```
disp("Scientific Computing is the collection of tools, techniques and theories required to solv
```

```
Scientific Computing is the collection of tools, techniques and theories required to solve on a computer mathematica
```

```
disp("problems n science and engineering")
```

```
problems n science and engineering
```

```

disp("Mathematical modelling is the application of mathematics to describe real world problems
```

```
Mathematical modelling is the application of mathematics to describe real world problems and investigating importan
```

```
disp("that arise from it. ")
```

```
that arise from it.
```

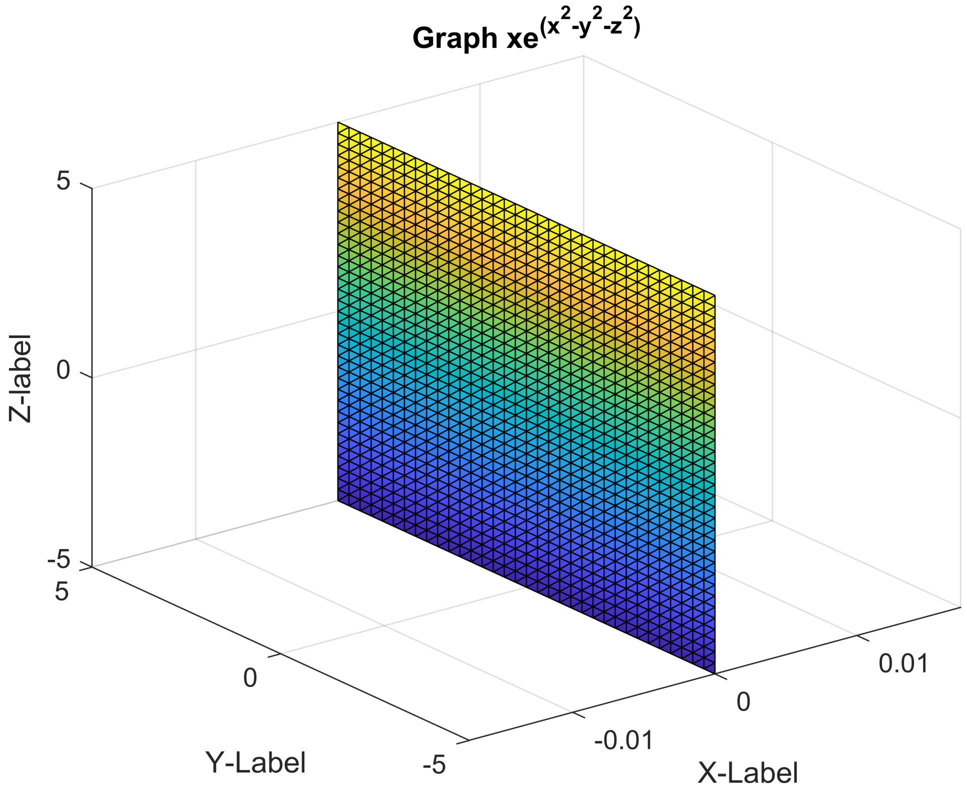
```

disp("An algorithm is a finite step by step process typically used to solve a class of specific
```

```
An algorithm is a finite step by step process typically used to solve a class of specific problems.
```

```

warning("off")
syms x y z
g = x.*exp(x.^2-y.^2-z.^2)
```

g = $x\,e^{x^2-y^2-z^2}$

```
fimplicit3(g),grid on,xlabel('X-Label'),...
    ylabel('Y-Label'),zlabel('Z-label'),...
    title('Graph xe^{(x^2-y^2-z^2)}')
```



```
syms f(x) f(y) f(z)

f = 3*x^2*y - 4*z^3 + 5
```

