Engineering and Scientific Applications: Using MatLab® for Data Processing and Visualization

Syamal K. Sen

Department of Mathematical Sciences, Florida Institute of Technology, 150 West University Boulevard, Melbourne, FL 32901-6975, United States

sksen@fit.edu

and

Gholam Ali Shaykhian

National Aeronautics and Space Administration (NASA), Information Technology (IT) Directorate, Technical Integration Office (IT-G
Kennedy Space Center, FL 32899, United States
ali.shaykhian@nasa.gov

Abstract: MatLab® (MATrix LABoratory) is a numerical computation and simulation tool that is used by thousands Scientists and Engineers in many countries. MatLab does purely numerical calculations, which can be used as a glorified calculator or interpreter programming language; its real strength is in matrix manipulations. Computer algebra functionalities are achieved within the MatLab environment using "symbolic" toolbox. This feature is similar to computer algebra programs, provided by Maple or Mathematica to calculate with mathematical equations using symbolic operations.

MatLab in its interpreter programming language form (command interface) is similar with well known programming languages such as C/C++, support data structures and cell arrays to define classes in object oriented programming. As such, MatLab is equipped with most of the essential constructs of a higher programming language. MatLab is packaged with an editor and debugging functionality useful to perform analysis of large MatLab programs and find errors.

We believe there are many ways to approach real-world problems; prescribed methods to ensure foregoing solutions are incorporated in design and analysis of data processing and visualization can benefit engineers and scientist in gaining wider insight in actual implementation of their perspective experiments. This presentation will focus on data processing and visualizations aspects of engineering and scientific applications. Specifically, it will discuss methods and techniques to perform intermediate-level data processing covering engineering and scientific problems. MatLab programming techniques including reading various data files formats to produce customized publication-quality graphics, importing engineering and/or scientific data, organizing data in tabular format, exporting data to be used by other software programs such as Microsoft Excel, data presentation and visualization will be discussed.

The presentation will emphasize creating practical scripts (programs) that extend the basic features of MatLab Topics include

- Matrix and vector analysis and manipulations
- Mathematical functions
- Symbolic calculations & functions
- Import/export data files
- Program logic and flow control
- Writing function and passing parameters
- Test application programs

Engineering and Scientific Applications: Using MatLab® for Data Processing and Visualization

Sixth International Conference on Dynamic Systems and Applications

May, 25-28, 2011
www.dynamicpublishers.com/icdsa6.htm
Morehouse College, Atlanta, GA, 30314, USA.

Syamal K. Sen *Department of Mathematical Sciences* *Florida Institute of Technology* *150 West University Boulevard* *Melbourne, FL 32901-6975* sksen@fit.edu	**Gholam Ali Shaykhian** *Information Technology (IT) Directorate* *Technical Integration Office (IT-G)* *NASA-KSC* *Kennedy Space Center, FL 32899* *ali.shaykhian@nasa.gov*

Agenda:

- Abstract
- Matrix and vector analysis and manipulations
- Mathematical functions
- Symbolic calculations & functions
- Import/export data files
- Writing function and passing parameters
- Test application programs

Abstract

MatLab® (MATrix LABoratory) is a numerical computation and simulation tool that is used by thousands of Scientists and Engineers in many countries. MatLab does purely numerical calculations, which can be used as a glorified calculator or interpreter programming language; its real strength is in matrix manipulations. Computer algebra functionalities are achieved within the MatLab environment using "symbolic" toolbox. This feature is similar to computer algebra programs, provided by Maple or Mathematica to calculate with mathematical equations using symbolic operations.

MatLab in its interpreter programming language form (command interface) is similar to well known programming languages such as C/C++, support data structures and cell arrays to define classes in object oriented programming. As such, MatLab is equipped with most of the essential constructs of a higher programming language. MatLab is packaged with an editor and debugging functionality useful to perform analysis of large MatLab programs and find errors.

We believe there are many ways to approach real-world problems; prescribed methods to ensure foregoing solutions are incorporated in design and analysis of data processing and visualization can benefit engineers and scientist in gaining wider insight in actual implementation of their perspective experiments/design. This presentation will focus on data processing and visualization aspects of engineering and scientific applications. Specifically, it will discuss methods and techniques to perform intermediate-level data processing covering engineering and scientific problems. MatLab programming techniques including reading various data files formats to produce customized publication-quality graphics, importing engineering and/or scientific data, organizing data in tabular format, exporting data to be used by other software programs such as Microsoft Excel, data presentation and visualization will be discussed.

The presentation will emphasize creating practical scripts (programs) that extend the basic features of MatLab.

Topics include:

Matrix and vector analysis and manipulations

Symbolic calculations & functions

Program logic and flow control

Test application programs

Mathematical functions

Import/export data files

Writing function and passing parameters

Vector

Row vector

There are different ways to declare a row vector

A row vector with 5 different elements declared and initialized below:

```
>> a = [1, 2, 3, 4, 5]

>> a = [1 2 3 4 5]

>> a = 1:5

>> a = [1:5]
```

Column vector

Column vector are defined by using a ; between each element of vector

Column vector can also be created by transposing a row vector

Single quote (') is used to transpose a matrix

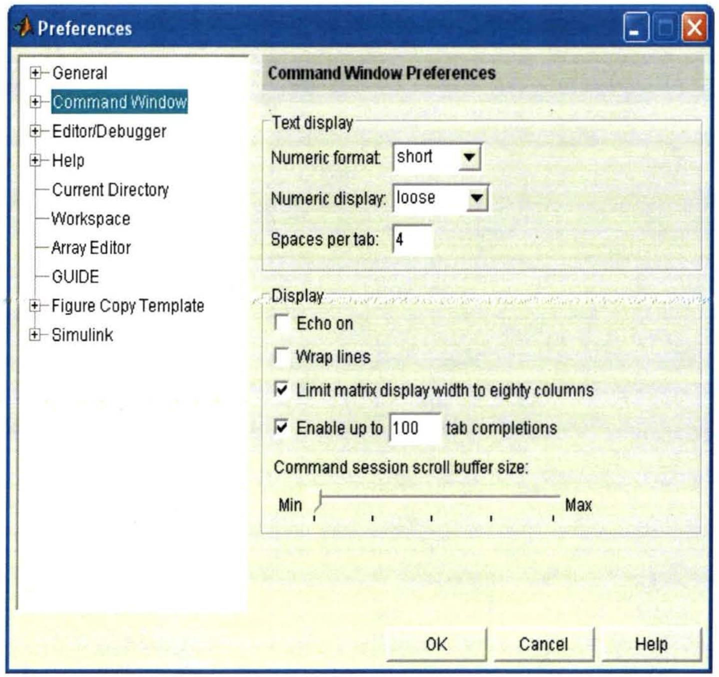
```
>> b = [1; 2; 3; 4; 5]

>> b = a'
```

Matrix

Matrix (two dimensional)

```
>> m = [1.2, 3, 4; -3.7, -2, 5; 1, 2, 3]

m =
    1.2000    3.0000    4.0000
   -3.7000   -2.0000    5.0000
    1.0000    2.0000    3.0000
```

Note: The default representation of numeric values can be changed through the File>Preferences>Numeric Format



Work with Matrix

Add extra row to a Matrix

```
>> a = [1 2 4;2 4 6]
a =
    1   2   4
    2   4   6
>> a = [a; 7 7 7]
a =
    1   2   4
    2   4   6
    7   7   7
```

Add extra column to a Matrix

```
>> a = [1 2 4; 2 4 6]
a =
    1   2   4
    2   4   6
>> a = [a, [9; 9] ]
a =
    1   2   4   9
    2   4   6   9
```

Work with Matrix

Colon Operator:

- Used to define new matrices
- Modify existing matrices
- Extract data from existing matrices

Note: M(:) Converts a two dimensional matrix to a single column

```
% delete second column
>> a(:,2) = [ ]
a =
   2   6
   3   7
   1   9
```

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

```
% delete third row
>> a(3,:) = [ ]
a =
   2   4   6
   3   5   7
```

Work with Matrix

Select a row/ column from a Matrix and assign it to a vector

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

% select the first row of Matrix a and assign it to vector b

```
>> b = a(1,:)
b =
    2    4    6
```

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

% select the second column of Matrix a and assign it to vector b

```
>> b = a(:,2)

b =
    4
    5
    8
```

Work with Matrix

Select a row/ column from a Matrix and assign it to a vector

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

% select the first row of Matrix a and assign it to vector b

```
>> b = a(1,:)
b =
   2   4   6
```

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

% select the second column of Matrix a and assign it to vector b

```
>> b = a(:,2)

b =
   4
   5
   8
```

Work with Matrix

Convert a matrix to a column vector use the MatLab colon operator (:)

```
> b=[2 3;1 6;7 8]

b =

    2    3
    1    6
    7    8
```

```
>> v=b(:)

v =
    2
    1
    7
    3
    6
    8
```

Elementary Matrix Manipulations

The command help elmat provides a complete list of elementary matrices and matrix manipulation commands

>> help elmat

- zeros - Zeros array, Creates a matrix of all zeros
- ones - Ones array, Creates a matrix of all ones
- eye - Identity matrix.
- repmat - Replicate and tile array.
- rand - Uniformly distributed random numbers.
- randn - Normally distributed random numbers.
- linspace - Linearly spaced vector.
- logspace - Logarithmically spaced vector.
- freqspace - Frequency spacing for frequency response.
- meshgrid - X and Y arrays for 3-D plots.
- : - Regularly spaced vector and index into matrix.

