MATLAB Tutorial

Based on IPLab@SUT
Matlab Tutorial

Contents

Contents Continued

What is MATLAB?

- high-performance software
  - ***Computation***
  - ***Visualization***
  - ***Easy-to-use environment***.
- high-level language
  - ***Data types***
  - ***Functions***
  - ***Control flow statements***
  - ***Input/output***
  - ***Graphics***
  - ***Object-oriented*** programming capabilities

Calculations at the Command Line

MATLAB as a calculator

```
» -5/(4.8+5.32)^2
ans =
   -0.0488
» (3+4i)*(3-4i)
ans =
    25
» cos(pi/2)
ans =
  6.1230e-017
» exp(acos(0.3))
ans =
    3.5470
```

Assigning Variables

```
» a = 2;
» b = 5;
» a^b
ans =
    32
» x = 5/2*pi;
» y = sin(x)
y =
     1
» z = asin(y)
z =
    1.5708
```

Semicolon suppresses screen output

Results assigned to "ans" if name not specified

() parentheses for function inputs

A Note about Workspace:
Numbers stored in double-precision floating point format

General Functions

- whos: List current variables
- clear: Clear variables and functions from memory
- cd: Change current working directory
- ls: List files in directory

Getting help

- *help* command (`>>help`)
- *lookfor* command (`>>lookfor`)
- Printable Documents
  - "Matlabroot\help\pdf_doc\"

Matrices

- Entering and Generating Matrices
- Subscripts
- Scalar Expansion
- Concatenation
- Deleting Rows and Columns
- Array Extraction
- Matrix and Array Multiplication

Entering Numeric Arrays

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

```
» a=[1 2;3 4]
a =
     1     2
     3     4
» b=[-2.8, sqrt(-7), (3+5+6)*3/4]
b =
   -2.8000    0 + 2.6458i   10.5000
» b(2,5) = 23
b =
   -2.8000  0 + 2.6458i  10.5000   0         0
         0            0         0   0  23.0000
```

Use square brackets [ ]

Row separator semicolon (;)

Column separator space / comma (,)

- Any MATLAB expression can be entered as a matrix element
- Matrices must be rectangular. (Set undefined elements to zero)

The Matrix in MATLAB

Columns (n)

A =

Rows (m)	1	2	3	4	5
1	4	10	1	6	2
2	8	1.2	9	4	25
3	7.2	5	7	1	11
4	0	0.5	4	5	56
5	23	83	13	0	10

A (2,4)

A (17)

Rectangular Matrix:
Scalar: 1-by-1 array
Vector: m-by-1 array
1-by-n array
Matrix: m-by-n array

Entering Numeric Arrays

Scalar expansion

Creating sequences: colon operator (:)

Utility functions for creating matrices.

```
» w=[1 2;3 4] + 5
w =
     6     7
     8     9
» x = 1:5
x =
     1     2     3     4     5
» y = 2:-0.5:0
y =
    2.0000    1.5000    1.0000    0.5000         0
» z = rand(2,4)
z =
    0.9501    0.6068    0.8913    0.4565
    0.2311    0.4860    0.7621    0.0185
```


Numerical Array Concatenation

Use [ ] to combine existing arrays as matrix "elements"

Row separator: semicolon (;)

Column separator: space / comma (,)

```
» a=[1 2;3 4]
a =
     1     2
     3     4
» cat_a=[a, 2*a; 3*a, 4*a; 5*a, 6*a]
cat_a =
     1     2     2     4
     3     4     6     8
     3     6     4     8
     9    12    12    16
     5    10     6    12
    15    20    18    24
```

Use square brackets [ ]

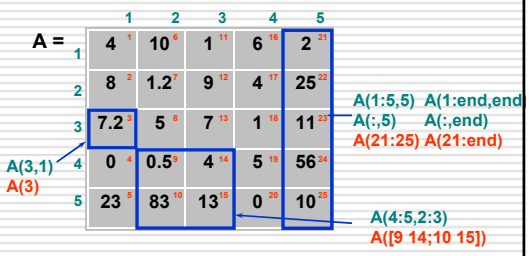
4*a

Note:
The resulting matrix must be rectangular

Deleting Rows and Columns

```
» A=[1 5 9;4 3 2.5; 0.1 10 3i+1]
A =
   1.0000            5.0000            9.0000
   4.0000            3.0000            2.5000
   0.1000           10.0000            1.0000+3.0000i
» A(:,2)=[]
A =
   1.0000            9.0000
   4.0000            2.5000
   0.1000            1.0000 + 3.0000i
» A(2,2)=[]
???  Indexed empty matrix assignment is not allowed.
```