f = $3\,y\,x^2 - 4\,z^3 + 5$

```
diff(f,x) + diff(f,y) + diff(f,z)
```

ans = $3\,x^2 + 6\,y\,x - 12\,z^2$

```
data = [43 35 90 56 77 12 45 67 89 12 34 56 78 90 12 45 67 89 56 78]
```

```
data = 1×20
    43    35    90    56    77    12    45    67    89    12    34    56    78 ···
```

```
mode(data)
```

```
ans = 12
```

```
mean(data)
```

```
ans = 56.5500
```

```
median(data)
```

```
ans = 56
```

```
var(data)
```

```
ans = 698.8921
```

```
std(data)
```

```
ans = 26.4366
```

```


r = [1 2 3 4; 5 7 3 2 ; 12 45 6 7; 2 1 4 6];
max(r)
```

```
ans = 1×4
    12    45     6     7
```

```
min(min(r))
```

```
ans = 1
```

```
t =[2:3,1:3]
```

```
t = 1×5
     2     3     1     2     3
```

```

help("isprime")
```

```
 isprime True for prime numbers.
    isprime(X) is 1 for the elements of X that are prime, 0 otherwise.
 
    Class support for input X:
       float: double, single
       integer: uint8, int8, uint16, int16, uint32, int32, uint64, int64
 
    See also factor, primes.

    Documentation for isprime
    Other functions named isprime
```

```
help("primes")
```

```
 primes Generate list of prime numbers.
    primes(N) is a row vector of the prime numbers less than or 
    equal to N.  A prime number is one that has no factors other
    than 1 and itself.
 
    Class support for input N:
       float: double, single
       integer: uint8, int8, uint16, int16, uint32, int32, uint64, int64
 
    See also factor, isprime.

    Documentation for primes
```

```
help("rand")
```

```
 rand Uniformly distributed pseudorandom numbers.
    R = rand(N) returns an N-by-N matrix containing pseudorandom values drawn
    from the standard uniform distribution on the open interval(0,1).  rand(M,N)
    or rand([M,N]) returns an M-by-N matrix.  rand(M,N,P,...) or
    rand([M,N,P,...]) returns an M-by-N-by-P-by-... array.  rand returns a
    scalar.  rand(SIZE(A)) returns an array the same size as A.

    Note: The size inputs M, N, P, ... should be nonnegative integers.
    Negative integers are treated as 0.

    R = rand(..., CLASSNAME) returns an array of uniform values of the
    specified class. CLASSNAME can be 'double' or 'single'.

    R = rand(..., 'like', Y) returns an array of uniform values of the
    same class as Y.

    The sequence of numbers produced by rand is determined by the settings of
    the uniform random number generator that underlies rand, RANDI, and RANDN.
    Control that shared random number generator using RNG.

    Examples:

       Example 1: Generate values from the uniform distribution on the
       interval (a, b).
          r = a + (b-a).*rand(100,1);

       Example 2: Use the RANDI function, instead of rand, to generate
       integer values from the uniform distribution on the set 1:100.
          r = randi(100,1,5);

       Example 3: Reset the random number generator used by rand, RANDI, and
       RANDN to its default startup settings, so that rand produces the same
       random numbers as if you restarted MATLAB.
          rng('default')
          rand(1,5)

       Example 4: Save the settings for the random number generator used by
       rand, RANDI, and RANDN, generate 5 values from rand, restore the
       settings, and repeat those values.
          s = rng
          u1 = rand(1,5)
          rng(s);
          u2 = rand(1,5) % contains exactly the same values as u1

       Example 5: Reinitialize the random number generator used by rand,
       RANDI, and RANDN with a seed based on the current time.  rand will
       return different values each time you do this.  NOTE: It is usually
       not necessary to do this more than once per MATLAB session.
          rng('shuffle');
          rand(1,5)

    See Replace Discouraged Syntaxes of rand and randn to use RNG to replace
    rand with the 'seed', 'state', or 'twister' inputs.

    See also randi, randn, rng, RandStream, RandStream/rand,
             sprand, sprandn, randperm.

    Documentation for rand
    Other functions named rand
```

```
help("randi")
```

```
 randi Pseudorandom integers from a uniform discrete distribution.
    R = randi(IMAX,N) returns an N-by-N matrix containing pseudorandom
    integer values drawn from the discrete uniform distribution on 1:IMAX.
    randi(IMAX,M,N) or randi(IMAX,[M,N]) returns an M-by-N matrix.
```

```
randi(IMAX,M,N,P,...) or randi(IMAX,[M,N,P,...]) returns an
M-by-N-by-P-by-... array.  randi(IMAX) returns a scalar.
randi(IMAX,SIZE(A)) returns an array the same size as A.

R = randi([IMIN,IMAX],...) returns an array containing integer
values drawn from the discrete uniform distribution on IMIN:IMAX.

Note: The size inputs M, N, P, ... should be nonnegative integers.
Negative integers are treated as 0.

R = randi(..., CLASSNAME) returns an array of integer values of class
CLASSNAME.

R = randi(..., 'like', Y) returns an array of integer values of the
same class as Y.

The arrays returned by randi may contain repeated integer values.  This
is sometimes referred to as sampling with replacement.  To get unique
integer values, sometimes referred to as sampling without replacement,
use RANDPERM.

The sequence of numbers produced by randi is determined by the settings of
the uniform random number generator that underlies RAND, RANDN, and randi.
randi uses one uniform random value to create each integer random value.
Control that shared random number generator using RNG.

Examples:

   Example 1: Generate integer values from the uniform distribution on
   the set 1:10.
      r = randi(10,100,1);

   Example 2: Generate an integer array of integer values drawn uniformly
   from 1:10.
      r = randi(10,100,1,'uint32');

   Example 3: Generate integer values drawn uniformly from -10:10.
      r = randi([-10 10],100,1);

   Example 4: Reset the random number generator used by RAND, randi, and
   RANDN to its default startup settings, so that randi produces the same
   random numbers as if you restarted MATLAB.
      rng('default');
      randi(10,1,5)

   Example 5: Save the settings for the random number generator used by
   RAND, randi, and RANDN, generate 5 values from randi, restore the
   settings, and repeat those values.
      s = rng
      i1 = randi(10,1,5)
      rng(s);
      i2 = randi(10,1,5) % i2 contains exactly the same values as i1

   Example 6: Reinitialize the random number generator used by RAND,
   randi, and RANDN with a seed based on the current time.  randi will
   return different values each time you do this.  NOTE: It is usually
   not necessary to do this more than once per MATLAB session.
      rng('shuffle');
      randi(10,1,5)

See also rand, randn, randperm, rng, RandStream

Documentation for randi
Other functions named randi
```

```
help("base2dec")
```

```
 base2dec Convert text representation of number in base B to double value
    D = base2dec(S,B) converts the integer represented by S, a number in
    base B, to the equivalent decimal number (base 10) and returns D as a
    double-precision value.  B must be an integer between 2 and 36. S must
    represent a non-negative integer value.

    If S represents an integer greater than or equal to FLINTMAX,
    then base2dec might not represent it exactly as a double-precision
    floating-point value.

    S can be a character array, a cell array of character vectors, or a
    string array. If S is a character array, each row is taken to represent
    a number in base B.

    Example
       base2dec('212',3) returns 23

    See also dec2base, hex2dec, bin2dec, flintmax.

    Documentation for base2dec
```

```
help("dec2binary")
```

```
dec2binary not found.

Use the Help browser search field to search the documentation, or
type "help help" for help command options, such as help for methods.
```

