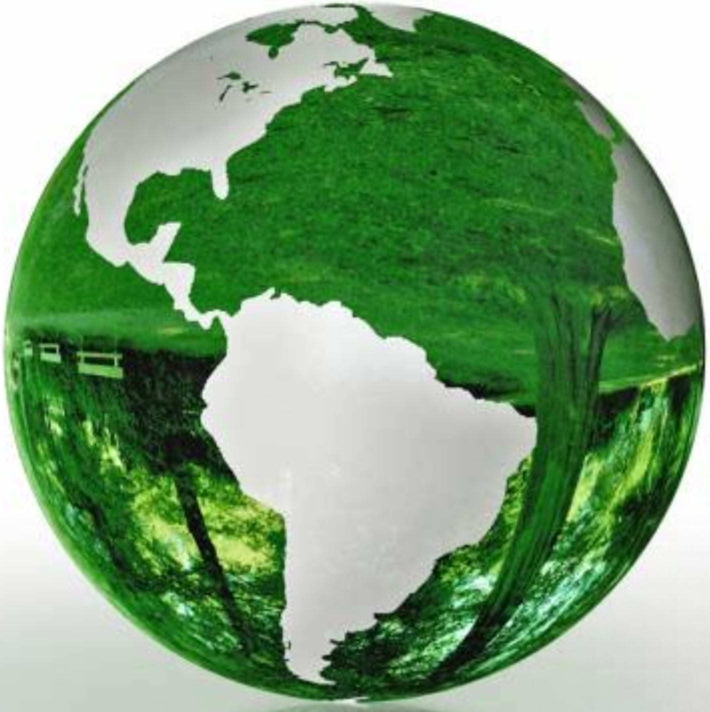
Computer Aided Design (CAD)



Lecture 3

Scalars, Complex Numbers, and Vectors in Matlab

Dr.Eng. Basem ElHalawany

Schedule (Draft)

Topics	Estimated Duration (# Lectures)
Introduction	1
Introduction to Matlab Environment	1
Matlab Programing (m-files)	5
Modeling using Matlab Simulink Tool	4
Communication Systems Simulation (Applications)	3
Midterm	8^{th} Week
Introduction to FPGA + Review on Digital Logic/Circuits	2
VHDL Modeling Language	4
VHDL Application	2
Introduction to OPNET Network Simulator	3
Course Closeout / Feedback/ project (s) Delivery	1

The Lecture is based on :

A. Matlab by Example: Programming Basics, Munther Gdeisat

1.3 Matlab Editor—Cell Mode

1.3.1 Enabling Cell Mode (Section Mode "in 2014")

```
clear; clc;
%part 1
x1 = 10
y1 = x1.^2
%part 2
x2 = 20
y2 = 3*x2.^3
%part 3
x3 = 30
y3 = 5*x3.^2
```

- Suppose that you have the following program that can be divided into three individual parts.
- Suppose that you would like to run the code one part at a time.

In order to do this, you can use the Cell Mode in Matlab (Section Mode)

- Right Click in the m-file and "Insert Section". Check how the file is changed by yourself

1.3 Matlab Editor—Cell Mode

1.3.3 Evaluating Code in a Cell

```
%% Run Standard Waterfilling algorithm in OFDM
%% Refresh
close all
clear
clc

%% Initialize
nSubChannel                 = 16;
totalPower                  = 1e-5;            %  -20 dBm
channelStateInformation = random('rayleigh',1/0.6552,1,nSubChannel);
bandwidth                   = 1e6;             %  1 MHz
noiseDensity                = 1e-11;           % -80 dBm

%% Run SWF
[Capacity PowerAllocated] = ofdmwaterfilling(nSubChannel,totalPower,channelS
```

2 Scalars in Matlab

- Variables are created either by Matlab or by the user
- Variables created by Matlab are considered to be special variables, whose values are assigned by Matlab.

2.1.1 Matlab Special Variables

```
>>pi
```

Then press **Enter**. Matlab responds with

```
ans =
      3.1416
```

- This command generates another special variable "*ans*"
- **ans** saves the result of any Matlab operation if the value of the result is not specifically assigned to a variable.