Array Subscripting / Indexing



Matrix Multiplication

```
» a = [1 2 3 4; 5 6 7 8];          [2x4]
» b = ones(4,3);                   [4x3]
» c = a*b                  [2x4]*[4x3] → [2x3]
c =
    10    10    10
    26    26    26   ← a(2nd row).b(3rd column)
```

Array Multiplication

```
» a = [1 2 3 4; 5 6 7 8];
» b = [1:4; 1:4];
» c = a.*b
c =
     1     4     9    16
     5    12    21    32   ← c(2,4) = a(2,4)*b(2,4)
```


Matrix Manipulation Functions

- zeros: Create an array of all zeros
- ones: Create an array of all ones
- eye: Identity Matrix
- rand: Uniformly distributed random numbers
- diag: Diagonal matrices and diagonal of a matrix
- size: Return array dimensions
- repmat: Replicate and tile a matrix

Matrix Manipulation Functions

- det: Matrix determinant
- inv: Matrix inverse
- eig: Evaluate eigenvalues and eigenvectors
- rank: Rank of matrix

Elementary Math

- Logical Operators
- Math Functions
- Polynomial and Interpolation

Logical Operations

= =	equal to
>	greater than
<	less than
>=	Greater or equa
<=	less or equal
~	not
&	and
|	or
isfinite(), etc. . . .	
all(), any()	
find	

```
» Mass = [-2 10 NaN 30 -11 Inf 31];
» each_pos = Mass>=0
each_pos =
     0     1     0     1     0     1     1
» all_pos = all(Mass>=0)
all_pos =
     0
» all_pos = any(Mass>=0)
all_pos =
     1
» pos_fin = (Mass>=0)&(isfinite(Mass))
pos_fin =
     0     1     0     1     0     0     1
```

Note:
- 1 = TRUE
- 0 = FALSE

Elementary Math Function

- abs, sign: Absolute value and Signum Function
- sin, cos, asin, acos…: Triangular functions
- exp, log, log10: Exponential, Natural and Common (base 10) logarithm
- ceil, floor: Round toward infinities
- fix: Round toward zero

Elementary Math Function

- round: Round to the nearest integer
- sqrt: Square root function
- real, imag: Real and Image part of complex
- rem: Remainder after division

Elementary Math Function

- max, min: Maximum and Minimum of arrays
- mean, median: Average and Median of arrays
- std, var: Standard deviation and variance
- sort: Sort elements in ascending order
- sum, prod: Summation & Product of Elements

Importing and Exporting Data

- Using the Import Wizard
- Using ***Save*** and ***Load*** command

```
save fname
save fname x y z
save fname -ascii
save fname -mat
```

```
load fname
load fname x y z
load fname -ascii
load fname -mat
```


Input/Output for Text File

•Read formatted data, reusing the format string N times.

```
»[A1…An]=textread(filename,format,N)
```

•Import and Exporting **Numeric** Data with General ASCII delimited files

```
» M = dlmread(filename,delimiter,range)
```

Graphics Fundamentals

Graphics

- Basic Plotting
  plot, title, xlabel, grid, legend, hold, axis
- Editing Plots
  Property Editor
- Mesh and Surface Plots
  meshgrid, mesh, surf, colorbar, patch, hidden
- Handle Graphics

2-D Plotting

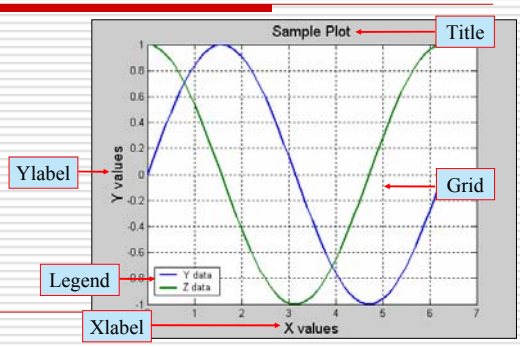
Syntax:

```
plot(x1, y1, 'clm1', x2, y2, 'clm2', ...)
```

Example:

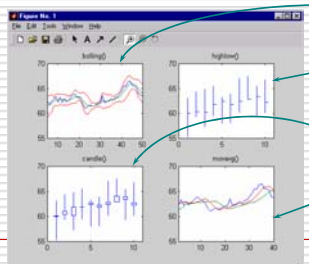
```
x=[0:0.1:2*pi];
y=sin(x);
z=cos(x);
plot(x,y,x,z,'linewidth',2)
title('Sample Plot','fontsize',14);
xlabel('X values','fontsize',14);
ylabel('Y values','fontsize',14);
legend('Y data','Z data')
grid on
```


Sample Plot



Subplots

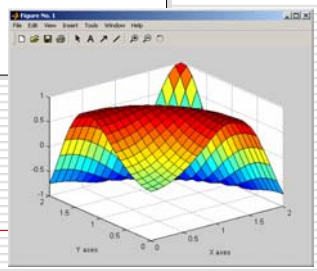
Syntax: `subplot(rows,cols,index)`



```
»subplot(2,2,1);
» ...

»subplot(2,2,2)
» ...

»subplot(2,2,3)
» ...

»subplot(2,2,4)
» ...
```

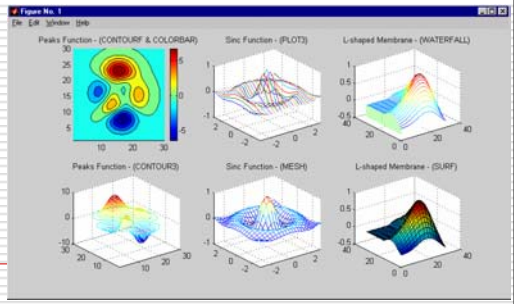

Surface Plot Example

```
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')
```



3-D Surface Plotting

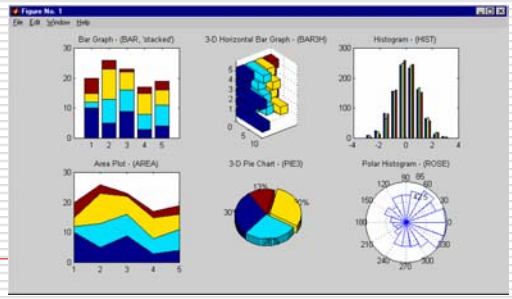
contourf-colorbar-plot3-waterfall-contour3-mesh-surf



41

Specialized Plotting Routines

bar-bar3h-hist-area-pie3-rose



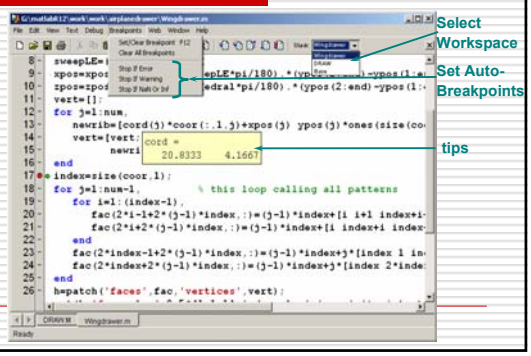
42

Editing and Debugging M-Files

- What is an M-File?
- The Editor/Debugger
- Search Path
- Debugging M-Files
  - Types of Errors (*Syntax Error* and *Runtime Error*)
  - Using ***keyboard*** and " **;** " statement
  - Setting Breakpoints
  - Stepping Through
    - Continue, Go Until Cursor, Step, Step In, Step Out
  - Examining Values
    - Selecting the Workspace
    - Viewing ***Datatips*** in the Editor/Debugger
    - Evaluating a Selection

43

Debugging



44

Programming and Dubugging

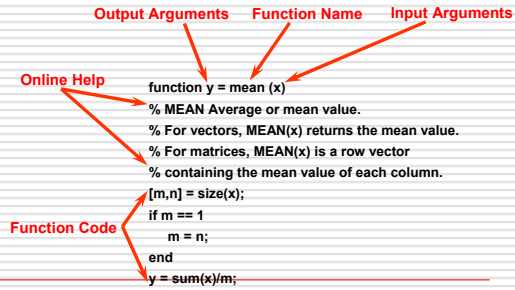
45

Script and Function Files

- Script Files
  - Work as though you typed commands into MATLAB prompt
  - Variable are stored in MATLAB workspace
- Function Files
  - Let you make your own MATLAB Functions
  - All variables within a function are **local**
  - All information must be passed to functions as parameters
  - Subfunctions are supported

