Statistical Pattern Recognition

An Introduction to MATLAB

Hamid R. Rabiee
Jafar Muhammadi, Mohammad R. Zolfaghari

Spring 2012
http://ce.sharif.edu/courses/90-91/2/ce725-1/
Agenda

✧ MATrix LABoratory
✧ Environment and desktop
✧ Basics
✧ Programming
✧ Graphics
MATrix LABoratory

✦ www.mathworks.com

✦ Advantages of MATLAB
  ✦ Ease of use
  ✦ Platform independence
  ✦ Predefined functions
  ✦ Plotting

✦ Disadvantages of MATLAB
  ✦ Can be slow
  ✦ Expensive
MATLAB Components

✧ The MATLAB language
  ✧ a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features.

✧ The MATLAB working environment
  ✧ the set of tools and facilities that you work with as the MATLAB user or programmer

✧ Handle Graphics
  ✧ the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics.

✧ The MATLAB function library
  ✧ a vast collection of computational algorithms

✧ The MATLAB Application Program Interface (API)
  ✧ a library that allows you to write C, Java and Fortran programs that interact with MATLAB.
MATLAB Desktop

Launch Pad

Command Window

History
MATLAB Desktop

Workspace

Command Window

Current Directory

To get started, select "MATLAB Help" from the menu.
MATLAB Help

- **Learning MATLAB**
  - **Getting Started** - Introduction to MATLAB.
  - **Using MATLAB** - User guides for all of MATLAB.
  - **Index of Documentation Examples** - Major examples in the MATLAB documentation.

- **Finding Functions and Properties**
  - **MATLAB Functions Listed by Category** - Browse MATLAB functions by category.

- **Printing the Documentation**
  - **Handle Graphics Property Browser** - View descriptions of all graphics object properties.
MATLAB Editor

```
function kg=assemble(k,nodes,kg)

% ASSEMBLE will assemble k with nodes into kg (global kg)
% kg=ASSEMBLE(k,dofs,kg)
% k=element k
% dofs=nodes numbers in k (es: [1 5 3])
% kg=global kg

    gdofs=dofs(nodes);
    for kr=1:6
        for kc=1:6
            ir=gdofs(kr);
            ic=gdofs(kc);
            kg(ir,ic)=kg(ir,ic)+k(kr,kc);
        end
    end
```

Color keyed text with auto indents

Tabbed sheets for other files being edited
MATLAB Basics

✧ A program can be input
  ✧ command by command using the command line (lines starting with “»” on the MATLAB desktop)
  ✧ as a series of commands using a file
    (a special file called M-file)

✧ If a command is followed by a semicolon (;), result of the computation is not shown on the command window
Data Types

✧ All numbers are double precision
✧ Text is stored as arrays of characters
✧ You don’t have to declare the type of data (defined when running)
✧ MATLAB is case-sensitive!!!
Data Types

ARRAY
[full or sparse]

logical char NUMERIC cell structure function handle

user classes java classes

int8, uint8, int16, uint16, int32, uint32, int64, uint64

single double
Variables

- Variable is a name given to a reserved location in memory
  - `class_code = 111;`
  - `number_of_students = 65;`
  - `name = 'Jafar Muhammadi';`
  - `radius = 5;`
  - `area = pi * radius^2;`

- Use meaningful names for variables

- MATLAB variable names
  - must begin with a letter
  - can contain any combination of letters, numbers and underscore (_)
  - must be unique in the first 31 characters

- MATLAB is case sensitive: “name”, “Name” and “NAME” are considered different variables

- Never use a variable with the same name as a MATLAB command

- Naming convention: use lowercase letters
Variables

✧ Initialization using assignment statements
  ✧ \( x = 5 \)
    \[
    \begin{align*}
    x &= 5 \\
    y &= x + 1 \\
    \text{vector} &= [1 \ 2 \ 3 \ 4] \\
    \text{matrix} &= \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}
    \end{align*}
    \]
  ✧ \( a = [5 (2+4)] \)
    \[
    \begin{align*}
    a &= 5 \ 6
    \end{align*}
    \]

EVERYTHING IN MATLAB IS A MATRIX
Variables

- **Initialization using shortcut statements**
  - **colon operator → first:increment:last**
    - $x = 1:2:10$
      
      \[
      x = \\
      1 \ 3 \ 5 \ 7 \ 9
      \]
    - $y = 0:0.1:0.5$
      
      \[
      y = \\
      0 \ 0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5
      \]
Variables