Elementary Matrix Manipulations

- reshape - Change size.
- diag - Diagonal matrices and diagonals of matrix, Extracts a diagonal or creates an identity matrix
- blkdiag - Block diagonal concatenation.
- tril - Extract lower triangular part.
- triu - Extract upper triangular part.
- fliplr - Flip matrix in left/right direction.
- flipud - Flip matrix in up/down direction.
- flipdim - Flip matrix along specified dimension.
- rot90 - Rotate matrix 90 degrees.
- find - Find indices of nonzero elements.
- end - Last index.
- sub2ind - Linear index from multiple subscripts.
- ind2sub - Multiple subscripts from linear index.

Specialized matrices

- compan - Companion matrix.
- gallery - Higham test matrices.
- hadamard - Hadamard matrix.
- hankel - Hankel matrix.
- hilb - Hilbert matrix.
- invhilb - Inverse Hilbert matrix.
- magic - Magic square, Creates a “magic” matrix
- pascal - Pascal matrix.
- rosser - Classic symmetric eigenvalue test problem.
- toeplitz - Toeplitz matrix.
- vander - Vandermonde matrix.
- wilkinson - Wilkinson's eigenvalue test matrix.

Work with Matrix

MatLab ones() function

ones(N) function returns a N-by-N matrix of ones

ones(M,N) returns a M-by-N matrix of ones

Example: Initialize 2 x 3 matrix with ones

```
>> b = ones(2,3)

b =
    1    1    1
    1    1    1
```

MatLab size() function

size(X) returns the size of a matrix (number of rows and columns in the matrix)

```
b =
    1    1    1
    1    1    1

>> size(b)
ans =    2   3
```

MatLab length() function

LENGTH(X) returns the length of (row or column) vector X

```
>> length(b)
ans =    3
```

Matrix Arithmetic Operations

MatLab matrix arithmetic operations are:

A+B

A-B

A*B　　A.*B

A\B　　A.\B

A/B　　A./B

A^B　　A.^B

A'　　A.'

Note: MatLab checks for the computational rules of matrix algebra, an error message is displayed when a rule is violated

+ **matrix addition** is the operation of adding two matrices by adding the corresponding entries together.

A + B adds A and B
A and B must have the same dimensions, unless one is scalar.

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

>> A+B

ans =
     4     7
     9    11
```

Matrix Arithmetic Operations

- **matrix subtraction** is the operation of subtracting two matrices by subtracting the corresponding entries together.

A - B subtracts B from A

A and B must have the same dimensions, unless one is scalar.

```
>> A= [1 2; 3 4]; B=[3 5; 6 7];
A-B
ans =
   -2   -3
   -3   -3
```

* **matrix multiplication** is the operation of multiplying two matrices

```
>> K= M * L
```

the number of columns in the matrix on the left (M) must equal the number of rows in the matrix on the right (L)

Note: $M * L \neq L * M$

```
>> M= [1 2 3; 4 5 6 ]
>> N = [2 3; 5 6; 2 7]
>> K = M*N

K =
   18   36
   45   84
```

Matrix Arithmetic Operations

.* term-by-term multiplication (array multiplication)

A.*B is the entry-by-entry product of A and B

A and B must have the same dimensions

```
>> M= [1 2; 3 4], N=[3 5; 6 7]
>> M.*N
ans =
    3   10
   18   28
```

Matrix Arithmetic Operations

matrix division

MatLab supports two division operators, namely right division / and left division \

\ Matrix left division

X = A\B solves the symbolic linear equations A*X=B

Note that A\B is roughly equivalent to inv(A)*B.

.\ Array left division

A.\B is the matrix with entries B(i,j)/A(i,j)

A and B must have the same dimensions, unless one is scalar.

/ Matrix right division

X=B/A solves the symbolic linear equation X*A=B

Note that B/A is the same as (A.'\B.')

./ Array right division

A./B is the matrix with entries A(i,j)/B(i,j)

A and B must have the same dimensions, unless one is scalar.

Matrix Arithmetic Operations

X2 + X3 = 5
3X1 + X3 = 6
-X1 + X2 = 1

A =	0	1	1
	3	0	1
	-1	1	0

b = [5 ; 6; 1]

```
>> A = [0 1 1; 3 0 1; -1 1 0]
>> b = [5; 6; 1]

>> x = A\b  % left division

x =
    1.0000
    2.0000
    3.0000
```

MatLab Mathematical Functions

MatLab has several standard (preprogrammed) mathematical functions

These preprogrammed functions are grouped as:

- Trigonometric Functions
- Exponential Functions
- Complex Functions
- Rounding and remainder functions

MatLab Mathematical Functions (Trigonometric)

acos	Inverse cosine.
acosh	Inverse hyperbolic cosine.
acot	Inverse cotangent.
acoth	Inverse hyperbolic cotangent.
acsc	Inverse cosecant.
acsch	Inverse hyperbolic cosecant.
asec	Inverse secant.
asech	Inverse hyperbolic secant.
asin	Inverse sine.
asinh	Inverse hyperbolic sine.
atan	Inverse tangent.
atan2	Four quadrant inverse tangent.
atanh	Inverse hyperbolic tangent.
cos	Cosine.
cosh	Hyperbolic cosine.
cot	Cotangent.
coth	Hyperbolic cotangent.
csc	Cosecant.
csch	Hyperbolic cosecant.
sec	Secant.
sech	Hyperbolic secant.
sin	Sine.
sinh	Hyperbolic sine.
tan	Tangent.
tanh	Hyperbolic tangent.

MatLab Mathematical Functions (Exponential/ Complex Functions)

Exp	Exponential (e^x).
Log	Natural logarithm.
Log10	Common (base 10) logarithm.
Log2	Base 2 logarithm and dissect floating-point numbers.
nextpow2	Next higher power of 2.
Pow2	Base 2 power and scale floating-point numbers.
Reallog	Guarantee output from log is a noncomplex matrix.
reallog10	Guarantee output from log10 is a noncomplex matrix.
realpow	Guarantee output from power is a noncomplex matrix.
Realsqrt	Guarantee output from sqrt is a noncomplex matrix.
Sqrt	Square root.

Abs	Absolute value.
Angle	Phase angle.
Conj	Complex conjugate.
cplxpair	Sort numbers into complex conjugate pairs.
Imag	Complex imaginary part.
Isreal	True for noncomplex arrays.
Real	Real part of complex array.
Unwrap	Remove phase angle jumps across 360° boundaries.

MatLab Mathematical Functions (Statistical /Discrete Mathematics Functions)

mean	arithmetic mean or average value of elements
median	median value of elements
min	smallest component
max	largest component
var	variance of the elements in a vector
std	standard deviation from the mean of elements
sum	sum of elements
prod	product of elements
sort	sorting elements within a vector
sortrows	sorting rows within a matrix by values in a column
cov	variance of a vector or covariance of a matrix
corrcoef	correlation coefficient

Discrete Mathematics

factor(x)	returns a vector containing the prime factors of x
gcd(x,y)	greatest common denominator
lcm(x)	lowest common multiple
rats(x)	represent x as a fraction
factorial(x)	returns factorial of x
primes(x)	generates a list of prime numbers less than or equal to x
isprime(x)	returns 1 if the elements of x which are prime, 0 otherwise

MatLab plot() function

plot() function is used to plot a two dimensional plot

plot3() function is used to plot a three dimensional plot

A plot can be made using various symbols, colors and line types

Line types, plot symbols and colors for plot() or plot3() functions are represented through a character string

For example, a character string 'go:' means a green dotted line with a circle at each data point

plot(X,Y,'go:') plots a green dotted line with a circle at each data point

plot3(X,Y,Z,'ks-.') plots a black dashdot line with square at each point

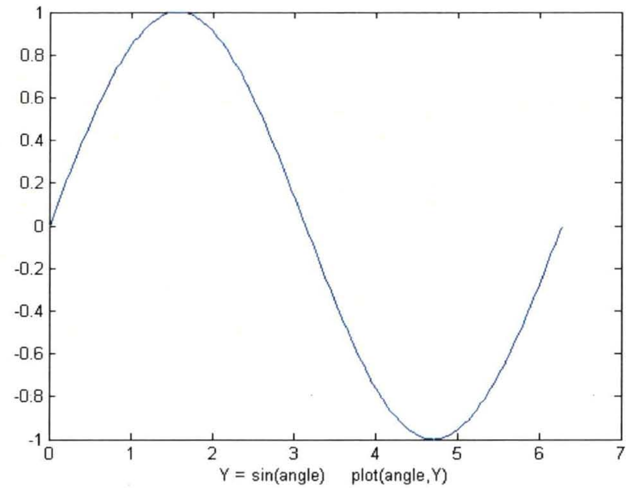
Color	
b	blue
g	green
r	red
c	cyan
m	magenta
y	yellow
k	black

Symbols	
.	point
o	circle
x	x-mark
+	plus
*	star
s	square
d	diamond
v	triangle (down)
^	triangle (up)
<	triangle (left)
>	triangle (right)
p	pentagram
h	hexagram

Line types	
-	solid
:	dotted
-.	dashdot
--	dashed

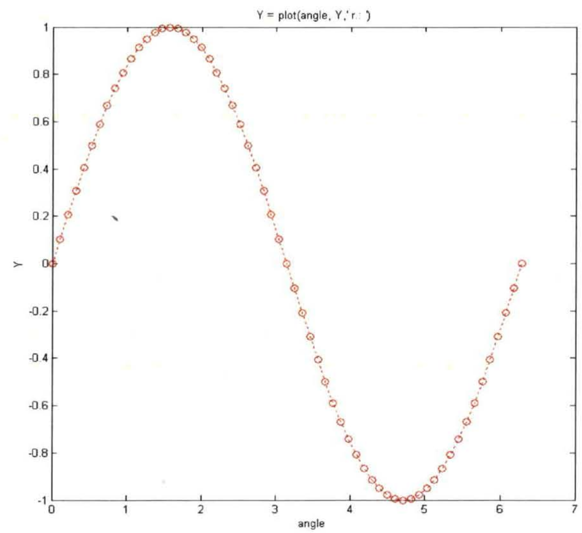
MatLab plot() function

