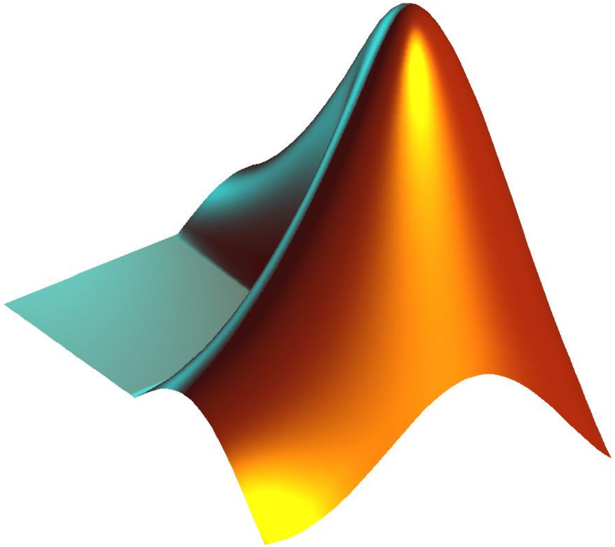
Introduction to Matlab



By:
Hossein Hamooni
Fall 2014

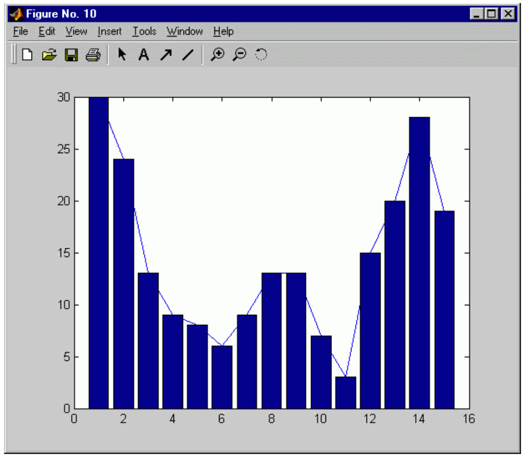
Why Matlab?

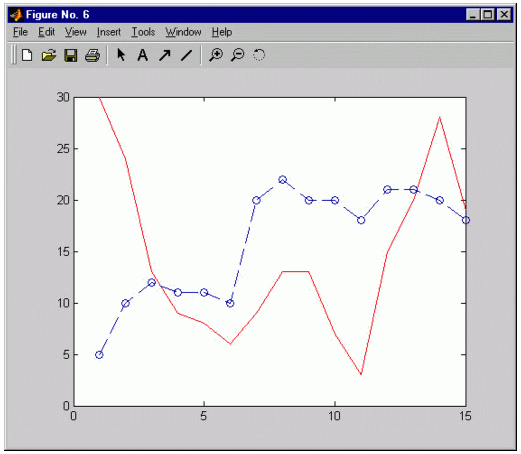
- Data analytics task
- Large data processing
- Multi-platform, Multi Format data importing
- Graphing
- Modeling
- Lots of built-in functions for rapid prototyping
- UNM students can have it for free (It is really expensive)

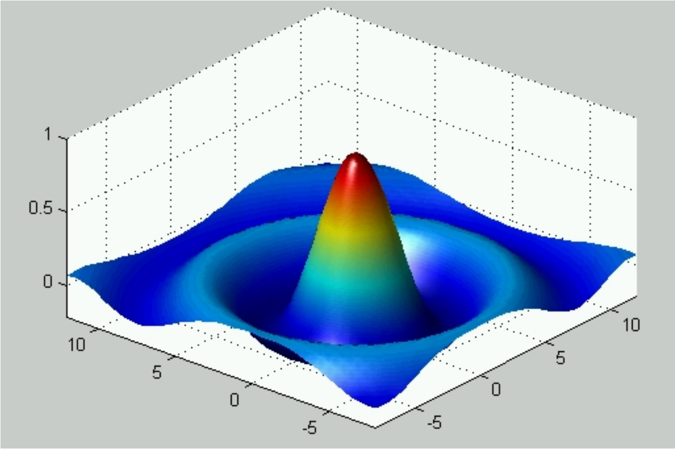
Why Matlab?