- transpose operator $\rightarrow ' $
  - $ u = [ 1:3 ]' $
    
    $ u = 
    1 
    2 
    3 
  - $ v = [ u \ u ] $
    
    $ v = 
    1 1 
    2 2 
    3 3 
  - $ v = [ u'; u' ] $
    
    $ v = 
    1 2 3 
    1 2 3 $
Variables

- Initialization using built-in functions
  - zeros()
    - x = zeros(2)
      x =
      0 0
      0 0
    - z = zeros(2,3)
      z =
      0 0 0
      0 0 0
  - ones(), size(), length()
    - y = zeros(1,4)
      y =
      0 0 0 0
    - t = zeros( size(z) )
      t =
      0 0 0
      0 0 0
Variables

✧ Initialization using keyboard input
  ✧ input()
    ✧ value = input( 'Enter an input value: ' )
      Enter an input value: 1.25
      value =
      1.2500
    ✧ name = input( 'What is your name: ', 's' )
      What is your name: Selim
      name =
      Selim
## Matrices

- **Some useful commands:**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zeros(n)</td>
<td>Returns a n X n matrix of zeros</td>
</tr>
<tr>
<td>zeros(m,n)</td>
<td>Returns a m X n matrix of zeros</td>
</tr>
<tr>
<td>ones(n)</td>
<td>Returns a n X n matrix of ones</td>
</tr>
<tr>
<td>ones(m,n)</td>
<td>Returns a m X n matrix of ones</td>
</tr>
<tr>
<td>size(A)</td>
<td>For a m X n matrix A, returns the row vector [m,n] containing the number of rows and columns in matrix</td>
</tr>
<tr>
<td>length(A)</td>
<td>Returns the larger of the number of rows or columns in A</td>
</tr>
</tbody>
</table>
Matrices, Colon Operator

✧ Creating new matrices from an existing matrix

\[ C = \begin{bmatrix} 1,2,5; -1,0,1; 3,2,-1; 0,1,4 \end{bmatrix} \]
\[ F = C(:, 2:3) = \begin{bmatrix} 2,5; 0,1; 2,-1; 1,4 \end{bmatrix} \]
Creating new matrices from an existing matrix

\[ C = \begin{bmatrix} 1 & 2 & 5 \\ -1 & 0 & 1 \\ 3 & 2 & -1 \\ 0 & 1 & 4 \end{bmatrix} \]

\[ E = C(2:3,:) = \begin{bmatrix} -1 & 0 & 1 \\ 3 & 2 & -1 \end{bmatrix} \]
Creating new matrices from an existing matrix

\[
C = \begin{bmatrix}
1 & 2 & 5 \\
-1 & 0 & 1 \\
3 & 2 & -1 \\
0 & 1 & 4
\end{bmatrix}
\]

\[
G = C(3:4, 1:2) = \begin{bmatrix}
3 & 2 \\
0 & 1
\end{bmatrix}
\]
Matrices, Colon Operator

✧ Variable_name = a:step:b
  ✧ time = 0.0:0.5:2.5
  ✧ time = [0.0, 0.5, 1.0, 1.5, 2.0, 2.5]

✧ Negative increment
  ✧ values = 10:-1:2
  ✧ values = [10, 9, 8, 7, 6, 5, 4, 3, 2]

✧ Whole row
  ✧ y = x ( 3 , : )

✧ Whole column
  ✧ y = x ( :, 3 )
Subarrays

✧ Array indices start from 1

✧ \( x = [-2 \ 0 \ 9 \ 1 \ 4] \) same as \( a = [-2,0,9,1,4] \)

✧ \( x(2) \)
  \[
  \text{ans} =
  \begin{array}{c}
  0
  \end{array}
  \]

✧ \( x(4) \)
  \[
  \text{ans} =
  \begin{array}{c}
  1
  \end{array}
  \]

✧ \( x(8) \)
  \[
  ??? \text{ Error}
  \]

✧ \( x(-1) \)
  \[
  ??? \text{ Error}
  \]
Subarrays

y = [ 1 2 3; 4 5 6 ];

- y(1,2)
  ans =
  2
- y(2,1)
  ans =
  4
- y(2)
  ans =
  4 (column major order)
Subarrays