```
>> angle = 0:pi/30:2*pi;  Y = sin(angle)
>>  plot(angle, Y)
```



plot(angle, Y,' r.: ') plots a red dotted line with a circle at each data point

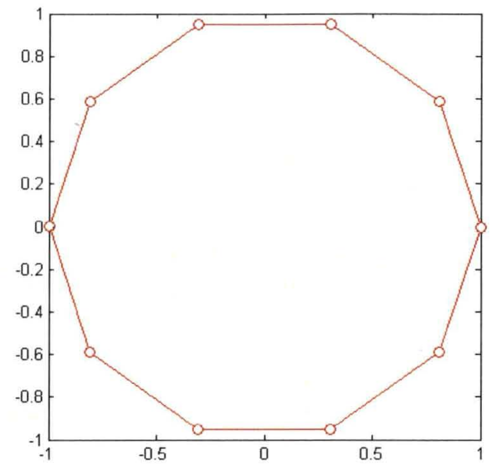
```
>> angle = 0:pi/30:2*pi; Y = sin(angle)
>> plot(angle, Y, 'ro:')
```



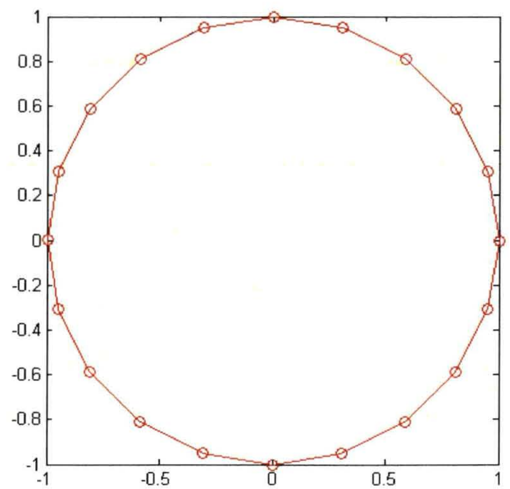
plot(angle, Y,' r.: ') plots a red dotted line with a circle at each data point

MatLab plot() function

```
>> angle=0:0.1:1
>> plot(cos(2*pi*angle),
sin(2*pi*angle),'ro-'); axis square
```

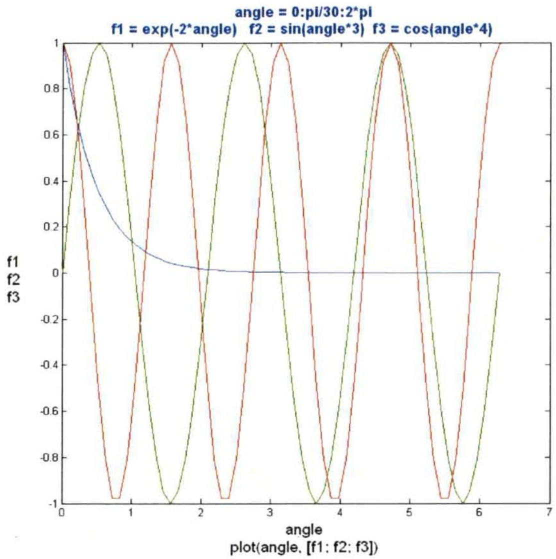



```
>> angle=0:0.05:1
>> plot(cos(2*pi*angle),
sin(2*pi*angle),'ro-'); axis square
```

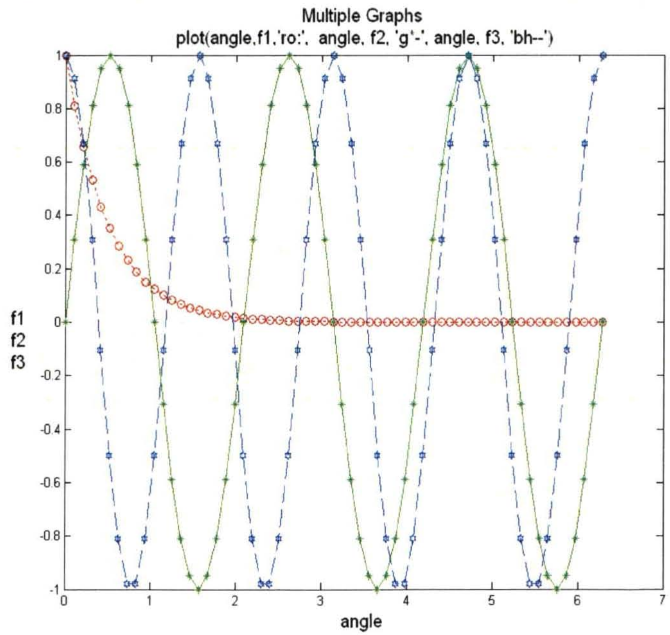


MatLab Multiple function plots

```
>> angle = 0:pi/30:2*pi
>> f1 = exp(-2*angle)
>> f2 = sin(angle*3)
>> f3 = cos(angle*4)
>> plot(angle,[f1; f2; f3])
>> % or plot(angle, f1, angle, f2, angle ,f3)
```

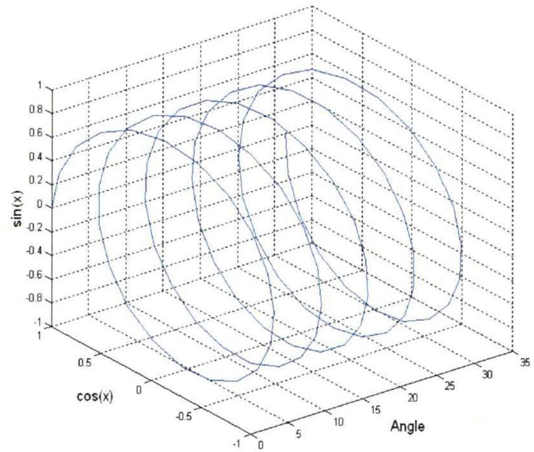


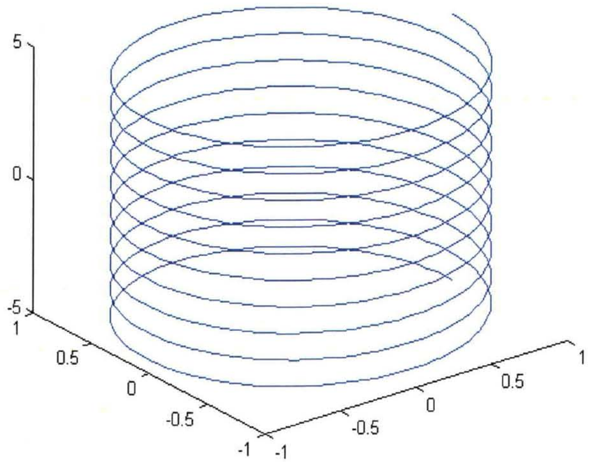