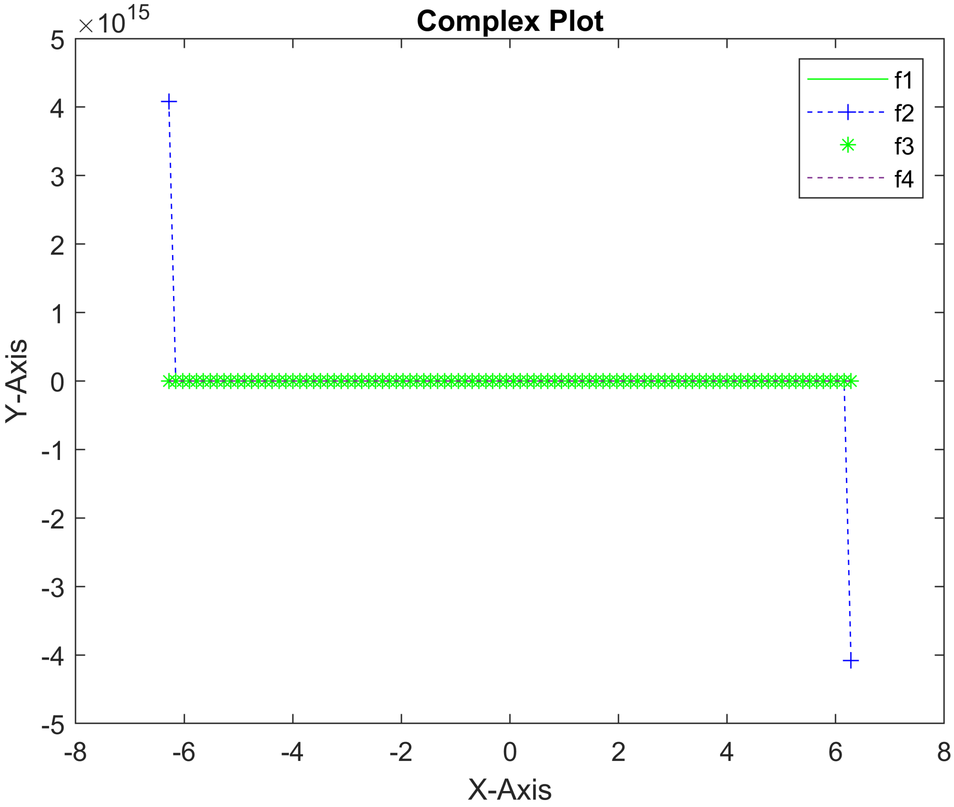
```
syms f1(x) f2(x) f3(x) f4(x)

x = linspace(-2*pi,2*pi)
```

```
x = 1×100
   -6.2832   -6.1563   -6.0293   -5.9024   -5.7755   -5.6485   -5.5216   -5.3947 ⋯
```

