Introduction to Matlab

MA 573

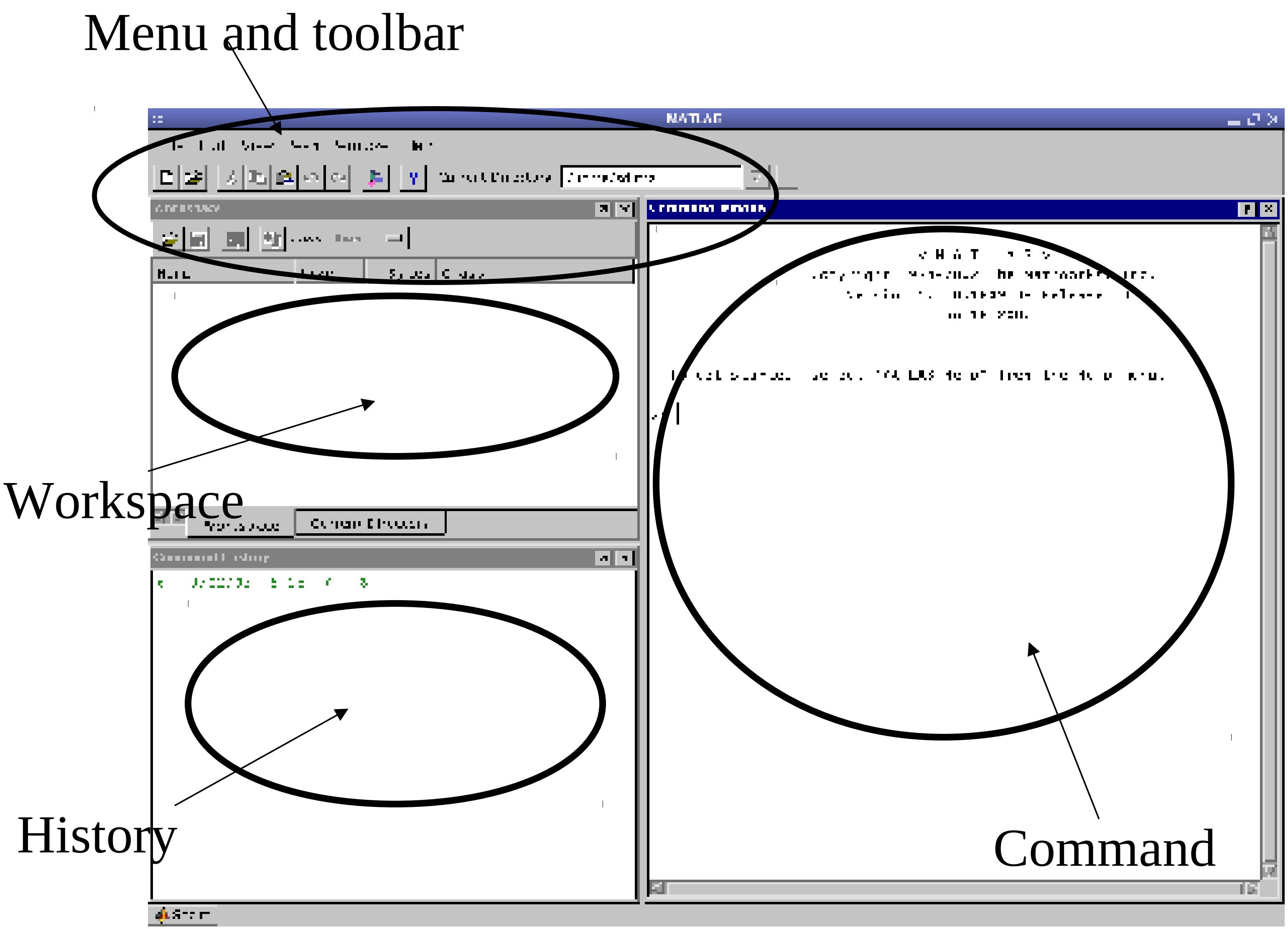
Lab Component

Date : 08.01.2020

History

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing Matlab in the late 1970s. Later In 1984, MathWorks started to continue its development. Matlab stands for Matrix Laboratory.