- \( y = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \);
- \( y(1,:) \)
  \[
  \text{ans} = \\
  \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}
  \]
- \( y(:,2) \)
  \[
  \text{ans} = \\
  \begin{bmatrix} 2 \\ 5 \end{bmatrix}
  \]
- \( y(2,1:2) \)
  \[
  \text{ans} = \\
  \begin{bmatrix} 4 & 5 \end{bmatrix}
  \]
- \( y(1,2:end) \)
  \[
  \text{ans} = \\
  \begin{bmatrix} 2 & 3 \end{bmatrix}
  \]
- \( y(:,2:end) \)
  \[
  \text{ans} = \\
  \begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}
  \]
Subarrays

\[ x = \begin{bmatrix} -2 & 0 & 9 & 1 & 4 \end{bmatrix}; \]

\[ x(2) = 5 \]
\[ x = \begin{bmatrix} -2 & 5 & 9 & 1 & 4 \end{bmatrix} \]
\[ x(4) = x(1) \]
\[ x = \begin{bmatrix} -2 & 5 & 9 & -2 & 4 \end{bmatrix} \]
\[ x(8) = -1 \]
\[ x = \begin{bmatrix} -2 & 5 & 9 & -2 & 4 & 0 & 0 & -1 \end{bmatrix} \]
Subarrays

✧ \( y = [1 \ 2 \ 3; \ 4 \ 5 \ 6] \);

✧ \( y(1,2) = -5 \)

\[
y = \\
1 \ -5 \ 3 \\
4 \ 5 \ 6
\]

✧ \( y(2,1) = 0 \)

\[
y = \\
1 \ -5 \ 3 \\
0 \ 5 \ 6
\]

✧ \( y(1,2:end) = [-1 \ 9] \)

\[
y = \\
1 \ -1 \ 9 \\
0 \ 5 \ 6
\]
Subarrays

- \( y = [ 1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9 ]; \)
  - \( y(2:end,2:end) = 0 \)
    \[
    \begin{array}{ccc}
    1 & 2 & 3 \\
    4 & 0 & 0 \\
    7 & 0 & 0 \\
    \end{array}
    \]
  - \( y(2:end,2:end) = [ -1 \ 5 ] \)
    
    ??? Error
  - \( y(2,[1 \ 3]) = -2 \)
    \[
    \begin{array}{ccc}
    1 & 2 & 3 \\
    -2 & 0 & -2 \\
    7 & 0 & 0 \\
    \end{array}
    \]

Deleting rows and columns: \( y(:,2) = [] \), \( y(2,3) = [] \)
Special Values

- **pi**: \( \pi \) value up to 15 significant digits
- **i, j**: \( \sqrt{-1} \)
- **Inf**: infinity (such as division by 0)
- **NaN**: Not-a-Number (such as division of zero by zero)
- **clock**: current date and time as a vector
- **date**: current date as a string (e.g. 16-Feb-2004)
- **eps**: epsilon
- **ans**: default variable for answers
Displaying Data

✧ The disp( array ) function
  ✧ disp( 'Hello' );
    Hello
  ✧ disp(5);
    5
  ✧ disp( [ 'Bilkent ' 'University' ] );
    Bilkent University
  ✧ name = 'Selim'; disp( [ 'Hello ' name ] );
    Hello Selim
Displaying Data

- The `num2str()` and `int2str()` functions
  
  - `d = [ num2str(16) ' -Feb-' num2str(2004) ];`
  
  - `disp(d);`
    
    16-Feb-2004
  
  - `x = 23.11;`
  
  - `disp( [ 'answer = ' num2str(x) ] );`
    
    answer = 23.11
  
  - `disp( [ 'answer = ' int2str(x) ] );`
    
    answer = 23
Displaying Data

- The `fprintf(format, data)` function
  - `%d` integer
  - `%f` floating point format
  - `%e` exponential format
  - `\n` new line character
  - `\t` tab character
Displaying Data

- `fprintf( 'Result is %d', 3 );`
  Result is 3
- `fprintf( 'Area of a circle with radius %d is %f', 3, pi*3^2 );`
  Area of a circle with radius 3 is 28.274334
- `x = 5;`
- `fprintf( 'x = %3d', x );`
  x = 5
- `x = pi;`
- `fprintf( 'x = %0.2f', x );`
  x = 3.14
- `fprintf( 'x = %6.2f', x );`
  x = 3.14
- `fprintf( 'x = %d\ny = %d\n', 3, 13 );`
  x = 3
  y = 13
Data Files

