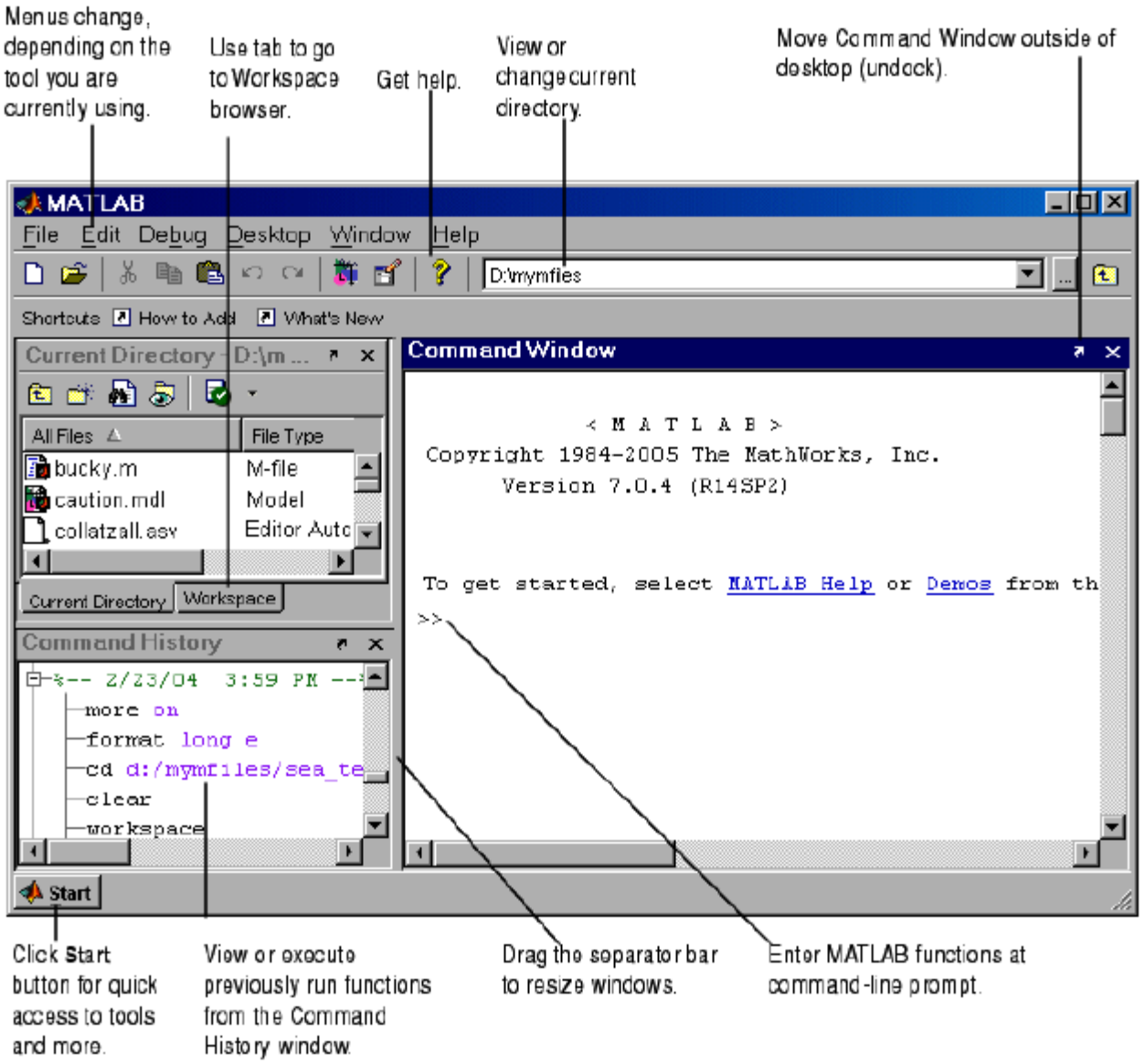
An introduction to MATLAB

- The name MATLAB stands for MATrix LABoratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.
- MATLAB is a high-performance language for technical computing. It integrates *computation, visualization, and programming environment.*
- MATLAB is an interactive system whose basic data element is an *array that does not require dimensioning.*
- It has powerful *built-in routines that enable a very wide variety of computations.*