Graphing

- A Comprehensive array of plotting options available from 2 to 4 dimensions
- Full control of formatting, axes, and other visual representational elements







How to install Matlab

1. Visit the following link: http://it.unm.edu/download
2. Select your operating System
3. Click on Matlab link
4. Log in to your UNM account
5. Follow the instructions

Understanding the Matlab Environment:

Executing Commands

Basic Calculation Operators:

+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponentiation

Using Matlab

Solving equations using variables

- Matlab is an *expression language*
- Expressions typed by the user are interpreted and evaluated by the Matlab system
- Variables are names used to store values
- Variable names allow stored values to be retrieved for calculations or permanently saved

Variable = *Expression*

Or

Expression

***Variable Names are Case Sensitive!*

```
>> x = 6
x = 6
>> y = 2
y = 2
>> x + y
Ans = 8
```

```
>> x * y
Ans = 12
>>  x / y
Ans = 3
>> x ^ y
Ans = 36
```

Using Matlab

Working with Matrices

- Matlab works with essentially only one kind of object, a rectangular numerical matrix
- A matrix is a collection of numerical values that are organized into a specific configuration of rows and columns.
- The number of rows and columns can be any number

Example

3 rows and 4 columns define a 3 x 4 matrix having 12 elements

- A scalar is a single number and is represented by a 1 x 1 matrix in matlab.
- A vector is a one dimensional array of numbers and is represented by an n x 1 column vector or a 1 x n row vector of n elements

Exercises

Working with Matrices

How to define the following Matrices in Matlab using spaces, commas, and semicolons to separate rows and columns:

$$A = \begin{bmatrix} 1 & 21 & 6 \\ 5 & 17 & 9 \\ 31 & 2 & 7 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 64 & 122 & 78 & 38 & 55 \end{bmatrix}$$

$$C = \begin{bmatrix} 4 \\ 22 \\ 16 \\ 160 \end{bmatrix}$$

Matrix Operations

Indexing Matrices

```
A = [1 2 4 5
     6 3 8 2]
```

- The colon operator can be used to remove entire rows or columns

```
>> A(:,3) = [ ]
A = [1 2 5
     6 3 2]

>> A(2,:) = [ ]
A = [1 2 5]
```

Matrix Operations

Scalar Operations

- Scalar (single value) calculations can be can performed on matrices and arrays

Basic Calculation Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponentiation

Matrix Operations

Element by Element Multiplication

- **Element by element multiplication of matrices is performed with the .* operator**
- **Matrices must have identical dimensions**

```
    A = [1 2        B = [1      D = [2 2     E = [2 4 3 6]
         6 3 ]           7           2 2 ]
                         3
                         3]
>>A .* D
Ans = [ 2  4
        12 6]
```

Matrix Operations

Element by Element Division

- **Element by element division of matrices is performed with the ./ operator**
- **Matrices must have identical dimensions**

```
    A = [1 2 4 5      B = [1      D = [2 2 2 2   E = [2 4 3 6]
         6 3 8 2]          7           2 2 2 2]
                           3
                           3]
>>A ./ D

Ans =   [ 0.5000    1.0000    2.0000    2.5000
          3.0000    1.5000    4.0000    1.0000 ]
```

Matrix Operations

Matrix Exponents

- **Built in matrix Exponentiation in Matlab is either:**

1. **A series of *Algebraic dot products***
2. ***Element by element exponentiation***

Examples:

- **A^2 = A * A (Matrix must be square)**
- **A.^2 = A .* A**

Matrix Operations

Shortcut: Transposing Matrices

- **The transpose of a matrix is the matrix formed by interchanging the rows and columns of a given matrix**

```
A = [1 2 4 5      B = [1
     6 3 8 2]          7
                       3
                       3]
>> transpose(A)                    >> B'
A = [1 6                              B = [1 7 3 3]
     2 3
     4 8
     5 2]
```