- `save filename var1 var2 ...`
  - `save homework.mat x y` → binary
  - `save x.dat x –ascii` → ascii

- `load filename`
  - `load filename.mat` → binary
  - `load x.dat –ascii` → ascii
MATLAB Basics: Scalar Operations

- variable_name = expression;
  - addition: \( a + b \) → \( a + b \)
  - subtraction: \( a - b \) → \( a - b \)
  - multiplication: \( a \times b \) → \( a \times b \)
  - division: \( a / b \) → \( a / b \)
  - exponent: \( a^b \) → \( a^b \)
  - Element by element multiplication: → \( a.* b \)
MATLAB Basics: Scalar Operations

- \( x = 3 \times 2 + 6 / 2 \)
  - \( x = ? \)

- Processing order of operations is important
  - parenthesis (starting from the innermost)
  - exponentials (left to right)
  - multiplications and divisions (left to right)
  - additions and subtractions (left to right)

- \( x = 3 \times 2 + 6 / 2 \)
  - \( x = 9 \)
# Arithmetic Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>.*</td>
<td>Multiplication (element wise)</td>
</tr>
<tr>
<td>./</td>
<td>Right division (element wise)</td>
</tr>
<tr>
<td>.\</td>
<td>Left division (element wise)</td>
</tr>
<tr>
<td>+</td>
<td>Unary plus</td>
</tr>
<tr>
<td>-</td>
<td>Unary minus</td>
</tr>
<tr>
<td>:</td>
<td>Colon operator</td>
</tr>
<tr>
<td>.^</td>
<td>Power (element wise)</td>
</tr>
<tr>
<td>'</td>
<td>Transpose</td>
</tr>
<tr>
<td>*</td>
<td>Matrix multiplication</td>
</tr>
<tr>
<td>/</td>
<td>Matrix right division</td>
</tr>
<tr>
<td>\</td>
<td>Matrix left division</td>
</tr>
<tr>
<td>^</td>
<td>Matrix power</td>
</tr>
</tbody>
</table>
## Relational and Logical Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>Returns 1 for every element location that is true (nonzero) in both arrays, and 0 for all other elements.</td>
</tr>
<tr>
<td></td>
<td>Returns 1 for every element location that is true (nonzero) in either one or the other, or both, arrays and 0 for all other elements.</td>
</tr>
<tr>
<td>~</td>
<td>Complements each element of input array, A.</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>~=</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>
## Matrices

### More commands:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Command(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transpose</td>
<td>B = A'</td>
</tr>
<tr>
<td>Identity Matrix</td>
<td>eye(n) -&gt; returns an n X n identity matrix</td>
</tr>
<tr>
<td></td>
<td>eye(m,n) -&gt; returns an m X n matrix with ones on the main diagonal and zeros elsewhere</td>
</tr>
<tr>
<td>Addition and Subtraction</td>
<td>C = A + B, C = A - B</td>
</tr>
<tr>
<td>Scalar Multiplication</td>
<td>B = ( \alpha ) A, where ( \alpha ) is a scalar</td>
</tr>
<tr>
<td>Matrix Multiplication</td>
<td>C = A * B</td>
</tr>
<tr>
<td>Matrix Inverse</td>
<td>B = \text{inv}(A), A must be a square matrix in this case</td>
</tr>
<tr>
<td>Matrix powers</td>
<td>B = A * A, A must be a square matrix</td>
</tr>
<tr>
<td>Determinant</td>
<td>\text{det}(A), A must be a square matrix</td>
</tr>
<tr>
<td>Reshape matrix</td>
<td>\text{reshape}(A,row,col)</td>
</tr>
</tbody>
</table>
MATLAB Basics: Built-in Functions

✧ result = function_name( input );
  ✧ abs, sign
  ✧ log, log10, log2
  ✧ exp
  ✧ sqrt
  ✧ sin, cos, tan
  ✧ asin, acos, atan
  ✧ max, min, mean, median, std, var, sum
  ✧ round, floor, ceil, fix
  ✧ mod, rem
  ✧ gcd
  ✧ sort