An introduction to MATLAB



Creating MATLAB variables

- MATLAB variables are created with an assignment statement (without type).
- You write it at the prompt command (>>) as follows:

```
>> t = 5;
>> t = t+1
t =
6
```

- You can suppress the numerical output by putting a semicolon (;) at the end of the line.
- MATLAB does numerical calculations in *double precision.*

Mathematical functions

Table 2.1: Elementary functions

`cos(x)`	Cosine	`abs(x)`	Absolute value
`sin(x)`	Sine	`sign(x)`	Signum function
`tan(x)`	Tangent	`max(x)`	Maximum value
`acos(x)`	Arc cosine	`min(x)`	Minimum value
`asin(x)`	Arc sine	`ceil(x)`	Round towards $+\infty$
`atan(x)`	Arc tangent	`floor(x)`	Round towards $-\infty$
`exp(x)`	Exponential	`round(x)`	Round to nearest integer
`sqrt(x)`	Square root	`rem(x)`	Remainder after division
`log(x)`	Natural logarithm	`angle(x)`	Phase angle
`log10(x)`	Common logarithm	`conj(x)`	Complex conjugate

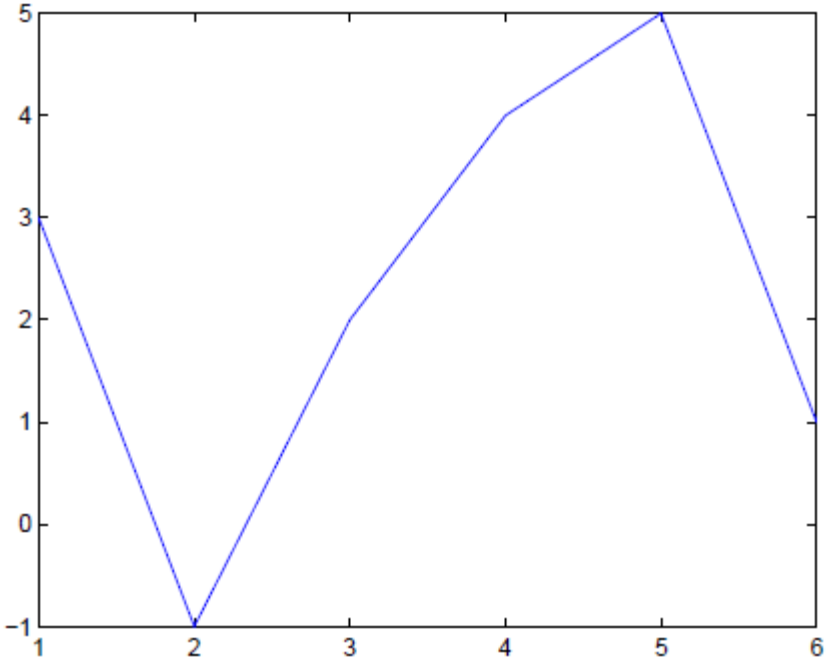
Table 2.2: Predefined constant values

`pi`	The π number, $\pi = 3.14159\ldots$
`i,j`	The imaginary unit i, $\sqrt{-1}$
`Inf`	The infinity, ∞
`NaN`	Not a number

Basic plotting

- The MATLAB command to plot a graph is *plot(x,y)*. The vectors *x = (1, 2, 3, 4, 5, 6)* and *y = (3, -1, 2, 4, 5, 1)* *produce the picture shown in figure.*