Matrix Operations

Other useful functions:

- inv(A)
- mean(A)
- std(A)
- size(A)
- ones and zeros
- sort(A)
- sortrows(A)
- rand

Relational Operators

- **Relational operators are used to compare two scaler values or matrices of equal dimensions**

Relational Operators

<	**less than**
<=	**less than or equal to**
>	**Greater than**
>=	**Greater than or equal to**
==	**equal**
~=	**not equal**

Relational Operators

- **Comparison occurs between pairs of corresponding elements**
- **A 1 or 0 is returned for each comparison indicating TRUE or FALSE**
- **Matrix dimensions must be equal!**

```
>> 5 == 5
Ans 1
>> 20 >= 15
Ans 1
```

Relational Operators

```
    A = [1 2 4 5       B = 7       C = [2 2 2 2
         6 3 8 2]                       2 2 2 2]
```

Try:

```
>>A > B
>> A < C
```

Relational Operators

The Find Function

```
A = [1 2 4 5      B = 7      C = [2 2 2 2      D = [0 2 0 5 0 2]
     6 3 8 2]                     2 2 2 2]
```

- The ‘*find*’ function can also return the row and column indexes of of matching elements by specifying row and column arguments

```
>> [x,y] = find(A == 5)
```

- The matching elements will be indexed by (x1,y1), (x2,y2), …

```
>> A(x,y) = 10
A = [ 1 2 4 10
      6 3 8 2  ]
```

Working with files

- **Matlab can work with almost all types of files designed to store numbers**
  - **text files**
  - **xls,xlsx**
  - **csv**

- **But the simplest way to store/load data in Matlab is using its own version of data file**
  - **mat**

Control and Flow

- **Control flow capability enables matlab to function beyond the level of a simple desk calculator**
- **With control flow statements, matlab can be used as a complete high-level matrix language**
- **Flow control in matlab is performed with condition statements and loops**

Matlab Scripts

Advantages of M-files

- Easy editing and saving of work
- Undo changes
- Readability/Portability - non executable comments can be added using the ‘%’ symbol to make commands easier to understand
- Saving M-files is far more memory efficient than saving a workspace
- run(‘script name’)

Condition Statements

- It is often necessary to only perform matlab operations when certain conditions are met
- Relational and Logical operators are used to define specific conditions
- Simple flow control in matlab is performed with the ‘*If*’, ‘*Else*’, ‘*Elseif*’ and ‘*Switch*’ statements

Condition Statements

If, Else, and Elseif

- An if statement evaluates a logical expression and evaluates a group of commands when the logical expression is true
- The list of conditional commands are terminated by the end statement
- If the logical expression is false, all the conditional commands are skipped
- Execution of the script resumes after the end statement

Basic form:

```
if logical_expression
   commands
end
```

Condition Statements

Example

```
A = 6              B = 0

if A > 3
    D = [1 2 6]
     A = A + 1
elseif A > 2
    D = D + 1
    A = A + 2
end
```

What is evaluated in the code above?

Condition Statements

Switch

- The switch statement can act as many elseif statements
- Only the one case statement who’s value satisfies the original expression is evaluated

Basic form:

```
switch expression (scalar or string)
    case value 1
        commands 1
    case value 2
        commands 2
    case value n
        commands n
end
```

Condition Statements

Example

```
A = 6              B = 0

switch A
    case 4
    D = [ 0 0 0]
        A = A - 1
    case 5
        B = 1
    case 6
        D = [1 2 6]
         A = A + 1
end
```

** Only case 6 is evaluated

Loops

- Loops are an important component of flow control that enables matlab to repeat multiple statements in specific and controllable ways
- Simple repetition in matlab is controlled by two types of loops:
    1. For loops
    2. While loops

Loops

For Loops

- The for loop executes a statement or group of statements a predetermined number of times