✧ help elfun
MATLAB Basics: Useful Commands

- `help command` → Online help
- `lookfor keyword` → Lists related commands
- `which` → Version and location info
- `clear` → Clears the workspace
- `clc` → Clears the command window
- `diary filename` → Sends output to file
- `who, whos` → Lists content of the workspace
- `more on/off` → Enables/disables paged output
- `Ctrl+c` → Aborts operation
- `...` → Continuation
- `%` → Comments
- `Doc` → Help Browser
The programming environment

- MATLAB can’t tell if identifier is variable or function
  - >> z = theta;

- MATLAB searches for identifier in the following order
  1. variable in current workspace
  2. built-in variable
  3. built-in m-file
  4. m-file in current directory
  5. m-file on search path

- Note: m-files can be located in current directory, or in path
Script files

- Script-files contain a sequence of MATLAB commands

  ```matlab
  %FACTSCRIPT - Compute n-factorial, n!=1*2*...*n
  y = prod(1:n);
  ``

- Executed by typing its name
  - `>> factscript`

- Operates on variables in global workspace
  - Variable `n` must exist in workspace
  - Variable `y` is created (or over-written)

- Use comment lines (starting with `%`) to document file!
Functions

Functions describe subprograms

- Take inputs, generate outputs
- Have local variables (invisible in global workspace)

\[ \text{[output_arguments]} = \text{function_name(input_arguments)} \]

% Comment lines
<function body>

```
function [z]=factfun(n)
% FACTFUN - Compute factorial
% Z=FACTFUN(N)
z = prod(1:n);
```

```
>> y=factfun(10);
```
Scripts or function: when use what?

✧ Functions
✧ Take inputs, generate outputs, have internal variables
✧ Solve general problem for arbitrary parameters

✧ Scripts
✧ Operate on global workspace
✧ Document work, design experiment or test
✧ Solve a very specific problem once

✧ Exam: all problems will require you to write functions

% FACTTEST – Test factfun
N=50;
y=factfun(N);
Flow control - selection

✧ The if-elseif-else construction

if <logical expression>
   <commands>
elseif <logical expression>
   <commands>
else
   <commands>
end

if height>170
   disp('tall')
elseif height<150
   disp('small')
else
   disp('average')
end
Logical expressions

✧ Relational operators (compare arrays of same sizes)

  == (equal to)  ~ (not equal)
  < (less than)  <= (less than or equal to)
  > (greater than)  >= (greater than or equal to)

✧ Logical operators (combinations of relational operators)

  & (and)
  | (or)
  ~ (not)

✧ Logical functions

  xor
  isempty
  any
  all

  if (x>=0) & (x<=10)
      disp('x is in range [0,10]')
  else
      disp('x is out of range')
  end
Flow control repetition

- Repeats a code segment a **fixed** number of times

```matlab
for index=<vector>
    <statements>
end
```

The `<statements>` are executed repeatedly.
At each iteration, the variable index is assigned a new value from `<vector>`.

```matlab
for k=1:12
    kfac=prod(1:k);
    disp([num2str(k), ' ', num2str(kfac)])
end
```
Example – selection and repetition

function y=fact(n)
% FACT – Display factorials of integers 1..n
if nargin < 1
    error('No input argument assigned')
elseif n < 0
    error('Input must be non-negative')
elseif abs(n-round(n)) > eps
    error('Input must be an integer')
end

for k=1:n
    kfac=prod(1:k);
    disp([num2str(k),’ ',num2str(kfac)])
    y(k)=kfac;
end;
Flow control – conditional repetition

♢ while-loops