MATLAB Desktop



Matlab Assignment & Operators

Assignment	=	a = b (assign b to a)
Addition	+	a + b
Subtraction	-	a - b
Multiplication	* or.*	a*b or a.*b
Division	/ or ./	a/b or a./b
Power	^ or .^	a^b or a.^b

```
a= input('enter the number a= ');
b= input('enter the number b= ');
s= a+b;
fprintf('the final sum is= %d\n', s);
```

Matrices & Vectors

» rowvec = [12 , 14 , 63] or [12 14 63]

rowvec = 12 14 63

» colvec = [13 ; 45 ; -2]

colvec =

13

45

-2

Matrices & Vectors

Matrices Easy to define:

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

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

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
```

Matrices

```
>>A = [1  2  3; 4  5  6; 7  8  9]
A =
    1    2    3
    4    5    6
    7    8    9
>> A'
A'=
    1    4    7
    2    5    8
    3    6    9
```

>> A(:,2) - For getting second column

>> A(2,:) - For getting second row

>> det(A) - getting determinant

>> inv(A) – getting inverse

>> A*B – matrix multiplication

>> A+B – matrix addition

>> diag(A) – diagonal element of A

>> zeros(n) – (n x n) zero matrix

>> eye(n) – (n x n) identity matrix

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
```

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

>> sqrt(-1)

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.

  s_{ij}= (i+j)! / i! j!

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

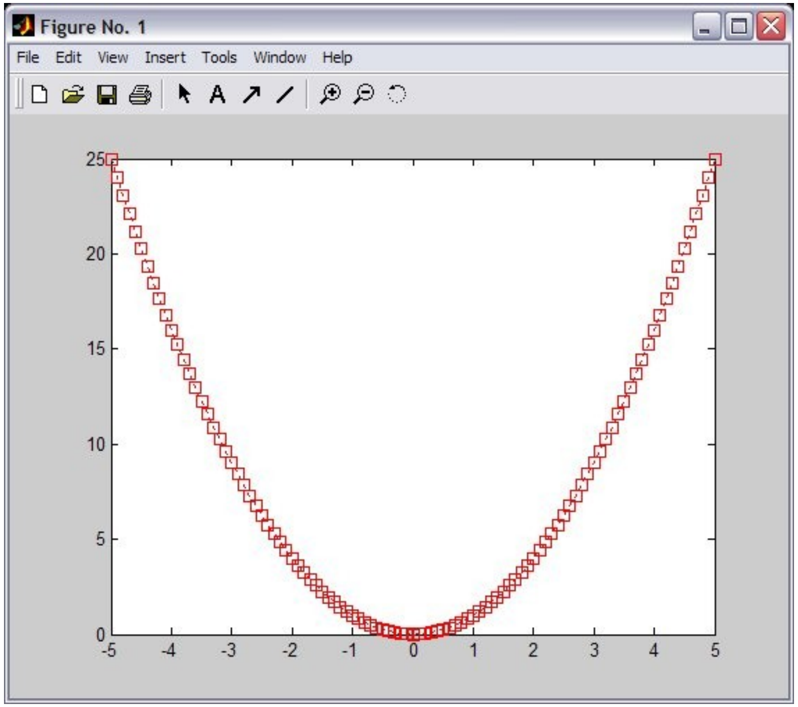
Graphics - 2D Plots

```
plot(xdata, ydata, ‘marker_style’);
```

For example:

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

Gives:

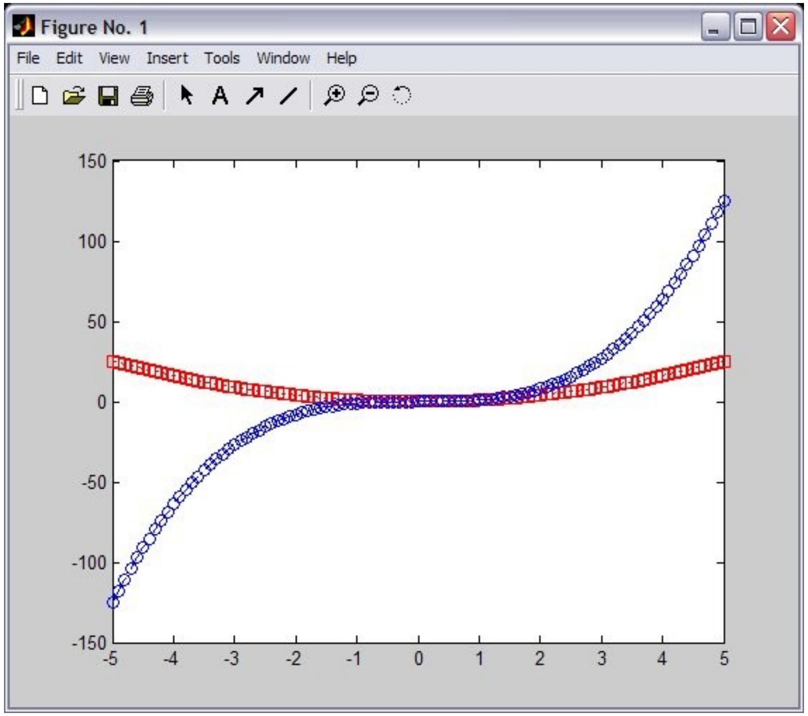


Graphics - Overlay Plots

Use `hold on` for overlaying graphs

So the following: Gives:

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

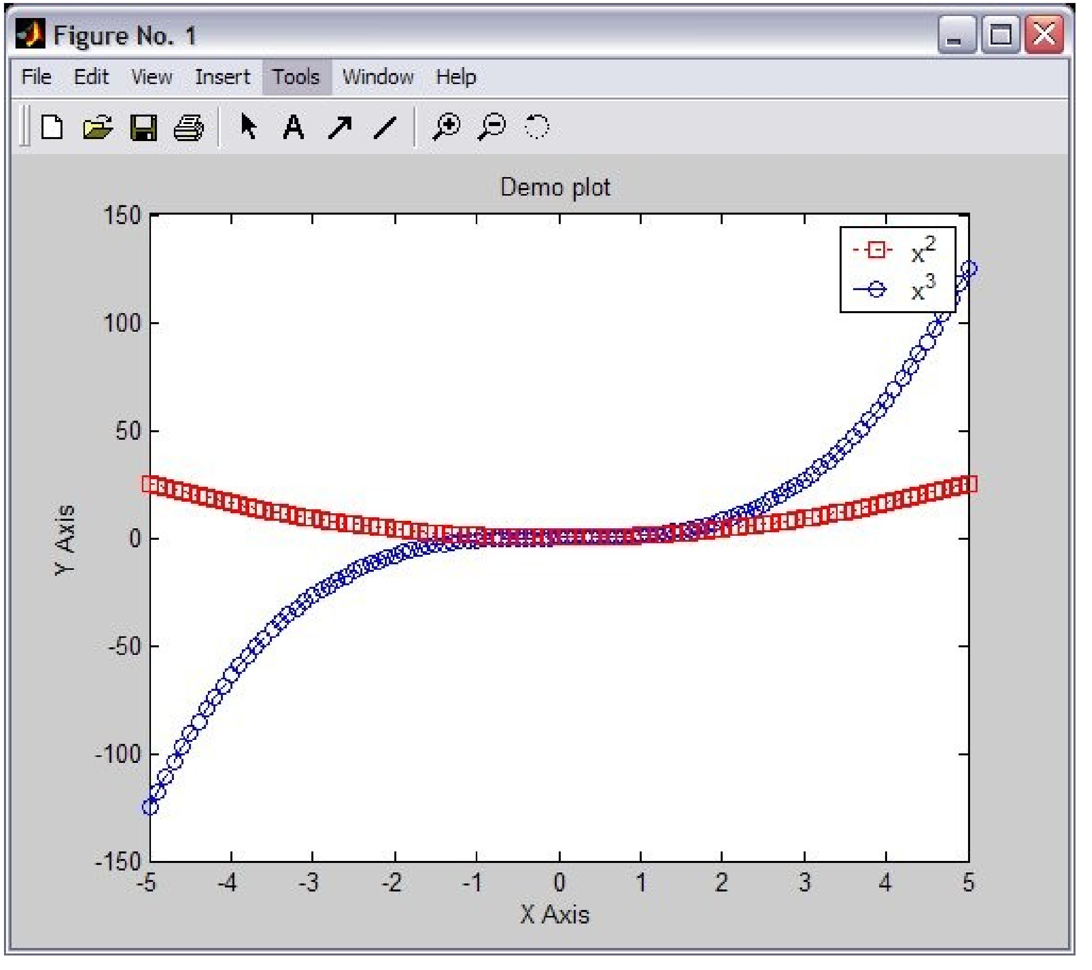


Graphics - Annotation

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

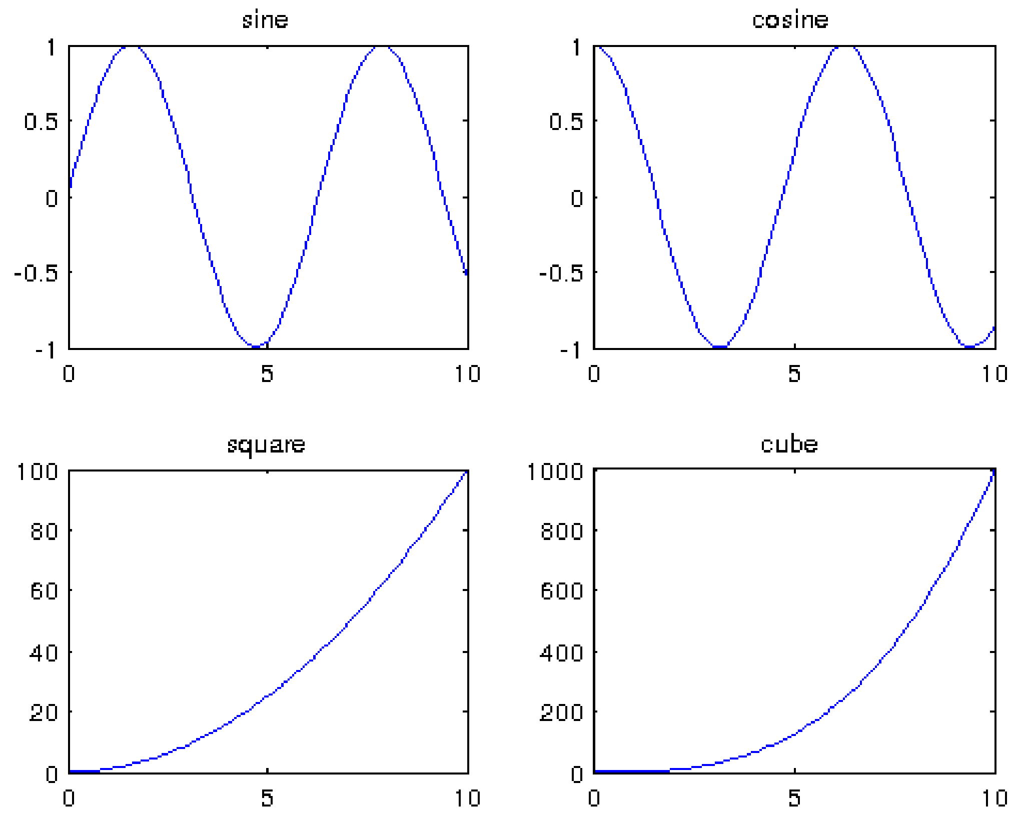
```
>> title('Demo plot');
>> xlabel('X Axis');
>> ylabel('Y Axis');
>> legend([pl1, pl2], 'x^2', 'x^3');
```

Graphics - Annotation



Subplot

```
x=0:0.1:10;
f1=sin(x);
f2=cos(x);
f3=x.^2;
f4=x.^3;
subplot(2,2,1);
plot(x,f1);
title('sine');
subplot(2,2,2);
plot(x,f2);
title('cosine');
subplot(2,2,3);
plot(x,f3);
title('square');
subplot(2,2,4);
plot(x,f4);
title('cube');
```



Functions

```
function f= myfunc(x)
    f=x.^3;
end
```

Save it an another file and give the file name myfunc.m . then in main file declare the function as :

```
x=linspace(2,3,10);
final= myfunc(x);
plot(x,final,'r:o');
```

Practice Problems

- Plot the following signals in linear scale

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

- 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

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

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