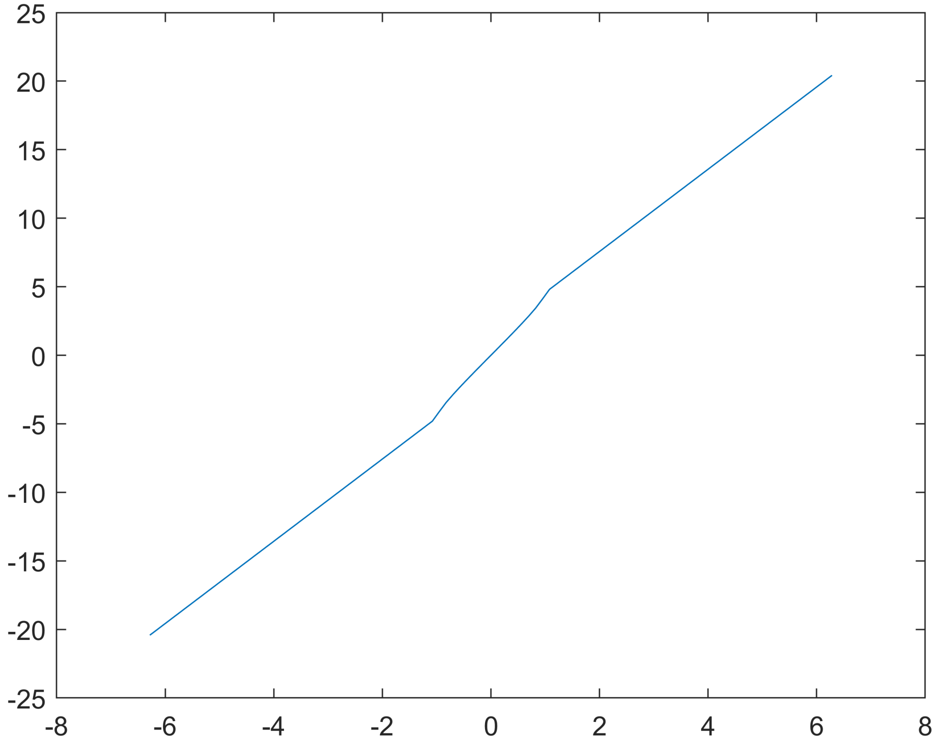
```
f1 = asin(x) + 3*x ;
f2 = 1 ./ tan(x);
f3 = 3*exp(3*x);
f4 = 2*sin(x);

figure
p = plot(x,f1,'g',x,f2,'b--o',x,f3,'c*',x,f4,'--'),xlabel('X-Axis')...
    ,ylabel('Y-Axis'),title('Complex Plot'),legend('f1','f2','f3','f4')
```

```
p = 
  4×1 Line array:

  Line    (f1)
  Line    (f2)
  Line    (f3)
  Line    (f4)
```