```matlab
k=1;
while prod(1:k)~=Inf,
    k=k+1;
end
disp(['Largest factorial in MATLAB:',num2str(k-1)]);
```
Programming tips and tricks

- Programming style has huge influence on program speed!

Loops are slow: Replace loops by vector operations!
Memory allocation takes a lot of time: Pre-allocate memory!
Use profile to find code bottlenecks!
Example

✧ What does the following code do?

A = [1, 2, 3, 4, 5, 1, 1]
Inx = (A == 1)
B = A(Inx)
Ind = find(A == 1)
Isempy(Ind)
<table>
<thead>
<tr>
<th>Topic</th>
<th>Command/Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Equation</td>
<td>( A = Xb ) \rightarrow \ X = A \backslash b )</td>
</tr>
<tr>
<td>null space of ( A )</td>
<td>( z = \text{null}(A) )</td>
</tr>
<tr>
<td>Inverse of matrix</td>
<td>( \text{Inv}(A) )</td>
</tr>
<tr>
<td>Determinant</td>
<td>( \text{det}(A) )</td>
</tr>
<tr>
<td>Eigen value and eigen vector</td>
<td>( [V, D] = \text{eig}(A) )</td>
</tr>
<tr>
<td>LU factorization</td>
<td>( [L, U, P] = \text{lu}(A) )</td>
</tr>
<tr>
<td>Cholesky factorization</td>
<td>( \text{chol}(A) )</td>
</tr>
<tr>
<td>( B = \text{sparse}(A), \ C = \text{full}(B) )</td>
<td></td>
</tr>
</tbody>
</table>
Example: Curve Fitting

✧ Consider the following code

\[
x = (0: 0.1: 2.5)';
\]
\[
y = \text{erf}(x);
\]
\[
p = \text{polyfit}(x,y,6)
\]
\[
x = (0: 0.1: 5)';
\]
\[
y = \text{erf}(x);
\]
\[
f = \text{polyval}(p,x);
\]
\[
\text{plot}(x,y,'o',x,f,'-')
\]
Symbolic Toolbox

- syms x y z mu
- r = sqrt(x^2 + y^2 + z^2)
- U = mu/r
- acell_x = diff(U,x)
- acell_y = diff(U,y)
- acell_z = diff(U,z)

\[ r = \left( x^2 + y^2 + z^2 \right)^{1/2} \]
\[ U = \frac{\mu}{\left( x^2 + y^2 + z^2 \right)^{1/2}} \]
\[ acell_x = -\frac{\mu}{\left( x^2 + y^2 + z^2 \right)^{3/2}} \times x \]
\[ acell_y = -\frac{\mu}{\left( x^2 + y^2 + z^2 \right)^{3/2}} \times y \]
\[ acell_z = -\frac{\mu}{\left( x^2 + y^2 + z^2 \right)^{3/2}} \times z \]
MATLAB Graphics

✧ 2-D Plotting
✧ Sub-Plots
✧ 3-D Plotting
  ✧ contourf-colorbar-plot3-waterfall-contour3-mesh-surf
✧ Specialized Plotting-bar
  ✧ 3h-hist-area-pie3-rose
MATLAB Graphics

\[ x = 0:pi/100:2*pi; \]
\[ y = \sin(x); \]
\[ \text{plot}(x,y) \]
\[ \text{xlabel('x = 0:2\pi')} \]
\[ \text{ylabel('Sine of x')} \]
\[ \text{title('Plot of the Sine Function')} \]
t = 0:pi/100:2*pi;
y1=sin(t);
y2=sin(t+pi/2);
plot(t,y1,t,y2)
grid on
MATLAB Graphics

t = 0:pi/100:2*pi;
y1=sin(t);
y2=sin(t+pi/2);
subplot(2,2,1)
plot(t,y1)
subplot(2,2,2)
plot(t,y2)
MATLAB Graphics

\[ x = \text{rand}(100,1); \]
\[ \text{stem}(x); \]

\[ \text{hist}(x, 100) \]
MATLAB Graphics

mu = [2 3];
SIGMA = [1 1.5; 1.5 3];
r = mvnrnd(mu,SIGMA,100);
plot(r(:,1),r(:,2),'+')

x=randn(100,2);
plotmatrix(x);figure(gcf);
MATLAB Graphics

\[ k=0:20; \]
\[ y=\text{binopdf}(k, 20, 0.5); \]
\[ \text{stem}(k, y) \]

\[ k=0:20; \]
\[ y=\text{binopdf}(k, 20, 0.2); \]
\[ \text{stem}(k, y) \]
MATLAB Graphics

```matlab
x = 0:0.1:2;
y = 0:0.1:2;
[xx, yy] = meshgrid(x, y);
zz = sin(xx.^2 + yy.^2);
surf(xx, yy, zz)
xlabel('X axes')
ylabel('Y axes')
```
MATLAB Graphics

Peaks Function - (CONTOURF & COLORBAR)

Sinc Function - (PLOT3)

L-shaped Membrane - (WATERFALL)

Peaks Function - (CONTOUR3)

Sinc Function - (MESH)

L-shaped Membrane - (SURF)
MATLAB Graphics
Give it a try!

Thank you!

Spring 2012

http://ce.sharif.edu/courses/90-91/2/ce725-1/