2.1.1 Matlab Special Variables

Other examples of special variables are i and j.

The value for both variables is defined as $\sqrt{-1}$.

```
>>i
```

Then press **Enter**. Matlab responds with

```
>>ans
        0 + 1.0000i
```

```
>>j
```

Then press **Enter**. Matlab responds with

```
>>ans
        0 + 1.0000i
```

2.1.1.2 *Changing the Values of Matlab Special Variables*

- The user can change the value of the special variables.

```
>>pi = 1
```

- To restore the value of the special variable pi, type at the Command Prompt

```
>>clear pi
```

2.1.2 User-Defined Variables

2.1.2.1 *Naming a User-Defined Variable*

- A variable name must not contain spaces or hyphens (-).
- A variable name can contain up to 63 characters.
- A variable name must start with a letter (a–z or A–Z), followed by any number of letters, digits (0–9), or underscores (_).
- Punctuation characters such commas (,) or apostrophes (') are not allowed, because many of them have special meanings in Matlab.
- A variable name must not be a **Script M-file** name or an existing **Matlab function** name.

Matlab is Case Sensitive

Clearing a User-Defined Variable

```
>>clear y
```

2.2.1 Approximating Numbers

round Function

- This function rounds a real number upward, or downward, toward the nearest integer.

```
>> x = round(2.51)          x =
                                  3

>> y = round(2.49)          y =
                                  2
```

fix Function

- This function truncates (eliminates) the decimal part of a real number, leaving the integer part unchanged.

```
>> x = fix(2.51)            x =
                                  2

>> y = fix(-2.51)           y =
                                 -2
```

2.2.1 Approximating Numbers

`ceil` Function

➢ This function rounds up a real number toward the nearest higher integer.

```
>>x=ceil(2.51)        x =
                          3

>>y=ceil(2.49)        y =
                          3
```

`floor` Function

➢ This function rounds down a real number toward the nearest lower integer.

```
>>x=floor(2.51)       x =
                          2
>>y=floor(2.49)       y =
                          2
```

2.3.2 Precedence of Mathematical Operations

- Matlab evaluates mathematical expressions from left to right.
- Mathematical expressions may contain addition, subtraction, multiplication, division, and exponential mathematical operations as well as parentheses.
- These mathematical operations are evaluated in the following order :

I. Parentheses, by starting with the innermost set and proceeding outward
II. The exponentiation operation
III. Multiplication and division
IV. Addition and subtraction.

Mathematical Expressions for Scalar Variables

2.3.3 From Mathematical Expressions to Matlab Expressions

Example 1

$$r = \frac{x+y}{z}, \quad \rightarrow \quad r = \left\{ \frac{\{x+y\}_1}{z} \right\}_2$$

- The addition operation needs to be evaluated first followed by the division.
- Since the division operation has a higher priority in Matlab than the addition operation, parentheses are needed to alter this priority order to give the addition operation a higher priority than that of the division operation.

```
r = (x + y)/z;
```

Example 2

$$r = x + \frac{y}{z}, \quad \rightarrow \quad r = \left\{ x + \left\{ \frac{y}{z} \right\}_1 \right\}_2$$

- In this mathematical expression, the division has a higher priority than the addition operation.
- Since the order of evaluating this mathematical expression exactly follows the priority of mathematical operations in Matlab, you should not use parentheses

```
r = x + y/z;
```

2.4 Relational and Logical Operations for Scalar Variables

2.4.1 The `logical` Class

- Any variable with a logical class has a value of either true or false.
- Matlab represents true as 1, and false as 0.

```
>>r=true
```

Matlab responds with

```
r =
     1
```

To check the class of `r`, type at the **Command Prompt**

```
>>whos r
```

Matlab responds with

```
Name   Size   Bytes   Class     Attributes
r      1×1    1       logical
```

2.4.2 The Relational Operators

- The relational operators require two operands, and they compare two values.
- The relational operators produce variables with a logical class

Matlab has six relational operators which are

1. Greater than “ > ”
2. Less than “ < ”
3. Greater than or equal “ >= ”
4. Less than or equal “ <= ”
5. Equal “ == ”
6. Not equal “ ~= ”

```
>> x = 1;
>> y = 2;
>> a = x > y
```

Matlab responds and displays the value of a as

```
a =
      0
```

2.4.3 The Logical Operators

Matlab has three logical operators which are

1. AND "&"
2. OR "|"
3. NOT "~"

The logical operators produce variables with the `logical` class.