Basic Form:

```
for index = start:increment:end
    statements
end
```

** If ‘*increment*’ is not specified, an increment of 1 is assumed by Matlab

Loops

For Loops

- Loops can be nested in other loops

```
A = [ ]
for i = 1:m
    for j = 1:n
        A(i,j) = i + j
    end
end
```

- Creates an m by n matrix A whose elements are the sum of their matrix position

Loops

Parallel For Loops

- Loops can be executed in parallel (if possible)

```
A = ones[1000 , 1000 ]
parfor i = 1:1000
    for j = 1:1000
        A(i,j) = i + j
    end
end
```

Loops

While Loops

- **The while loop executes a statement or group of statements repeatedly as long as the controlling expression is true**

Basic Form:

```
while expression
    statements
end
```

Loops

While Loops

Examples:

```
A = 6  B = 15
while A > 0 & B < 10
        A = A + 1
        B = B – 2
end
```

- Iteratively increase A and decrease B until the two conditions of the while loop are met

** Be very careful to ensure that your while loop will eventually reach its termination condition to prevent an infinite loop

Loops

Breaking out of loops

- The ‘*break*’ command instantly terminates a for and while loop
- When a break is encountered by matlab, execution of the script continues outside and after the loop

Loops

Breaking out of loops

Example:

```
A = 6  B = 15
count = 1
while A > 0 & B < 10
        A = A + 1
        B = B + 2
        count = count + 1
        if count > 100
                break
        end
end
```

- Break out of the loop after 100 repetitions if the while condition has not been met

Other Data Types

- String
  - A = ‘a test string’
  - strcat(A, ‘another string’)
  - strcmp(A, ‘a string’)
  - findstr(A , ‘est’)
  - num2str(234)
  - lower(A)
  - upper(A)
  - sprintf(‘this is %d out of %f’ , 12^2, sqrt(45))
- Struct
- Cell
- Map

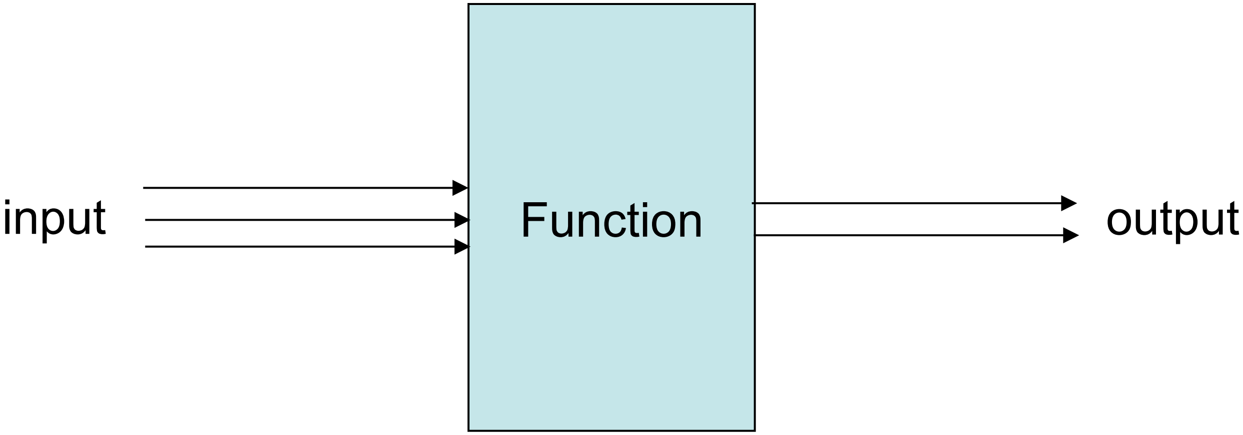
Figures

- plot a line
  - hold on
  - set color
  - plot tools
  - close
- hist
  - set number of bins
- scatter
- 3D plots
- surf

Functions in Matlab

- In Matlab, each function is a .m file
  - It is good protocol to name your .m file the same as your function name, i.e. funcname.m

- function *outargs*=*funcname*(*inargs*)


input
Function
output