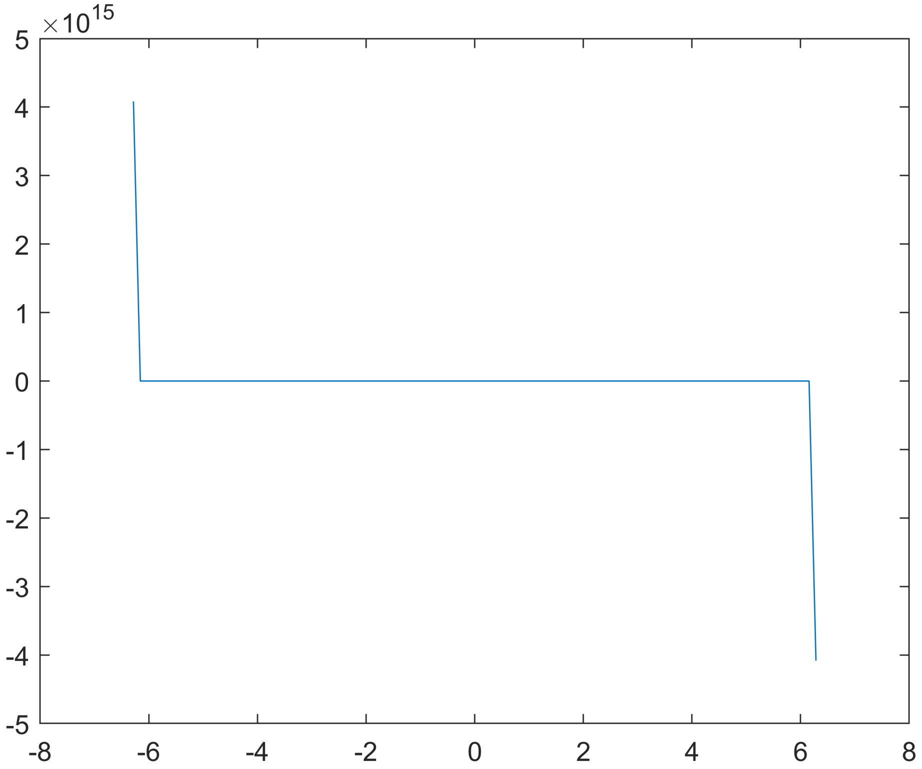
```
p(1).LineWidth = 0.5;
p(2).Marker = "+";
p(3).Color = "green";
```


Complex Plot
×10^15
5
4
3
2
1
0
-1
-2
-3
-4
-5
-8
-6
-4
-2
0
2
4
6
8
X-Axis
Y-Axis
f1
f2
f3
f4