46

Basic Parts of a Function M-File



Flow Control Statements

if Statement

```
if ((attendance >= 0.90) & (grade_average >= 60))
   pass = 1;
end;
```

while Loops

```
eps = 1;
while (1+eps) > 1
    eps = eps/2;
end
eps = eps*2
```


Flow Control Statements for Loop

```
a = zeros(k,k)  % Preallocate matrix
for m = 1:k
    for n = 1:k
        a(m,n) = 1/(m+n -1);
    end
end
```

switch Statement

```
method = 'Bilinear';
switch lower(method)
   case {'linear','bilinear'}
      disp('Method is linear')
   case 'cubic'
      disp('Method is cubic')
otherwise
      disp('Unknown method.')
end
Method is linear
```


M-file Programming Features

- SubFunctions
- Varying number of input/output arguments
- ***Local*** and ***Global*** Variables
- Obtaining User Input
  - Prompting for Keyboard *Input*
  - Pausing During Execution
- Errors and Warnings
  - Displaying *error* and *warning* Messages
- Shell Escape Functions (*! Operator*)
- Optimizing MATLAB Code
  - Vectorizing loops
  - Preallocating Arrays

Function M-file

```
function r = ourrank(X,tol)
% rank of a matrix
s = svd(X);
if (nargin == 1)
  tol = max(size(X)) * s(1)* eps;
end
r = sum(s > tol);
```

Multiple Input Arguments use ( )

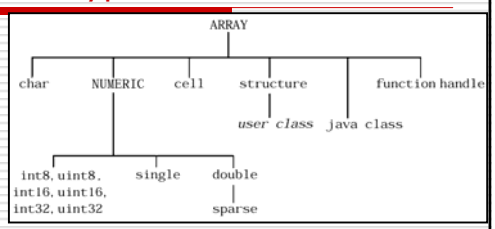
```
»r=ourrank(rand(5),.1);
```

Multiple Output Arguments, use [ ]

```
»[m std]=ourstat(1:9);
```

```
function [mean,stdev] = ourstat(x)
[m,n] = size(x);
if m == 1
  m = n;
end
mean = sum(x)/m;
stdev = sqrt(sum(x.^2)/m - mean.^2);
```


Data Types



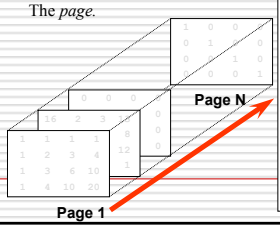
- Numeric Arrays
- Multidimensional Arrays
- Structures and Cell Arrays

Multidimensional Arrays

The first references array dimension 1, the row.

The second references dimension 2, the column.

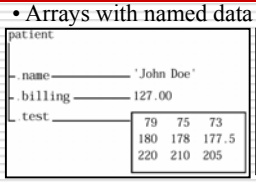
The third references dimension 3, The *page*.




```
» A = pascal(4);
» A(:,:,2) = magic(4)
A(:,:,1) =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
A(:,:,2) =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
» A(:,:,9) =
  diag(ones(1,4));
```


Structures

- Arrays with named data containers called ***fields***.




```
» patient.name='John Doe';
» patient.billing = 127.00;
» patient.test= [79 75 73;
180 178 177.5;
220 210 205];
```

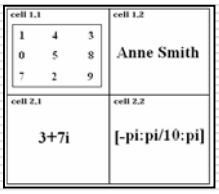
- Also, Build structure arrays using the ***struct*** function.
- Array of *structures*

```
» patient(2).name='Katty Thomson';
» Patient(2).billing = 100.00;
» Patient(2).test= [69 25 33; 120 128 177.5; 220
  210 205];
```


Cell Arrays

- Array for which the elements are *cell*s and can hold other MATLAB arrays of different types.

```
» A(1,1) = {[1 4 3;
0 5 8;
7 2 9]};
» A(1,2) = {'Anne Smith'};
» A(2,1) = {3+7i};
» A(2,2) = {-pi:pi/10:pi};
```



- Using braces {} to point to elements of cell array
- Using ***celldisp*** function to display cell array

Next Tutorial Session

- ☐ Image Processing Toolbox
- ☐ Movie Making
- ☐ Problem Set 1

Getting more help

- Contact http://www.mathworks.com/support
  - You can find more help and FAQ about mathworks products on this page.
- Getting started with Matlab
  - http://www.indiana.edu/~statmath/math/matlab/gettingstarted/index.html
- Matlab Primer
  - http://math.ucsd.edu/~driver/21d-s99/matlab-primer.html

Questions?

?