Matlab Programming
Introduction[1] [2]

Mili I. Shah

August 10, 2009

[1] *Matlab, An Introduction with Applications, 2nd ed.* by Amos Gilat

[2] *Matlab Guide, 2nd ed.* by D. J. Higham and N. J. Higham

Starting Matlab

Go to

http://www.loyola.edu/moresoftware/

and login with your Loyola name and password...

Matlab has eight main windows:

Command Window	Main window, enter variables, runs programs
Figure Window	Contains output from graphic commands
Editor Window	Creates and debugs script and function files
Help Window	Provides help information
Launch Pad Window	Provides access to tools, demos, and documentation
Command History Window	Logs commands entered in the Command Window
Workspace Window	Provides information about the variables that are used
Current Directory Window	Shows the files in the current directory

Command Window

- To type a command the cursor must be placed next to the command prompt (>>).
- Press **Enter** for the command to be executed. Multiple commands can be typed by typing a comma (,) between them.
- A semicolon (;) at the end of a command suppresses the screen output.
- Upper and lower case characters are not equivalent.
- The up and down arrow keys can be used to scroll through previous commands. Also an old command can be recalled by typing the first few characters followed by the up arrow.
- Type `help topic` to access online help on the command, function, or symbol `topic`.
- Type `clc` to clear the screen
- Type `exit` or `quit` to quit Matlab.

Built-In Functions

`sqrt(x)`	Square root
`exp(x)`	Exponential (e^x)
`abs(x)`	Absolute value
`log(x)`	Natural logarithm
`sin(x)`	Sine of x
`cos(x)`	Cosine of x
`tan(x)`	Tangent of x
`cot(x)`	Cotangent of x
`pi`	π

Built-In Functions

`sqrt(x)`	Square root
`exp(x)`	Exponential (e^x)
`abs(x)`	Absolute value
`log(x)`	Natural logarithm
`sin(x)`	Sine of x
`cos(x)`	Cosine of x
`tan(x)`	Tangent of x
`cot(x)`	Cotangent of x
`pi`	π

```
>> sqrt(4)
ans =
2
>> pi
ans =
3.1416
```

Defining Scalar Variables

Variable = Numerical value or computable expression

- = is the **assignment operator** which assigns a value to a variable
- Left-hand side can include only **one** variable name
- Right-hand side can be a number or an expression made up of numbers and/or variables previously assigned numerical values
- Variables must begin with a letter
- Press **Enter** to make the assignment
- `ans` is the value of the last expression that is not assigned

Remember:

- Use semicolon (;) to suppress screen output
- Multiple commands can be typed by typing a comma (,) between them.

Defining Scalar Variables

Example: Assign the number 3 to variable **a** and 4 to variable **b**. Print out $\sqrt{a^2 + b^2}$ and assign the solution to the variable **c**.

Defining Scalar Variables

Example: Assign the number 3 to variable **a** and 4 to variable **b**. Print out $\sqrt{a^2 + b^2}$ and assign the solution to the variable **c**.

```
>> a=3; b=4; c = sqrt(a^2+b^2)
c =
5
```

Defining Scalar Variables

Example: Assign the number 3 to variable **a** and 4 to variable **b**. Print out $\sqrt{a^2 + b^2}$ and assign the solution to the variable **c**.

```
>> a=3; b=4; c = sqrt(a^2+b^2)
c =
5
```

Example: Verify

$$\cos^2 \frac{x}{2} = \frac{\tan x + \sin x}{2 \tan x}$$

by calculating each side of the equation for $x = \pi/5$.

Defining Scalar Variables

Example: Assign the number 3 to variable **a** and 4 to variable **b**. Print out $\sqrt{a^2 + b^2}$ and assign the solution to the variable **c**.

```
>> a=3; b=4; c = sqrt(a^2+b^2)
c =
5
```

Example: Verify

$$\cos^2 \frac{x}{2} = \frac{\tan x + \sin x}{2 \tan x}$$

by calculating each side of the equation for $x = \pi/5$.

```
>> x = pi/5;
>> LHS = cos(x/2)^2, RHS = (tan(x)+sin(x))/(2*tan(x))
LHS =
0.9045
RHS =
0.9045
```

Arrays

- Used to store and manipulate numbers
- Arranged in rows or columns

Arrays

Arrays

- Used to store and manipulate numbers
- Arranged in rows or columns

One-Dimensional Array (Vector)

- Represents point in n-dimensional space
  Ex: (x, y) in 2D and (x, y, z) in 3D

Arrays

Arrays

- Used to store and manipulate numbers
- Arranged in rows or columns

One-Dimensional Array (Vector)