AND "&" Logical Operator

Operand 1	Operand 2	&
0	0	0
0	nonzero	0
nonzero	0	0
nonzero	nonzero	1

```
x = 1;
y = 2;
g = x&y
```

```
g =
     1
```

OR "|" Logical Operator

x	y	x | y
0	0	0
0	nonzero	1
nonzero	0	1
nonzero	nonzero	1

```
x = 1;
y = 2;
n = x|y
```

```
n =
     1
```

NOT "~" Logical Operator

```
x = 0;
z = ~x
```

```
z =
     1
```

```
x = 1;
w = ~x
```

```
w =
     0
```

```
x = -1;
y = ~x
```

```
y =
     0
```

2.4.4 Combining Logical and Rational Operators

Logical and rational operators can be combined. For example,

```
x = 1;
y = 2;
n = (x < 3)&(y < 0)
```

Matlab responds with

```
n =
    0
```

2.5 Complex Scalar Variables

2.5.2 Creating Complex Scalar Variables

```
>> z = 1 + 2i
```

```
z =

   1.0000 + 2.0000i
```

You can use j instead of i to represent $\sqrt{-1}$. For example,

```
>> z = 1 + 2j;
```

A third method to create a complex number is

```
>> z = 1 + 2*i;
```

Note

```
>> j = 2;
>> x = 1 + 2*j
```

Matlab responds as follows

```
x =

   5
```

2.5 Complex Scalar Variables

- Be careful: These expressions are different

```
y = 7/2*i
and
x = 7/2i
```

$y = (7/2)i = 3.5i$

and

$x = 7/(2i) = -3.5i.$

2.5.7 Conjugate of a Complex Number

```
>>z=1+2i;
>>z1=conj(z)
```

Matlab responds as follows:

```
z1 =
      1.0000 - 2.0000i
```

2.5 Complex Scalar Variables

2.5.8 Modulus and Angle of a Complex Number

```
>>z = 3 + 4i;
>>a = abs(z)

a =

     5
```

```
>>b = angle(z)

b =
      0.9273
```

- Note that the angle is given here in radians.
- To convert the angle from radians to degrees, multiply it by 180/ π .

```
>>angle_in_degrees = angle(z)*180/pi
angle_in_degrees =
  53.1301
```

3 Vectors in Matlab

➢ A vector is an array that contains only one row or one column.

```
>> x = [2,3,5];
```

```
>> y = [4;9;7;12];
```

➢ Note that in this book that we use **bold fonts** to distinguish vector variables from scalar variables.

3.1.2.3 Transpose Operation

➢ Applying the transpose operation to vectors changes a row vector to a column vector and vice versa.

```
>> x = [2, 3, 5];
>> x = x';
```

```
x =
     2
     3
     5
```

3.1.2.4 Determining the Number of Elements in a Vector

```
>>x = [2,3,5];
>>n = length(x)
```

Matlab responds and displays the result as

```
n =
    3
```

3.1.2.5 Converting a Vector to a Column Vector

- The Matlab colon operator, ":", can be used to convert a vector to a column vector.

```
>>y=[1,2,3,4,5];
>> y=y(:)
   y =
        1
        2
        3
        4
        5
```

Creating Vectors Using the Linear Method

- The linear method can be used to create a row vector that has linearly spaced elements, that is, the difference between two successive elements in the vector is constant.

```
>> x = 0:2:10
x =
     0   2   4   6   8   10

>> y = 10: -2:0;
y =
     10   8   6   4   2   0
```

Creating Vectors Using the Linear Spacing Method

The Matlab function `linspace(x1,x2,N)` can be used to create a row vector.

