01203479 computer application in transportation

2553/2

Introduction to MATLAB

Prepared by Lect.Weerakaset Suanpaga

Outline

- Introduction and where to get MATLAB
- Data structure: matrices, vectors and operations
- Basic line plots
- File I/O

Where to get MATLAB

- FAS computing:
  - Download from http://fas.harvard.edu/computing/software/
  - You must be on FAS network to use MATLAB
- HSEAS IT
  - Maxwell Dworkin Rooms G107-G111
- Mathworks:
  - Student version is affordable and complete.

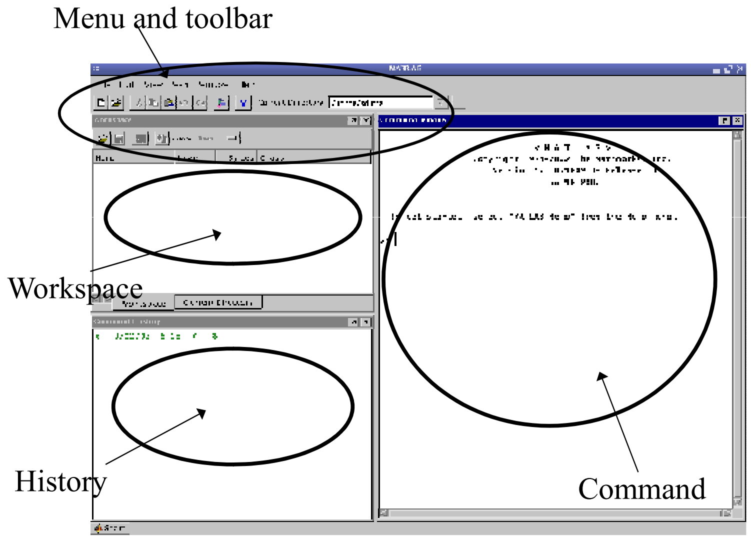
What is MATLAB

- High level language for technical computing
- Stands for **MAT**rix **LAB**oratory
- Everything is a matrix - easy to do linear algebra

The MATLAB System

- Development Environment
- Mathematical Function Library
- MATLAB language
- Application Programming Language (not discussed today)

MATLAB Desktop



Matrices & Vectors

- All (almost) entities in MATLAB are matrices
- Easy to define:

```
>> A = [16 3; 5 10]
  A =     16      3
           5     10
```

- Use ‘,’ or ‘ ’ to separate row elements -- use ‘;’ to separate rows

Matrices & Vectors - II

- Order of Matrix - $m \times n$
  - m=no. of rows, n=no. of columns
- Vectors - special case
  - n = 1 column vector
  - m = 1 row vector

Creating Vectors and Matrices

- Define

```
>> A = [16 3; 5 10]
  A =    16     3
          5    10
>> B = [3 4 5
  6 7 8]
  B = 3  4  5
      6  7  8
```

- Transpose

Vector :

```
>> a=[1 2 3];
>> a'
     1
     2
     3
```

Matrix:

```
>> A=[1 2; 3 4];
>> A'
ans =
     1     3
     2     4
```


Creating Vectors

Create vector with equally spaced intervals

```
>> x=0:0.5:pi
x =
 0 0.5000 1.0000 1.5000 2.0000 2.5000 3.0000
```

Create vector with *n* equally spaced intervals

```
>> x=linspace(0, pi, 7)
x =
  0 0.5236 1.0472 1.5708 2.0944 2.6180 3.1416
```

Equal spaced intervals in logarithm space

```
>> x=logspace(1,2,7)
x =
  10.0000 14.6780 21.5443 … 68.1292  100.0000
```

Note: MATLAB uses pi to represent π , uses i or j to represent imaginary unit

Creating Matrices

- `zeros(m, n)`: matrix with all zeros
- `ones(m, n)`: matrix with all ones.
- `eye(m, n)`: the identity matrix
- `rand(m, n)`: uniformly distributed random
- `randn(m, n)`: normally distributed random
- `magic(m)`: square matrix whose elements have the same sum, along the row, column and diagonal.
- `pascal(m)` : Pascal matrix.