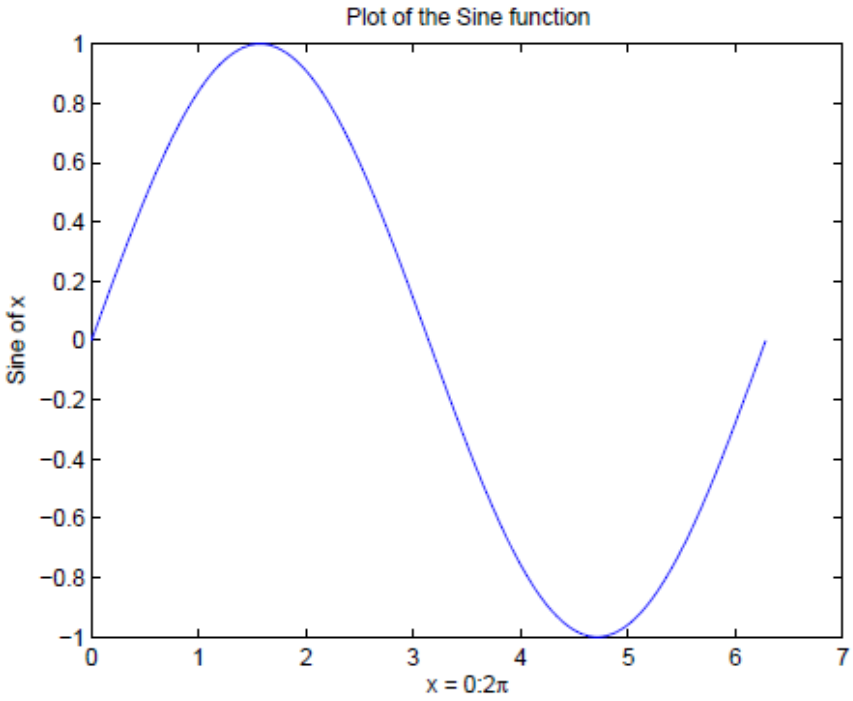
```
>> x = [1 2 3 4 5 6];
>> y = [3 -1 2 4 5 1];
>> plot(x,y)
```



Basic plotting

`0:pi/100:2*pi` yields a vector that

- starts at 0,
- takes steps (or increments) of π=*100,*
- Stops when 2π *is reached.*




```
>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> plot(x,y)
>> xlabel('x = 0:2\pi')
>> ylabel('Sine of x')
>> title('Plot of the Sine function')
```

Entering a vector

- Matrices are the basic elements of the MATLAB environment. A matrix is a two-dimensional array consisting of *m rows and n columns. Special cases are column vectors (n = 1) and row vectors (m = 1).*

```
>> v = [1 4 7 10 13]
v =
1 4 7 10 13
```

```
>> w = [1;4;7;10;13]
w =
1
4
7
10
13
```

- The transpose operation is denoted by an apostrophe or a single quote (').

Entering a vector

- Thus, v(1) is the first element of vector v, v(2) its second element, and so forth.
- Furthermore, to access *blocks of elements, we use MATLAB's colon notation (:).* For example, to access the first three elements of v, we write,

```
>> v(1:3)
ans =
1 4 7
```

Entering a matrix

- type,

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

- MATLAB then displays the 3x3 *matrix as follows,*

```
A =
1 2 3
4 5 6
7 8 9
```

```
>> A(2,1)
ans =
4
```

- A(2,1) is an element located in the second row and first column.

Entering a matrix

- MATLAB provides functions that generates elementary matrices.

Table 2.4: Elementary matrices

`eye(m,n)`	Returns an m-by-n matrix with 1 on the main diagonal
`eye(n)`	Returns an n-by-n square identity matrix
`zeros(m,n)`	Returns an m-by-n matrix of zeros
`ones(m,n)`	Returns an m-by-n matrix of ones
`diag(A)`	Extracts the diagonal of matrix A
`rand(m,n)`	Returns an m-by-n matrix of random numbers

Array operations

- MATLAB allows arithmetic operations: +, -, *, and ^ to be carried out on matrices.

  e.g. A*B is valid if A's number of column equals B's number of rows
- On the other hand, array arithmetic operations or array operations for short, are done *element-by-element.*

```
>> C = A.*B
C =
10  40  90
160 250 360
490 640 810
```

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{bmatrix}$$

M-File Scripts

- A *script file* is an external file that contains a sequence of MATLAB statements. Script files have a filename extension `.m` and are often called M-files.
- Use the MATLAB *editor to create a file: File -> New -> M-file.*
- Enter the statements in the file
- Save the file, for example, example1.m.
- Run the file, in the command line, by typing:

```
>> example1
```