- `x1` is the start value.
- `x2` is the final value.
- `N` is the number of elements in a vector.

```
>> x = linspace(0, 10, 6)
x =
     0   2   4   6   8   10
```

3.1.6 Empty Vectors

- An empty vector is a vector that does not contain any elements.

```
>> x = [];
```

3.1.7 Vector Concatenation

- Two vectors can be concatenated and become a single vector.

```
>> x1 = [1,2,3];
>> x2 = [4,5,6];
>> x  = [x1,x2]
```

Matlab responds and produces the output

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

3.1.8.1.3 *Transpose Operation for Complex Vectors*

- Applying the transpose operation to a complex vector not only changes rows to columns and vice versa, but also conjugates the vector's elements (Vector Hermitan)

```
>> x = [2 + i, 3 - 2i, 5 + 3i]
>> z = x';
```

Matlab responds with

```
z =
    2.0000 - 1.0000i
    3.0000 + 2.0000i
    5.0000 - 3.0000i
```

```
y  =  [4 - 3i; 9 + 4i; 7 - 5i; 12 + 11i];
z  =  y.';
```

Applying the command y.' changes rows to columns and columns to rows only. It does not conjugate the vector y elements.

3.2.1 Relational Operations on Vectors

```
>> x = [2,4,7,9,-1,2];
>> y = [-1,4,8,1,-4,6];
>> z = x > y

 z =
     1   0   0   1   1   0
```

- This command determines whether the value of each element in the vector x is greater than the corresponding element in the y.

3.2.2 The Logical Operations on Vectors

```
x = [0,4,7,0,-1,2];
y = [1,2,8,0,-4,6];
z = x&y
```

Matlab produces the output

```
 z =
     0   1   1   0   1   1
```

- Remember: An input to relational and logical operators is considered to be true if it has a nonzero value,
- An input with a 0 value is considered to be false.
- An input with a negative value is considered to be true.

3.3 Accessing Elements in Vectors

3.3.1 Accessing an Individual Element in a Vector Using its Index

- you can access an individual element within the vector using the “index”

```
x =
     3    6    9    12    15    18
```

Index	1	2	3	4	5	6
Value	3	6	9	12	15	18

- To access the first element in the vector:

```
>> x(1)
```

Matlab responds with

```
ans =
      3
```

- Note that in some languages, for example, the C programming language, the first element of a vector may be referred to as element 0

3.3.1 Accessing an Individual Element in a Vector Using its Index

To access the last element in the vector, type at the Matlab **Command Prompt**

```
>> s = x(end);
>> s
```

Matlab responds with

```
s =
      18
```

Let us try to access the seventh element in the vector x as follows:

```
>> x(7)
```

Matlab responds with the error message

```
??? Index exceeds matrix dimensions.
```

3.3.2 Accessing a Group of Elements in a Vector Using Their Indices

```
>> y = 2:3:18
```

```
y =
      2   5   8   11   14   17
```

To access the first three elements of the vector y

```
>> a = y(1:3);
```

To access the last three elements of the vector y

```
>> b = y(end - 2:end);
```

To access the second, third, and the fourth elements of the vector y

```
>> c = y(2:4);
```

or

```
>> c = y([2,3, 4]);
```

3.3.3 Accessing Elements in a Vector Using Their Values

- Matlab enables you to easily search for an individual element, or a group of elements, in a vector, depending on their values.

```
>> y = [2,3,5,5,7,10,12];
```

- To find the indices of the elements whose values are equal to 5

```
>> a = find(y == 5)
```

Matlab responds with

```
a =
    3   4
```

- To find the indices of the elements whose values are less than or equal to 9,

```
>> c = find(y <= 9)

c =
    1   2   3   4   5
```

- To find the values of the elements in the vector y that are less than, or equal to, 9;

```
>> d = y(c)
```

```
d =
    2   3   5   5   7
```