Matrix operations

- `^`: exponentiation
- `*`: multiplication
- `/`: division
- `\`: left division. The operation `A\B` is effectively the same as `INV(A)*B`, although left division is calculated differently and is much quicker.
- `+`: addition
- `-`: subtraction

Array Operations

- Evaluated element by element
  - `.'` : array transpose (non-conjugated transpose)
  - `.^` : array power
  - `.*` : array multiplication
  - `./` : array division
- Very different from Matrix operations

```
>> A=[1 2;3 4];
>> B=[5 6;7 8];
>> A*B
    19    22
    43    50
```

But:

```
>> A.*B
     5    12
    21    32
```


Some Built-in functions

- `mean(A)`: mean value of a vector
- `max(A), min (A)`: maximum and minimum.
- `sum(A)`: summation.
- `sort(A)`: sorted vector
- `median(A)`: median value
- `std(A)`: standard deviation.
- `det(A)` : determinant of a square matrix
- `dot(a,b)`: dot product of two vectors
- `Cross(a,b)`: cross product of two vectors
- `Inv(A)`: Inverse of a matrix A

Indexing Matrices

Given the matrix:

```
A =  <------------ n ------------>
 m   0.9501    0.6068    0.4231
     0.2311    0.4860    0.2774
```

Then:

`A(1,2) = 0.6068` ⟶ $A_{ij}, i = 1...m, j = 1...n$

`A(3) = 0.6068` ⟶ $index = (i-1)m + j$

`A(:,1) = [0.9501`
(↑ 1:*m*)
`0.2311 ]`

`A(1,2:3)=[0.6068  0.4231]`

Adding Elements to a Vector or a Matrix

```
>> A=1:3
A=
   1  2  3
>> A(4:6)=5:2:9
A=
   1  2  3  5  7  9

>> B=1:2
B=
   1  2
>> B(5)=7;
B=
   1  2  0  0  7
```

```
>> C=[1 2; 3 4]
C=
   1  2
   3  4
>> C(3,:)=[5 6];
C=
   1  2
   3  4
   5  6

>> D=linspace(4,12,3);
>> E=[C D']
E=
   1  2  4
   3  4  8
   5  6  12
```

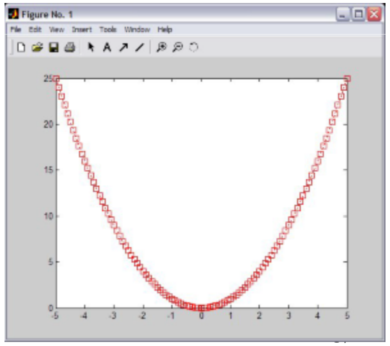
Graphics - 2D Plots

`plot(xdata, ydata, 'marker_style');`

For example:

```
>> x=-5:0.1:5;
>> sqr=x.^2;
>> pl1=plot(x, sqr, 'r:s');
```

Gives:



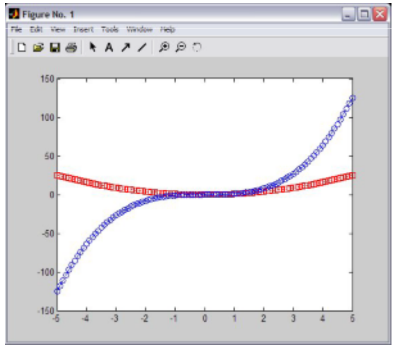
Graphics - Overlay Plots

Use `hold on` for overlaying graphs

So the following:

```
>> hold on;
>> cub=x.^3;
>> pl2=plot(x, cub,'b-o');
```

Gives:

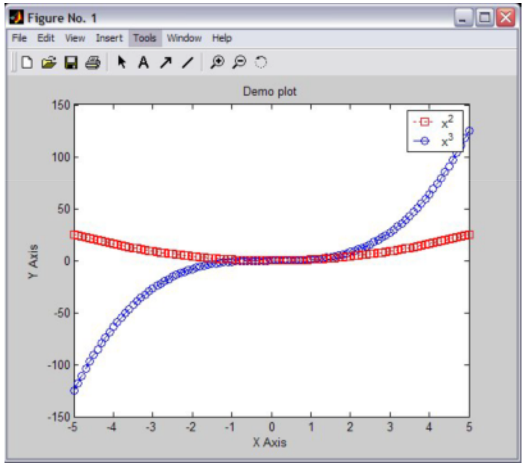


Graphics - Annotation

Use `title`, `xlabel`, `ylabel` and `legend` for annotation

```
>> title('Demo plot');
>> xlabel('X Axis');
>> ylabel('Y Axis');
>> legend([pl1, pl2], 'x^2', 'x^3');
```

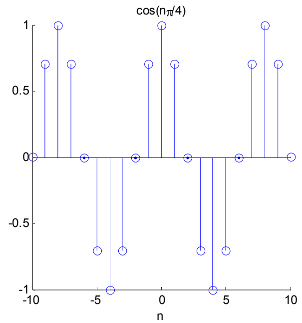
Graphics - Annotation



Graphics-Stem()

- `stem()` is to plot discrete sequence data
- The usage of `stem()` is very similar to `plot()`

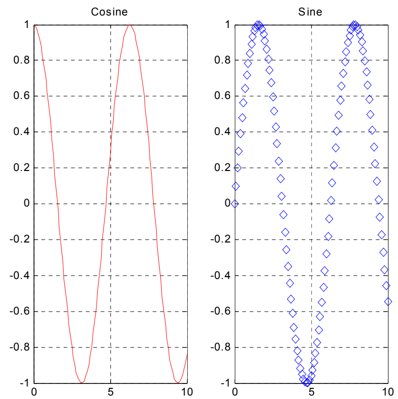
```
>> n=-10:10;
>> f=stem(n,cos(n*pi/4))
>> title('cos(n\pi/4)')
>> xlabel('n')
```



subplots

- Use subplots to divide a plotting window into several panes.

```
>> x=0:0.1:10;
>> f=figure;
>> f1=subplot(1,2,1);
>> plot(x,cos(x),'r');
>> grid on;
>> title('Cosine')
>> f2=subplot(1,2,2);
>> plot(x,sin(x),'d');
>> grid on;
>> title('Sine');
```



Save plots

- Use `saveas(h,'filename.ext')` to save a figure to a file.

```
>> f=figure;
>> x=-5:0.1:5;
>> h=plot(x,cos(2*x+pi/3));
>> title('Figure 1');
>> xlabel('x');
>> saveas(h,'figure1.fig')
>> saveas(h,'figure1.eps')
```

Useful extension types:

bmp: Windows bitmap
emf: Enhanced metafile
eps: EPS Level 1
fig: MATLAB figure
jpg: JPEG image
m: MATLAB M-file
tif: TIFF image, compressed

Workspace

- Matlab remembers old commands
- **And** variables as well
- Each Function maintains its own scope
- The keyword `clear` removes all variables from workspace
- The keyword `who` lists the variables

File I/O

- Matlab has a native file format to save and load workspaces. Use keywords `load` and `save`.
- In addition MATLAB knows a large number of popular formats. Type "`help fileformats`" for a listing.
- In addition MATLAB supports 'C' style low level file I/O. Type "`help fprintf`" for more information.

Practice Problems

- Plot the following signals in linear scale

$$x(t) = \sin(3t) \qquad -5 < t < 5$$
$$y(t) = e^{2t+3} \qquad 0 < t < 5$$

- Plot the following signals, use log scale for y-axis

$$x(t) = e^{t+2}(2t+1) \quad 0 < t < 10$$

- Plot the real part and imaginary part of the following signal

$$x(t) = e^{0.5t + j(t+\pi/3)} \qquad 0 < t < 10$$

- For the signal in previous question, plot its phase and magnitude