```
>> angle = 0:pi/30:2*pi
>> f1 = exp(-2*angle)
>> f2 = sin(angle*3)
>> f3 = cos(angle*4)
>> plot(angle,f1,'ro:', angle, f2, 'g*-',angle, f3, 'bh--')
```



MatLab Multiple function plots

```
>> angle = linspace(0,10*pi,100)
>> % generates 100 points between 0 and 10* pi
>> y = cos(angle); z = sin(angle)
>> plot3(angle, y, z); grid ...
>> xlabel('Angle'); ylabel('cos(x)'); zlabel('sin(x)')
```

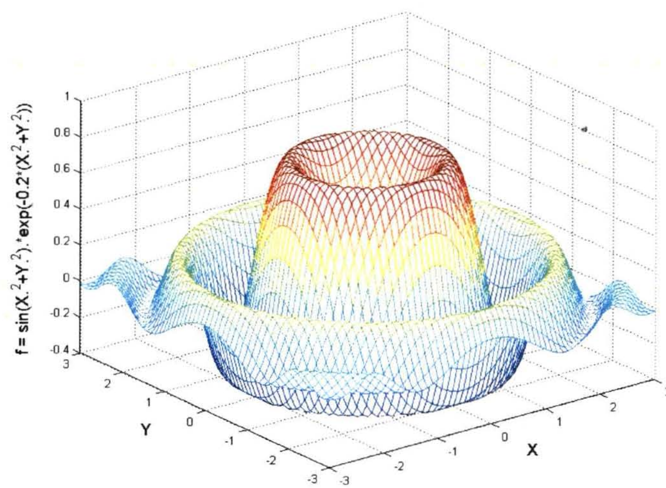




```
>> angle=-5:0.01:5; plot3(cos(2*pi*angle), sin(2*pi*angle),angle)
```

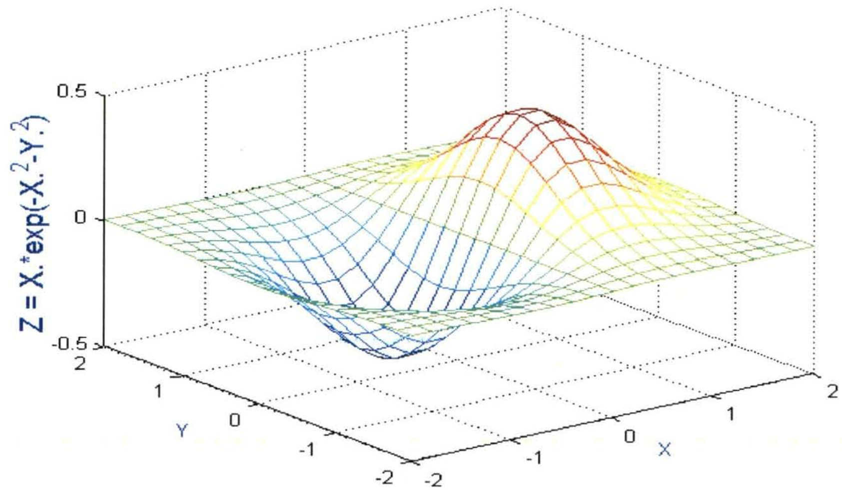
MatLab three-dimensional plots

```
>> %  A three-dimensional plot using the mesh command
>> x = (-3:0.1:3);              % grid frame in x direction
>> y = (-3:0.1:3)';             % grid frame in y direction
>> v = ones(length(x),1);       % auxiliary vector
>> X = v*x;                     % grid matrix of the x values
>> Y = y*v';                    % grid matrix of the y values
>> f = sin(X.^2+Y.^2).*exp(-0.2*(X.^2+Y.^2));
>> % function value
>> mesh(X, Y, f)            % mesh plot with mesh
>> zlabel('f = sin(X.^2+Y.^2).*exp(-0.2*(X.^2+Y.^2))')
>> xlabel('X')
>> ylabel('Y')
```



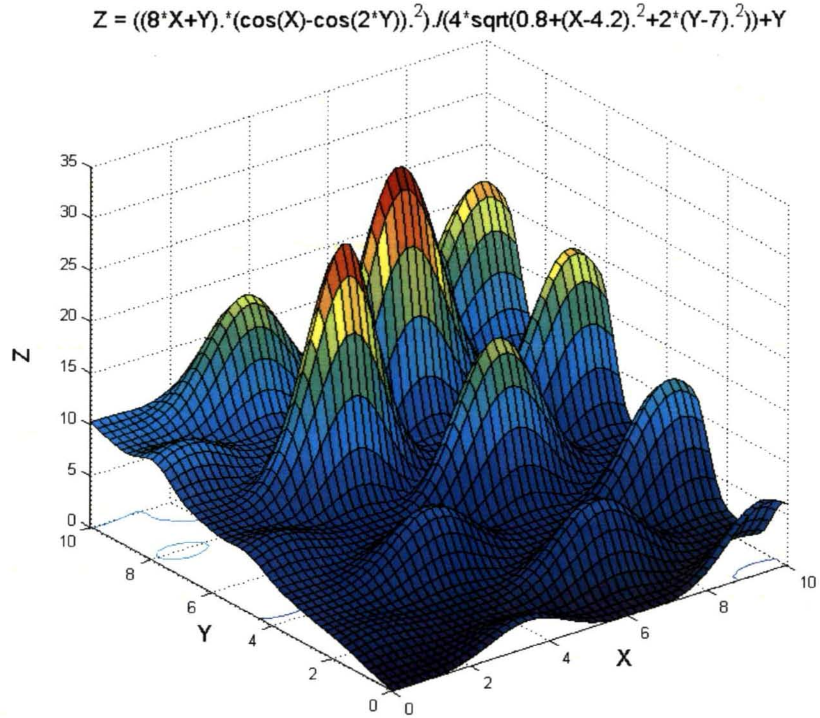
MatLab three-dimensional plots

```
>> % A three-dimensional plot using the mesh command
>> x = [-2:0.2:2]; y = [-2:0.2:2];
>> [X,Y] = meshgrid(x,y);
>> Z=X.*exp(-X.^2-Y.^2);
>> mesh(X,Y,Z);
>> xlabel('X')
>> ylabel('Y')
>> zlabel('Z = X.*exp(-X.^2-Y.^2)')
```



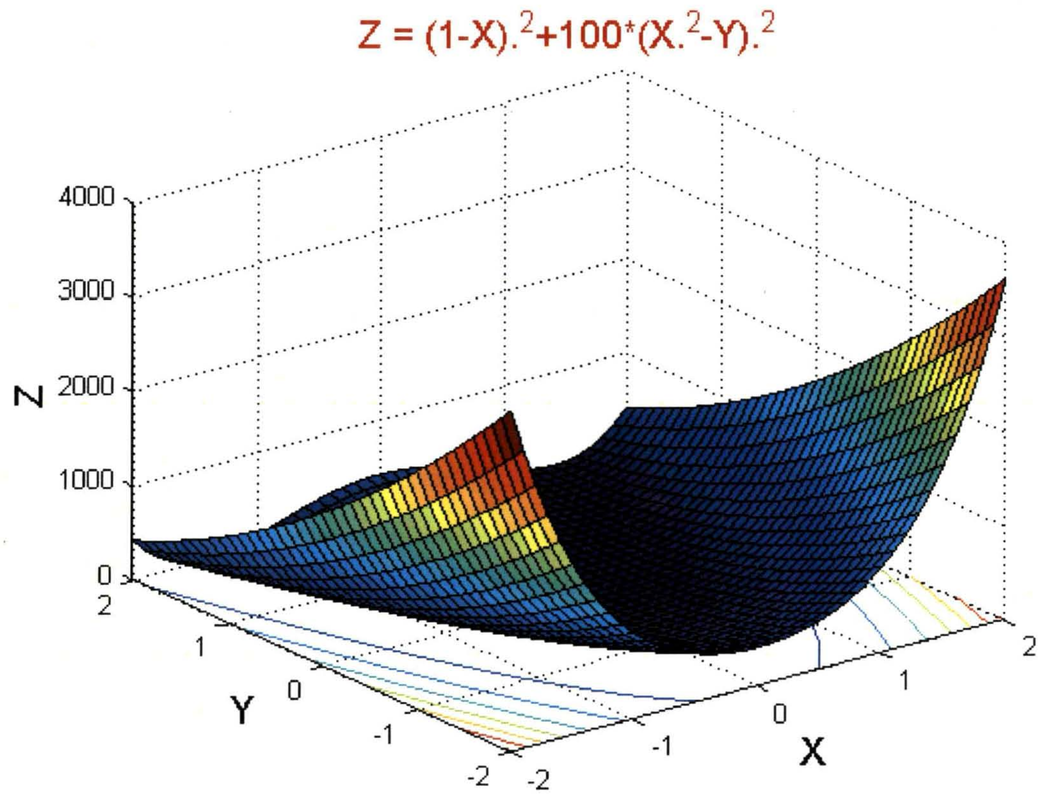
MatLab three-dimensional plots