M-File functions

- Functions are programs (or *routines) that accept input arguments and* return *output arguments.*
- This simple function shows the basic parts of an M-file.

```
function f = factorial(n)
% FACTORIAL(N) returns the factorial of N.
% Compute a factorial value.

f = prod(1:n);
```

Control flow

- The simplest form of the if statement is

```
if expression
     statements
end
```

```
discr = b*b - 4*a*c;
if discr < 0
     disp('Warning: discriminant is
     negative, roots are  imaginary');
elseif discr == 0
     disp('Discriminant is zero, roots are
     repeated')
else
     disp('Roots are real')
end
```

Control flow

- The for ... end loop

```
for variable = expression
     statements
end
```

```
for ii=1:5
     x=ii*ii
end
```

```
x =
     1
x =
     4
x =
     9
x =
    16
x =
    25
```

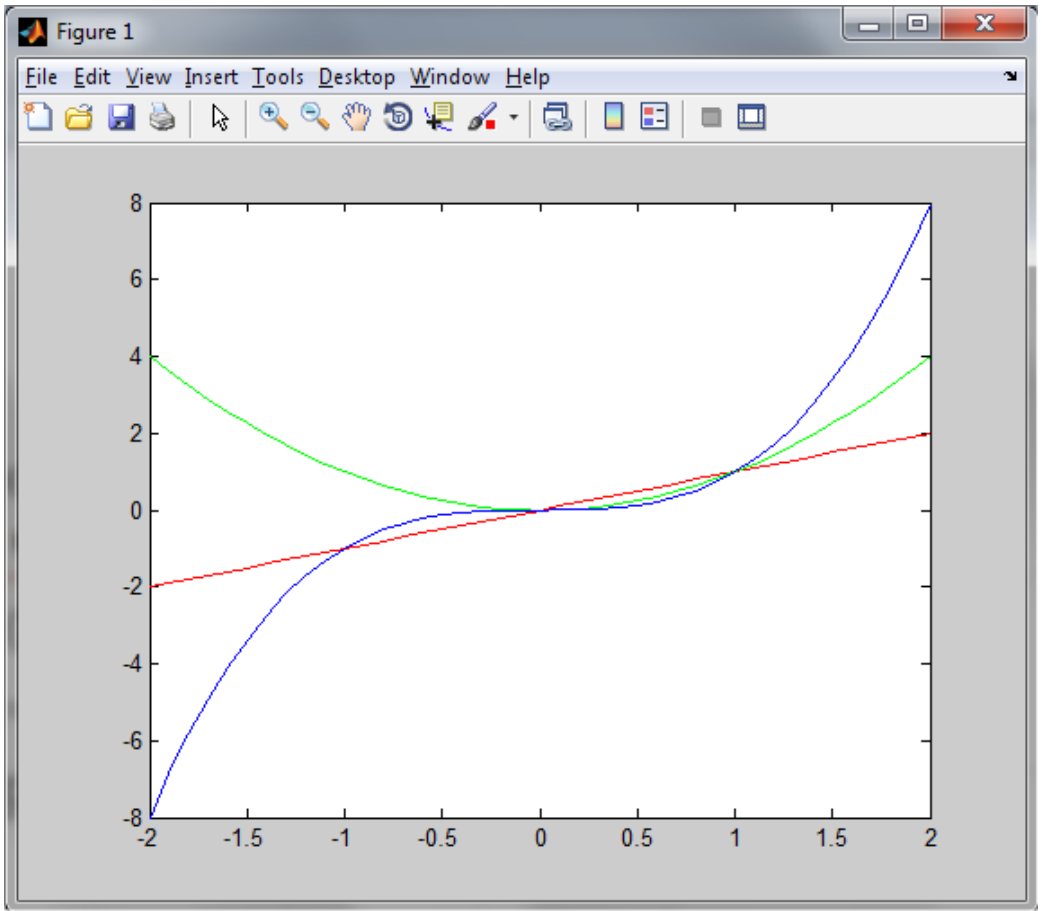
Relational and logical operators

- Table 5.1: Relational and logical operators

Operator	Description
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
~=	Not equal to
&	AND operator
|	OR operator
~	NOT operator

Example

```
figure
c=['r','g','b'];
x=-2:0.1:2;
for a=1:3
     plot(x,x.^a,c(a)), hold on
end
```

