)` gives the second, third and fourth elements of *v*.
  - `v(1:2:6)` gives the first, third and fifth elements of *v*.
  - `v([1,7,3])` gives the first, seventh and third elements of *v*.

Matrices

- A *matrix* may be thought of as a table consisting of rows and columns. We can create a matrix just as we do a vector, except that a semicolon is used to indicate the end of a row.

  For example, `a = [1 2 3; 4 5 6]` results in

```
a =
1     2     3
4     5     6
```

- A matrix can also be transposed, i.e. `a’` results in

```
ans =
1     4
2     5
3     6
```

Matrices ctd.

- A matrix can be constructed from column vectors of the same length. For example,

```
x = 0:30:180;
table = [x' sin(x*pi/180)']
```

result in,

```
table =
          0          0
    30.0000     0.5000
    60.0000     0.8660
    90.0000     1.0000
   120.0000     0.8660
   150.0000     0.5000
   180.0000     0.0000
```

Matrices ctd.

- Similar to the vectors, particular elements of matrices can also be referred by means of subscripts. For example, let

```
m =

     1     2     3
     4     5     6
```

then, `m(2,3)` results in 6.

Some built-in functions

Function	**Description**
`zeros(n)`	returns *nxn* matrix of zeros where *n* is an integer
`zeros(n,m)`	returns *nxm* matrix of zeros where *n* and *m* are integers
`ones(n)`	returns *nxn* matrix of ones where *n* is an integer
`ones(n,m)`	returns *nxm* matrix of ones where *n* and *m* are integers
`size(y)`	gives the sizes of each dimension of array *y* in a vector
`eye(n)`	returns *nxn* identity matrix
`eye(n,m)`	returns an *nxm* matrix with 1's on the diagonal and 0's elsewhere.
`length(y)`	returns the length of the vector *y* or largest array dimension of *y*

Examples

```
>>x = [1,2;3,4];
>>y = ones(size(x))
y =
     1    1
     1    1
>>a = [1 2 3;4 5 6];
>>b = [2 4 6;1 3 5];
>>c = ones(size(a))-eye(size(b))
c =
     0    1    1
     1    0    1
>>d = ones(size(a))-length(b)
d =
     -2    -2    -2
     -2    -2    -2
```

Arithmetic operations for arrays

Operation	**Operator**	**Description**
Addition	`a+b`	Element by element matrix addition
Substraction	`a-b`	Element by element matrix substraction
Multiplication	`a.*b`	Element by element matrix multiplication (`a` or `b` can be a scalar)
Matrix multiplication	`a*b`	Matris multiplication
Right division	`a./b`	Element by element right division (`a` or `b` can be a scalar)
Left division	`a.\b`	Element by element left division (`a` or `b` can be a scalar)
Right matrix division	`a/b`	The inverse of the matrix `b` is multiplied by the matrix `a` from left
Left matrix division	`a\b`	The matrix `b` is multiplied by the inverse of the matrix `a` from left
Power	`a.^b`	Element by element power (`a` or `b` can be a scalar)

Examples

```
>> x = [2 4;6 8];y = [2 8;5 9];
>> z = x-y
z =
     0    -4
     1    -1

>> x = rand(2,2);y = rand(3,2)
y =
    0.6324    0.5469
    0.0975    0.9575
    0.2785    0.9649
>> z = x+y
??? Error using ==> plus
Matrix dimensions must agree.
```

```
>> x = [2 4;6 8];y = [2 8;5 9];
>> b = x+1
b =
     3     5
     7     9
>> x*y
ans =
    24    52
    52   120
>> x.*y
ans =
     4    32
    30    72
```

Submatrices, Examples

```
>> x = [1 2;3 4];
>> x(:,1)
ans =
     1
     3
>> x(2,:)
ans =
     3     4
>> x(:)
ans =
     1
     3
     2
     4
```

```
>> y = [1 2 3; 4 5 6; 7 8 9; 10 11 12];
>> y(2,:)

ans =


     4     5     6

>> y(2:4,1:2)

ans =


     4     5
     7     8
    10    11
```

Some constants and permanent variables in MATLAB

Function	Description
`ans`	Most recent answer
`pi`	The π number in double precision
`i,j`	Imaginary unit
`eps`	Floating-point relative accuracy
`Inf`	Infinity (i.e. 1/0)
`NaN`	Not-a-Number (i.e. 0/0)
`clock`	Current time as date vector
`date`	Current date string
`tic, toc`	Measure performance using stopwatch timer
`realmax, realmin`	Largest and smallest positive floating-point number
`bitmax`	Maximum double-precision floating-point integer