```
>> % A three-dimensional plot using the surf command
>> x = [-2:0.2:2]; y = [-2:0.2:2];
>> [X,Y] = meshgrid(x,y);
>> Z = ((8*X+Y).*(cos(X)-cos(2*Y)).^2)./(4*sqrt(0.8+(X-4.2).^2+2*(Y-7).^2))+Y;
>> surf(X,Y,Z);
>> xlabel('X')
>> ylabel('Y')
>> zlabel('Z ')
```



MatLab three-dimensional plots

```
>> % A three-dimensional plot using the surf command
>> x = [-2:0.2:2]; y = [-2:0.2:2];
>> [X,Y] = meshgrid(x,y);
>> Z = Z = (1-X).^2+100*(X.^2-Y).^2
>> surf(X,Y,Z);
>> xlabel('X')
>> ylabel('Y')
>> zlabel('Z ')
```



Symbolic Calculations

MATLAB is a numerical simulation tool (not a symbolic algebraic)

MatLab provides a Symbolic Math toolbox to perform symbolic calculations

Type the command help symbolic to get the MatLab symbolic capabilities

The Symbolic Math Toolbox uses "symbolic objects" produced by the "sym" function. For example, the statement

x = sym('x') produces a symbolic variable named x

The statements x = sym('x'); y = sym('y'); can be combined into one statement involving the "syms" function.

syms x y

Symbolic variables can be use in expressions or as arguments to many different functions.

Symbolic Calculations - Simplify a function : simple(f)

Simplify f = cos(x)^2 + sin(x)^2

```
>> f = cos(x)^2 + sin(x)^2
>> f = simple(f)
>> simple(f)

simplify:
1

radsimp:

sin(x)^2+cos(x)^2

combine(trig):

1
```

```
factor:  sin(x)^2+cos(x)^2
expand:  sin(x)^2+cos(x)^2

combine: 1
convert(exp): -1/4*(exp(i*x)-
1/exp(i*x))^2+(1/2*exp(i*x)+1/2/exp(i*x)
)^2

convert(sincos): sin(x)^2+cos(x)^2
convert(tan):
4*tan(1/2*x)^2/(1+tan(1/2*x)^2)^2+(1-
tan(1/2*x)^2)^2/(1+tan(1/2*x)^2)^2
collect(x):
sin(x)^2+cos(x)^2
ans =
1
>>
```

Symbolic Calculations – diff - Differentiate

Steps to execute a symbolic calculation

(1) Use command symbols to declare the variables necessary to perform a symbolic calculation

(2) Use a MatLab symbolic command

```
>> syms x y
% or we can use  x = sym('x'); y= sym('y');

>> f 1 = sin(x*y)*cos(2*y)
f 1 =sin(x*y)*cos(2*y)

>> diff(f1)
>> % differentiate with respect to symbol x
ans =
cos(x*y)*y*cos(2*y)

>> pretty (ans)
                        cos(x y) y cos(2 y)
```

Symbolic Calculations – diff - Differentiate

$f(x, y) = \sin(xy)\cos(2y)$

$df(x, y)/dx$

```
>> syms x y  % or we can use  x = sym('x'); y= sym('y');
>> f 1 = sin(x*y)*cos(2*y)
f 1 =sin(x*y)*cos(2*y)
>> diff(f1)    % differentiate with respect to symbol x
ans =
cos(x*y)*y*cos(2*y)
>> pretty (ans)
                    cos(x y) y cos(2 y)
```

$f(x, y) = \sin(xy)\cos(2y)$

$df(x, y)/dy$

```
>> syms x y    % or we can use  x = sym('x'); y= sym('y');
>> f2 = sin(x*y)*cos(2*y)
f2  =  sin(x*y)*cos(2*y)
>> diff(f2,y)    % differentiate with respect to symbol y
ans =  cos(x*y)*x*cos(2*y)-2*sin(x*y)*sin(2*y)
>> pretty(ans)
cos(x y) x cos(2 y) - 2 sin(x y) sin(2 y)
```

Symbolic Calculations – int - Integrate

Perform the following integrals symbolically, and for the indefinite integrals

$$\int_0^{\pi/2} \cos x \sin x dx$$

*>> int(cos(x)*sin(x),x, 0,pi/2)*

ans = 1/2

$$\int \cos x \sin x dx$$

*>> int(cos(x)*sin(x),x)*

*ans = 1/2*sin(x)^2*

$$\int_{-\infty}^{\infty} 3e(-x\text{^}2)dx$$

*>> f=3*exp(-x^2);*

>> int(f,-Inf,Inf)

*ans = 3*pi^(1/2)*

Symbolic Calculations - Summation

Compute the following sums:

$$\sum_{k=1}^{n} k\wedge 3$$

```
>> syms k n
>> symsum(k^3,k,1, n)
ans =  1/4*(n+1)^4-1/2*(n+1)^3+1/4*(n+1)^2

>> pretty(ans)
                  4              3               2
     1/4 (n + 1)  - 1/2 (n + 1)  + 1/4 (n + 1)

>> simplify(ans)
ans =  1/4*n^4+1/2*n^3+1/4*n^2
```

Symbolic Calculations – MatLab Symbolic functions

```
>> help symbolic
Calculus
    diff       - Differentiate.
    int        - Integrate.
    limit      - Limit.
    taylor     - Taylor series.
    jacobian   - Jacobian matrix.
    symsum     - Summation of series.
```

```
Linear Algebra.
    diag       - Create or extract diagonals.
    triu       - Upper triangle.
    tril       - Lower triangle.
    inv        - Matrix inverse.
    det        - Determinant.
    rank       - Rank.
    rref       - Reduced row echelon form.
    null       - Basis for null space.
    colspace   - Basis for column space.
    eig        - Eigenvalues and eigenvectors.
    svd        - Singular values and singular vectors.
    jordan     - Jordan canonical (normal) form.
    poly       - Characteristic polynomial.
    expm       - Matrix exponential.
```

MatLab Programming

MatLab is a programming language in itself; it includes programming language constructs such as loops, branches and writing of functions or scripts

- *M-File*
- *Scripts*
- *MatLab Functions*
- *Function Handle*

M-files

M-File can be either scripts or functions

Scripts are simply files containing a sequence of MatLab statements/commands

Functions make use of their own local variables and accept input arguments

MATLAB Script

Steps in writing and running a MatLab script program:

Use an editor; to create a file (from the menu) File > New > m-file

Type in the MatLab command sequences

Save the file (e.g. dynamicSys1.m) ; make sure the file path is accessible to MatLab

Run the file (e.g. dynamicSys1.m) ; in Command Window, type the file name (e.g. dynamicSys1) M-File name

View the file (e.g. dynamicSys1.m); type LS in Command Window

Note : The name of an M-file begins with an alphabetic character and has a filename extension of .m.

Import/export data files

```
% Simple program script – Example 1
clear all % clear all variables from memory

% S=load(filename)  load data from the specified file into a matrix
% laod filename  - it uses the file name as the matrix variable name
%
% load student.dat reads all data from the student.dat file into S matrix variable
S=load('student.dat');
height = S(:,1);
weight = S(:,2);
% Calculate Mean and standard deviations
average_weight = mean(weight);
std_weight = std(weight);
maximum_weight = max(weight);
minimum_weight = min(weight);
```

```
disp(sprintf('\n\tAverage Weight: %6.2f', average_weight));
disp(sprintf('\n\tStandard Deviation: %6.2f', std_weight));
disp(sprintf('\n\tMaximum Weight: %6.2f', maximum_weight));
disp(sprintf('\n\tMinimum Weight: %6.2f', minimum_weight));
disp(sprintf('\n\n'));
%divide figure in one row, two columns, then select pane 1
subplot(1,2,1);% subplot: row=1, column=2, pane=1
% plot (linear 2-D plot)of weight data
plot(weight)
title('Weight of Freshman Class Men')
xlabel('Count(students)')
ylabel('Weight, lbs')
grid on
hold on
```