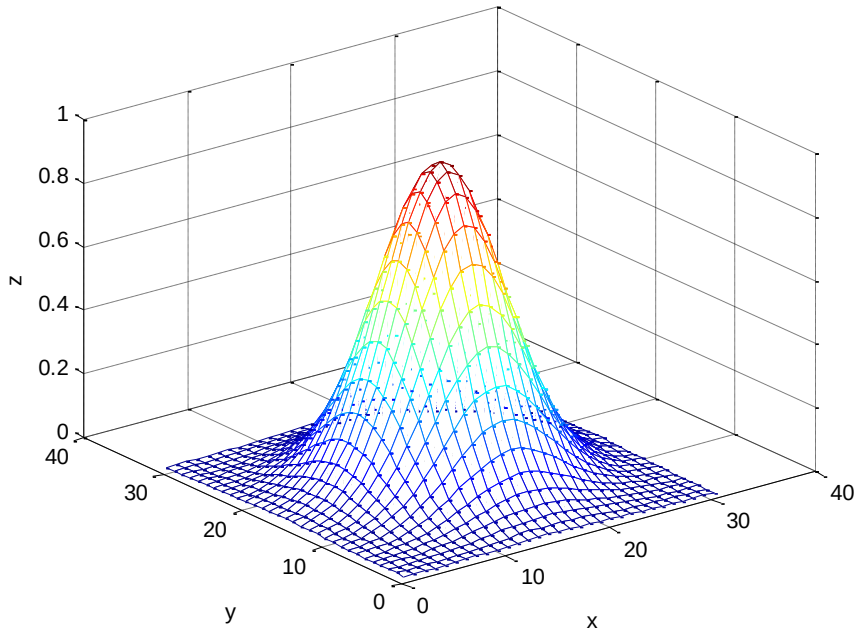


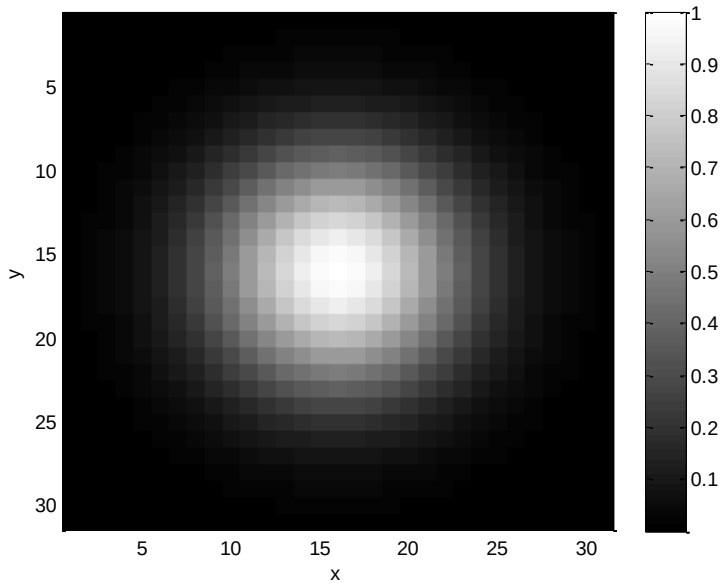
To evaluate functions of two variables

- The `meshgrid` function transforms the domain specified by two vectors, x and y, into matrices X and Y.
- You then use these matrices to evaluate functions of two variables: the rows of X are copies of the vector x and the columns of Y are copies of the vector y.
- To plot Z=f(X,Y) you can use: `mesh` to obtain the 3D surface or `images` to obtain the 2D view from the above.

To evaluate functions of two variables

```
%Gaussian
[X,Y]=meshgrid(-15:15);
Z=exp((-X.^2 -Y.^2)/(2*5^2));%standard deviation: 5 pixels
figure,mesh(Z),xlabel('x'),ylabel('y'),zlabel('z')
figure,imagesc(Z),colormap gray,xlabel('x'),ylabel('y')
```





MATLAB

- http://www.mathworks.com
- http://www.mathworks.com/help/
- http://www.mathworks.com/support/product/examples-index.html
- http://www.mathworks.com/academia/student_center/tutorials/launchpad.html