- Represents point in n-dimensional space
  Ex: (x, y) in 2D and (x, y, z) in 3D
- **Row Vector** (Use space or comma between numbers)

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

Arrays

Arrays

- Used to store and manipulate numbers
- Arranged in rows or columns

One-Dimensional Array (Vector)

- Represents point in n-dimensional space
  Ex: (x, y) in 2D and (x, y, z) in 3D
- **Row Vector** (Use space or comma between numbers)

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

- **Column Vector** (Use semicolon between numbers)

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

Arrays

- **Constant Spaced Vectors**:
  - From m spaced by q to n

$$\text{variable} = [m : q : n]$$

- **Constant Spaced Vectors**:
  - From m spaced by q to n

variable $= [m : q : n]$

```
>> x = [1:2:7]
x =
1 3 5 7
```

Arrays

- **Constant Spaced Vectors**:
  - From m spaced by q to n

    variable $= [m : q : n]$

    ```
    >> x = [1:2:7]
    x =
    1 3 5 7
    ```

  - From m to n with q elements

    variable = linspace(m,n,q)

- **Constant Spaced Vectors**:
  - From m spaced by q to n

    variable = $[m : q : n]$

    ```
    >> x = [1:2:7]
    x =
    1 3 5 7
    ```

  - From m to n with q elements

    variable = linspace(m,n,q)

    ```
    >> x = linspace(0,1,5)
    x =
    0 0.2500 0.5000 0.7500 1.0000
    ```

Two-Dimensional Array (Matrix)

- Can store information like a table
- Solve systems of equations such as

$$\begin{aligned} 2x + 3y + z &= 4 \\ x - 5y + 3z &= 3 \\ 4x - 2y + 3z &= 2 \end{aligned}$$

variable = [1st row; 2nd row; . . . ; last row]

```
>>x = [ 2 3 1; 1 -5 3; 4 -2 3]
x =
 2   3   1
 1  -5   3
 4  -2   3
```

Addressing Elements

- **Vector**:
  - `ve(k)` picks the kth element of `ve`
  - `ve(m:n)` picks the mth through nth elements of `ve`

```
>> ve = [1 5 2 6 8 7]
ve =
1 5 2 6 8 7
>> ve(5)
ans =
8
>> ve(2:4)
ans =
5 2 6
```

Addressing Elements

- **Matrix**:
  - `mat(m,n)` picks the (m, n)th element of `mat`
  - `mat(m:n,p:q)` picks the $(m : n) \times (p : q)$ submatrix of `mat`

```
>> mat = [1 4 2 3; 3 6 9 2;1 4 9 7; 2 5 1 8]
mat =
 1   4   2   3
 3   6   9   2
 1   4   9   7
 2   5   1   8
>> mat(2,3)
ans =
9
>> mat(2:4, 1:3)
        3   6   9
ans =   1   4   9
        2   5   1
```

Adding Elements

- Can add elements by using the variable within vector/matrix
- Must be of appropriate size

```
>> mat = [1 4 2 3; 3 6 9 2;1 4 9 7]
mat =
 1  4  2  3
 3  6  9  2
 1  4  9  7
>> [mat; 2 5 1 8]
ans =
 1  4  2  3
 3  6  9  2
 1  4  9  7
 2  5  1  8
```

Deleting Elements

- Delete elements by assigning nothing to these elements

```
>> ve = [1 5 2 6 8 7]
ve =
1 5 2 6 8 7
>> ve(2:4) = [ ]
ve =
1 8 7
>> mat = [1 4 2 3; 3 6 9 2;1 4 9 7]
mat =
 1   4   2   3
 3   6   9   2
 1   4   9   7
>> mat(2:3,:) = [ ]
mat =
1 4 2 3
```

Helpful Tips for Arrays

`length(A)`	Returns number of elements in the vector A
`size(A)`	Returns size of matrix A
`reshape(A,m,n)`	Rearranges A to have m rows and n columns (arranged column-wise)

Strings

- Is an array of characters
- Created by typing characters within single quotes
- Can include letters, digits, symbols and spaces

```
>> a = 'Matlab is AWESOME'
a =
Matlab is AWESOME
>> a(1) =
ans =
M
>> a(1:6)
ans =
Matlab
```

Strings

- Is an array of characters
- Created by typing characters within single quotes
- Can include letters, digits, symbols and spaces

```
>> a = 'Matlab is AWESOME'
a =
Matlab is AWESOME
>> a(1) =
ans =
M
>> a(1:6)
ans =
Matlab
 >> a(1:6) = 'M Shah'
a =
M Shah is AWESOME
```