```
% plot average line
plot(1:size(weight),average_weight,'r.:');
% subplot: row=1, column=2, pane=2
subplot(1,2,2)
plot(height)
title('Height of Freshman Class Men')
xlabel('Count(students)')
ylabel('Height, inches')
average_height = mean(height);
std_height = std(height);
maximum_height = max(height);
minimum_height = min(height);
disp(sprintf('\n\tAverage Height: %6.2f', average_height));
disp(sprintf('\n\tStandard Deviation: %6.2f', std_weight));
disp(sprintf('\n\tMaximum Height: %6.2f', maximum_height));
disp(sprintf('\n\tMinimum Height: %6.2f', minimum_height));
disp(sprintf('\n\tDone'));
```

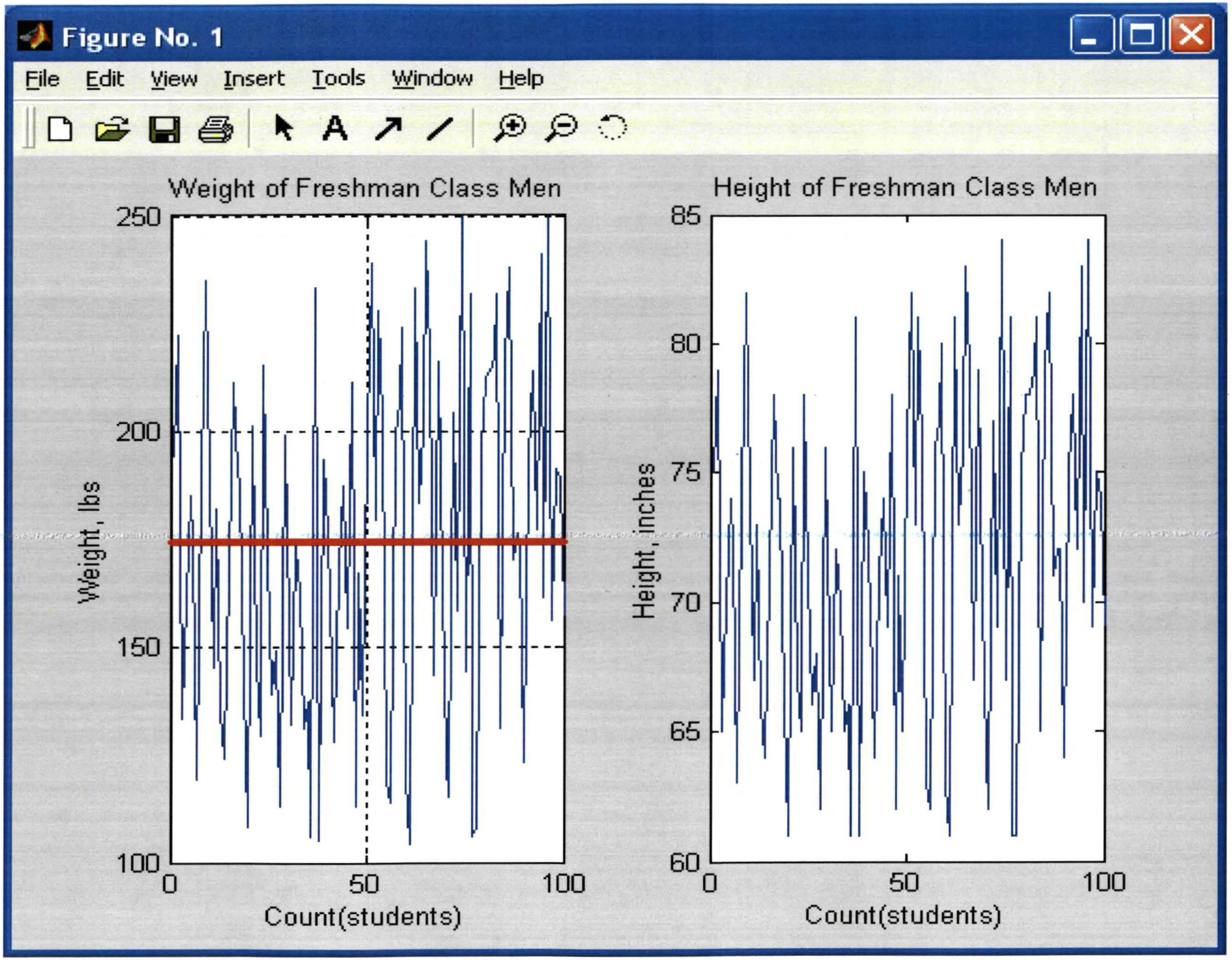


Figure No. 1
File Edit View Insert Tools Window Help
Weight of Freshman Class Men
Height of Freshman Class Men
Weight, lbs
Height, inches
Count(students)
Count(students)
250
200
150
100
85
80
75
70
65
60
0
50
100
0
50
100

```
% Simple program script – Example 2
% clear removes all variables from the workspace; to free up system
% memory.
clear all

% clc clears all input and output from the Command Window display,
% giving you a "clean screen."
clc

% The linspace function generates linearly spaced vectors. It is
% similar to the colon operator ":", but gives direct control over
% the number of points.
% y = linspace(a,b,n) generates a row vector y of n points linearly
% spaced between and including a and b.
%
% x=linspace(0,10*pi,1000);
```

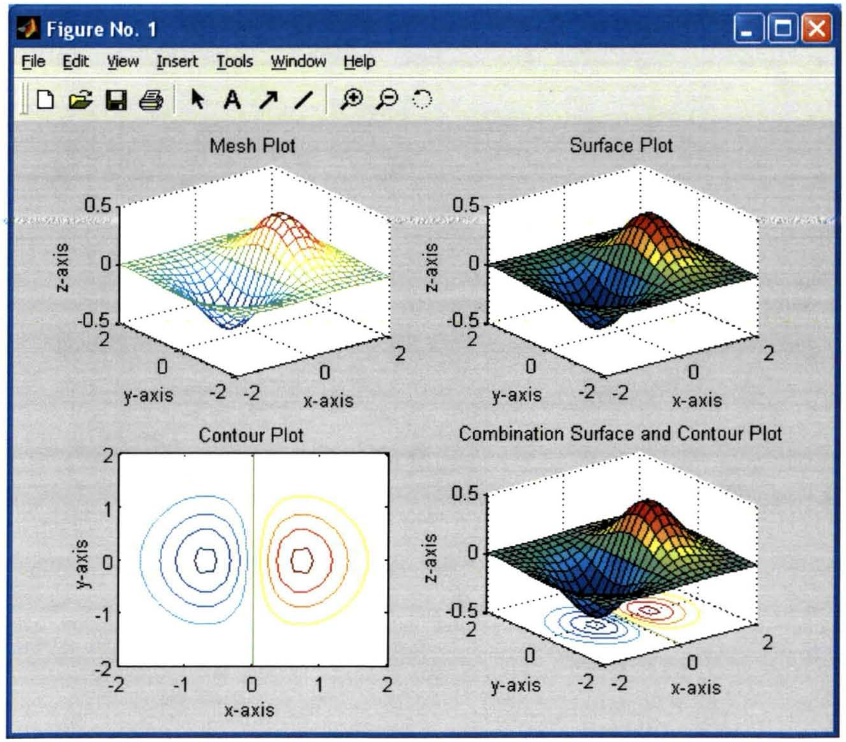
```
% The Colon Operator (:) is used to create a vector containing:
% -2.0000   -1.8000   -1.6000 ..... 1.6000    1.8000    2.0000
% with 21 cells
x=[-2:0.2:2];
y=[-2:0.2:2];

% meshgrid - Generates X and Y matrices for three-dimensional plots
% [X,Y] = meshgrid(x,y) transforms the domain specified by vectors x
% and y into arrays X and Y, which can be used to evaluate functions
% of two variables and three-dimensional mesh/surface plots. The rows
% of the output array X are copies of the vector x; columns of the
% output array Y are copies of the vector y.
%
% For example, the [X,Y] = meshgrid(1:3,10:14) are:
%
```

```
% X =
%
%    1   2   3
%    1   2   3
%    1   2   3
%    1   2   3
%    1   2   3
%
% Y =
%
%   10  10  10
%   11  11  11
%   12  12  12
%   13  13  13
%   14  14  14
```

```
[X,Y]=meshgrid(x,y);

% .* operator is used to perform an element by element multiplication
Z=X.*exp(-X.^2-Y.^2);

subplot(2,2,1)

% mesh(X,Y,Z) draws a wireframe mesh with color determined by Z so color
% is proportional to surface height.
mesh(X,Y,Z)
title('Mesh Plot')
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
```

```
subplot(2,2,2)
% surf(X,Y,Z) creates a shaded surface using Z for the color data as well
% as surface height.
surf(X,Y,Z)
title('Surface Plot')
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')

subplot(2,2,3)
% contour(X,Y,Z)draws contour plot of Z
contour(X,Y,Z)
title('Contour Plot')
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
```

```
subplot(2,2,4)
% surfc(...) draws a contour plot beneath the surface.
surfc(X,Y,Z)
title('Combination Surface and Contour Plot')
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
```



M-file function

Syntax

function [out1, out2, ...] = funname(in1, in2, ...)

function [out1, out2, ...] = funname(in1, in2, ...) defines function funname that accepts inputs in1, in2, etc. and returns outputs out1, out2, etc.

Multiple functions within an M-File

MatLab allows multiple subfunctions within a M-File

These subfunctions are not visible to other functions in the other M-Files

A function is terminated using an end statement or use a return statement to force an early return

Local variables

The variables within the body of the function are all local variables

When MATLAB does not recognize a function by name, it searches for a file of the same name on disk. If the function is found, MATLAB compiles it into memory for subsequent use

When you call an M-file function from the command line or from within another M-file, MATLAB parses the function and stores it in memory. The parsed function remains in memory until cleared with the clear command or you quit MATLAB

Example

M-File Name: stat.m

Input: vector x

Output: mean and stdev

```
function [mean, stdev] = stat(x)
  n = length(x);
  mean = sum(x)/n;
  stdev = sqrt(sum((x-mean).^2/n));
end
```

function_handle (@)

Function handle is used to call functions indirectly

Syntax
handle = @functionname

handle = @functionname returns a handle to the specified MATLAB function

Function handle

A function handle is a MATLAB value that provides a means of calling a function indirectly

M-Files

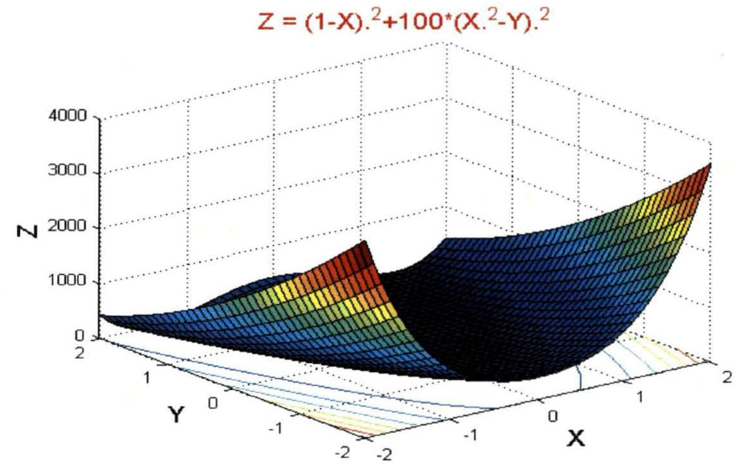
rosenbrock.m function

test.m script

minimize.m function

Test calls minimize function, function Handle to rosenbrock function is passed as parameter to minimize function

```
% Partial example
function y = rosenbrock()
    % Rosenbrock banana function
    % The minimum is at (1,1),  (y=8.1777e-010).
    % The traditional starting point is (-1.2,1).

        y =100*(x(2)-x(1)^2)^2+(1-x(1))^2;
end
```



```
% MatLab Script
% File name: Test
x=[-1.2 1];
[x, y , history] = minimize(@rosenbrock,x);
x
y
history
 m1=min(history(:,1));
 m2=max(history(:,1));
 m3=min(history(:,2));
 m4=max(history(:,2));
 m1,m2,m3,m4
........
```

```
% File Name: minimize.m
function [x, y , history] = minimize(functionHandle, x_init)
   history = [];
   options = optimset('OutputFcn', @myoutput);
   [x y] = fminsearch(functionHandle, x_init,options);
   function stop = myoutput(x,optimvalues,state);
      stop = [];
      if state == 'iter'
        history = [history; x];
      end
   end
end
```

```
% MatLab Script – Example

x=[-2:0.2:2];
y=[-2:0.2:2];
[X,Y]=meshgrid(x,y);
Z =100*(Y-X.^2).^2+(1-X).^2;
subplot(2,2,1)


% mesh(X,Y,Z) draws a wireframe mesh with color determined by Z so color
% is proportional to surface height.
mesh(X,Y,Z)
title('Mesh Plot')
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
subplot(2,2,2)
```

```
% MatLab Script
% surf(X,Y,Z) creates a shaded surface using Z for the color data as well
% as surface height.
surf(X,Y,Z)
title('Surface Plot')
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')

subplot(2,2,3)
% contour(X,Y,Z)draws contour plot of Z
contour(X,Y,Z)
title('Contour Plot')
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
```

```
% MatLab Script
% File name: minimizeTest

subplot(2,2,4)
% surfc(...) draws a contour plot beneath the surface.
surfc(X,Y,Z)
title('Combination Surface and Contour Plot')
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')

close all; % close all previously drawn graphs
figure; % create a new figure/plot
grid on;
title('History points')
xlabel('x(1)');
ylabel('x(2)');
plot(history(:,1),history(:,2), '--rs','LineWidth',2, 'MarkerEdgeColor',
    'k',  'MarkerFaceColor','g', 'MarkerSize',2)
```

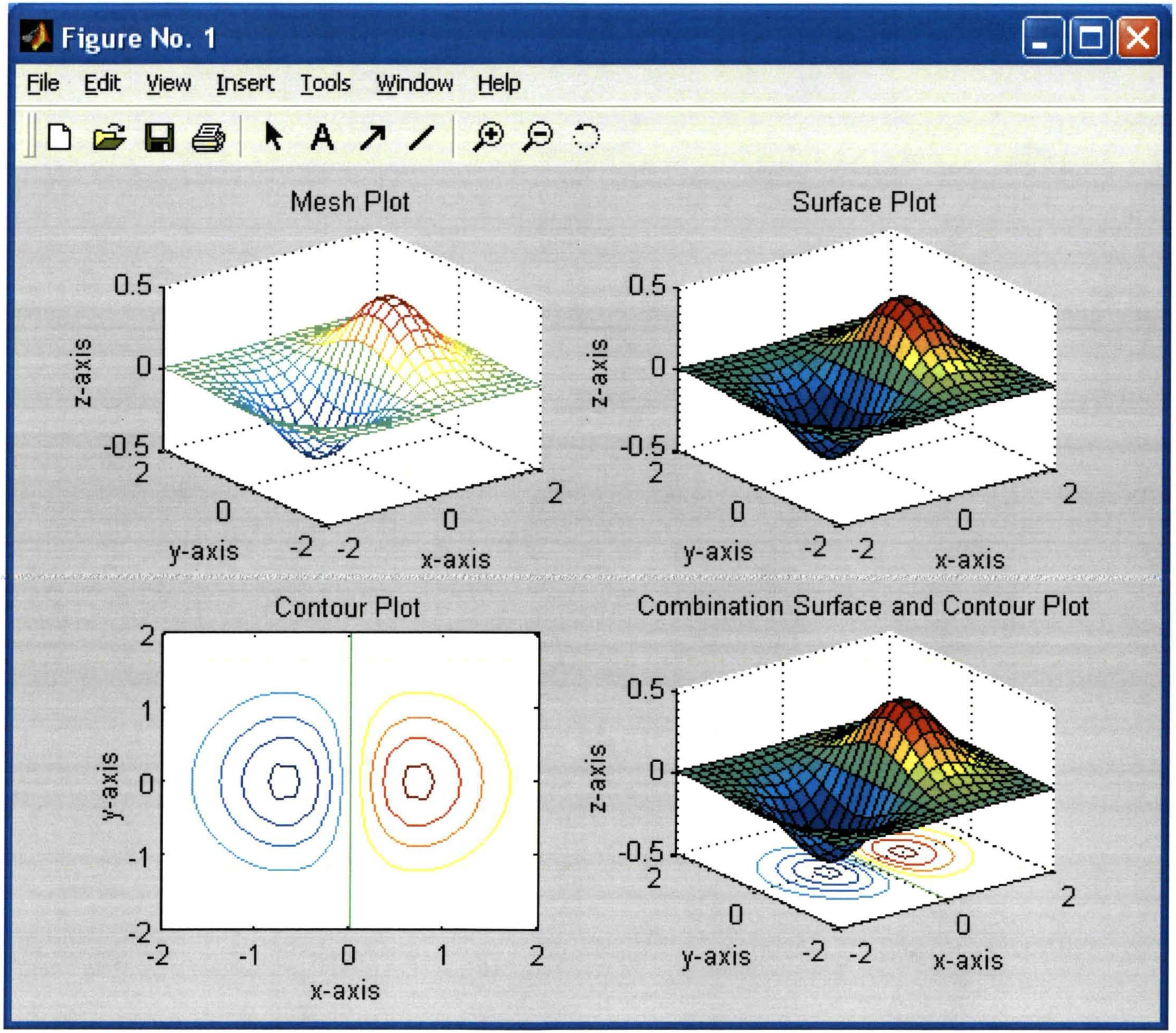


Figure No. 1
File Edit View Insert Tools Window Help
Mesh Plot
Surface Plot
Contour Plot
Combination Surface and Contour Plot
z-axis
y-axis
x-axis
0.5
0
-0.5
2
-2
1
-1

% The first 300,000 digits of pi

Pi with million digits: $\pi \quad 2\sin^{-1}(1)$

3.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117067
9821480865132823066470938446095505822317253594081284811174502841027019385211055596446229489549303819
6442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127
3724587006606315588174881520920962829254091715364367892590360011330530548820466521384146951941511609
4330572703657595919530921861173819326117931051185480744623799627495673518857527248912279381830119491
2983367336244065664308602139494639522473719070217986094370277053921717629317675238467481846766940513
2000568127145263560827785771342757789609173637178721468440901224953430146549585371050792279689258923
5420199561121290219608640344181598136297747713099605187072113499999983729780499510597317328160963185
9502445945534690830264252230825334468503526193118817101000313783875288658753320838142061717766914730
3598253490428755468731159562863882353787593751957781857780532171226806613001927876611195909216420198
9380952572010654858632788659361533818279682303019520353018529689957736225994138912497217752834791315
1557485724245415069595082953311686172785588907509838175463746493931925506040092770167113900984882401
2858361603563707660104710181942955596198946767837449448255379774726847104047534646208046684259069491
2933136770289891521047521620569660240580381501935112533824300355876402474964732639141992726042699227
9678235478163600934172164121992458631503028618297455570674983850549458858692699569092721079750930295
5321165344987202755960236480665499119881834797753566369807426542527862551818417574672890977772793800
0816470600161452491921732172147723501414419735685481613611573525521334757418494684385233239073941433
3454776241686251898356948556209921922218427255025425688767179049460165346680498862723279178608578438
3827967976681454100953883786360950680064225125205117392984896084128488626945604241965285022210661186
3067442786220391949450471237137869609563643719172874677646575739624138908658326459958133904780275900
9946576407895126946839835259570982582262052248940772671947826848260147699090264013639443745530506820
3496252451749399651431429809190659250937221696461515709858387410597885959772975498930161753928468138
2686838689427741559918559252459539594310499725246808459872736446958486538367362226260991246080512438
8439045124413654976278079771569143599770012961608944169486855584840635342207222582848864815845602850
6016842739452267467678895252138522549954666727823986456596116354886230577456498035593634568174324112
5150760694794510965960940252288797108931456691368672287489405601015033086179286809208747609178249385
8900971490967598526136554978189312978482168299894872265880485756401427047755513237964145152374623436
4542858444795265867821051141354735739523113427166102135969536231442952484937187110145765403590279934
4037420073105785390621983874478084784896833214457138687519435064302184531910484810053706146806749192
7

```
% Utility program to group digits of Pi
% (group of 1, 10 and 15 digits) –
% Use the first 300000 digits of Pi
fid = fopen('PI.txt');
a = fscanf(fid,'%s'); % It has two rows
now.

b=[];
 for d=1:300000
    if(a(1,d)>='0' && a(1,d)<='9')
       a(1,d);
       b=[b,a(1,d)];
    end
 end

size(b);
k=10;  l=1; m=15;
fid1 = fopen('PI10Out-150000.txt','w');
fid2 = fopen('PI15Out-150000.txt','w');
fid3 = fopen('PI1Out-150000.txt','w');

% group of 15, 5 groups per row
for d=1:20000
  ac1=b(l:m);    l=l+15; m=m+15;
  ac2=b(l:m);    l=l+15; m=m+15;
  ac3=b(l:m);    l=l+15; m=m+15;
  ac4=b(l:m);    l=l+15; m=m+15;
  ac5=b(l:m);    l=l+15; m=m+15;
  fprintf(fid2,'%15s %15s %15s %15s ...
             %15s\n',ac1,ac2,ac3,ac4,ac5);
end
```

```
% group of 10, 5 groups per row
for d=1:30000
  ab1=b(j:k);    j=j+10; k=k+10;
  ab2=b(j:k);    j=j+10; k=k+10;
  ab3=b(j:k);    j=j+10; k=k+10;
  ab4=b(j:k);    j=j+10; k=k+10;
  ab5=b(j:k);    j=j+10; k=k+10;
  fprintf(fid1,'%10s %10s %10s %10s %10s\n',ab1, ab2, ab3 ,ab4 ,ab5 );
End
% group of 1, 75 groups per row
j=1; k=1;
for d=1:2000
   b1=[];
   for b2=1:75
     b1=[b1,b(j:k)];
     b1=[b1,' '];
     j=j+1; k=k+1;
   end
   fprintf(fid3,'%s\n',b1);
End
fclose(fid); fclose(fid1); fclose(fid2); fclose(fid3);
```

% The first 300,000 digits of pi

Pi with million digits: $\bar{\pi} \quad 2\sin^{-1}(1)$

3.1415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253421170679
8214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196
4428810975665933446128475648233786783165271201909145648566923460348610454326648213393607260249141273
7245870066063155881748815209209628292540917153643678925903600113305305488204665213841469519415116094
3305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912
9833673362440656643086021394946395224737190702179860943702770539217176293176752384674818467669405132
0005681271452635608277857713427577896091736371787214684409012249534301465495853710507922796892589235
4201995611212902196086403441815981362977477130996051870721134999999837297804995105973173281609631859
5024459455346908302642522308253344685035261931188171010003137838752886587533208381420617177669147303
5982534904287554687311595628638823537875937519577818577805321712268066130019278766111959092164201989
3809525720106548586327886593615338182796823030195203530185296899577362259941389124972177528347913151
5574857242454150695950829533116861727855889075098381754637464939319255060400927701671139009848824012
8583616035637076601047101819429555961989467678374494482553797747268471040475346462080466842590694912
9331367702898915210475216205696602405803815019351125338243003558764024749647326391419927260426992279
6782354781636009341721641219924586315030286182974555706749838505494588586926995690927210797509302955
3211653449872027559602364806654991198818347977535663698074265425278625518184175746728909777727938000
8164706001614524919217321721477235014144197356854816136115735255213347574184946843852332390739414333
4547762416862518983569485562099219222184272550254256887671790494601653466804988627232791786085784383
8279679766814541009538837863609506800642251252051173929848960841284886269456042419652850222106611863
0674427862203919494504712371378696095636437191728746776465757396241389086583264599581339047802759009
9465764078951269468398352595709825822620522489407726719478268482601476990902640136394437455305068203
4962524517493996514314298091906592509372216964615157098583874105978859597729754989301617539284681382
6868386894277415599185592524595395943104997252468084598727364469584865383673622262609912460805124388
4390451244136549762780797715691435997700129616089441694868555848406353422072225828488648158456028506
0168427394522674676788952521385225499546667278239864565961163548862305774564980355936345681743241125
1507606947945109659609402522887971089314566913686722874894056010150330861792868092087476091782493858
9009714909675985261365549781893129784821682998948722658804857564014270477555132379641451523746234364
5428584447952658678210511413547357395231134271661021359695362314429524849371871101457654035902799344
0374200731057853906219838744780847848968332144571386875194350643021845319104848100537061468067491927

```
% Utility Program to count digits 0 – 9 in the first  300000 digits of Pi
%
fid = fopen('PI.txt');
b = fscanf(fid,'%s'); % It has two rows now.

d1=0; d2=0;  d3=0;  d4=0;  d5=0;d6=0; d7=0;  d8=0;  d9=0;  d0=0;

a=[];

for d=1:300000
    if(b(1,d)>='0' && b(1,d)<='9')
       a=[a,b(1,d)];
    end
end
size(b)
% Change 300000 to a desire count
%
```

```
% Program utility to count 0 – 9 in the first  300000 digits of Pi
% inside 'file name'
%
fid = fopen('PI.txt');
b = fscanf(fid,'%s'); % It has two rows now.

d1=0; d2=0;  d3=0;  d4=0;  d5=0;d6=0; d7=0;  d8=0;  d9=0;  d0=0;

a=[];

for d=1:300000
    if(b(1,d)>='0' && b(1,d)<='9')
       a=[a,b(1,d)];
    end
end
size(b)
```

```
% Change 300000 to a desire count
%
for d=1:300000
   if a(1,d) == '1'    d1=d1+1;
   elseif a(1,d) == '2' d2=d2+1;
   elseif a(1,d) == '3' d3=d3+1;
   elseif a(1,d) == '4' d4=d4+1;
   elseif a(1,d) == '5' d5=d5+1;
   elseif a(1,d) == '6' d6=d6+1;
   elseif a(1,d) == '7' d7=d7+1;
   elseif a(1,d) == '8' d8=d8+1;
   elseif a(1,d) == '9' d9=d9+1;
   elseif a(1,d) == '0' d0=d0+1;
   else
      a(1,d)
   end
end
fclose(fid);
% show counts of digits 0-9
d0, d1, d2, d3, d4, d5, d6, d7, d8, d9
```

Thank you!