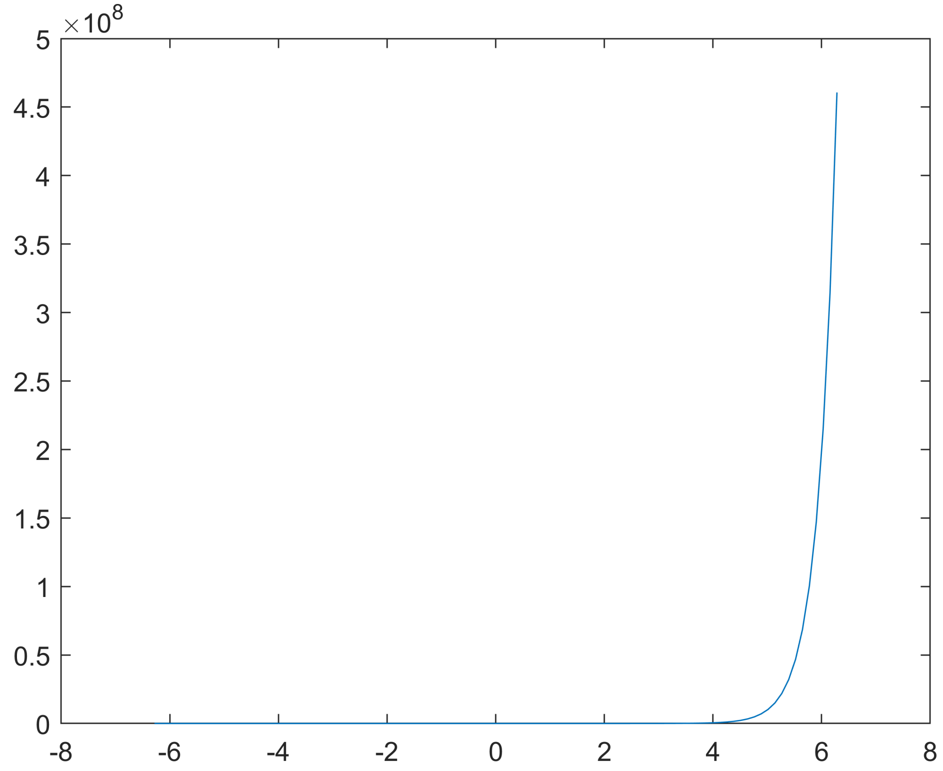
```
plot(x,f1)
```


25
20
15
10
5
0
-5
-10
-15
-20
-25
-8
-6
-4
-2
0
2
4
6
8

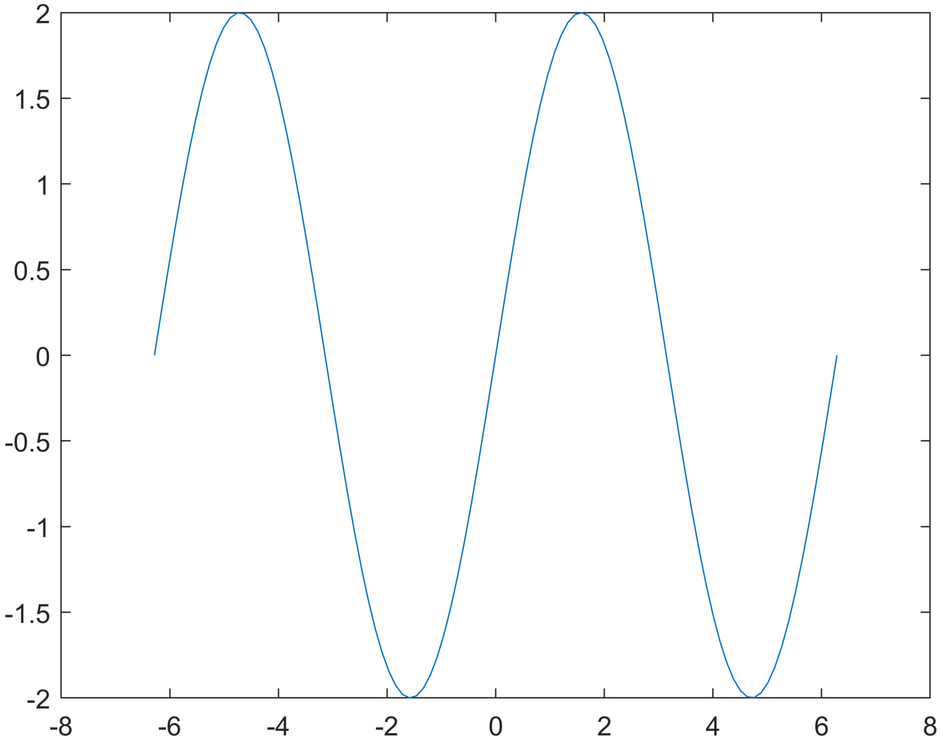
```
plot(x,f2)
```



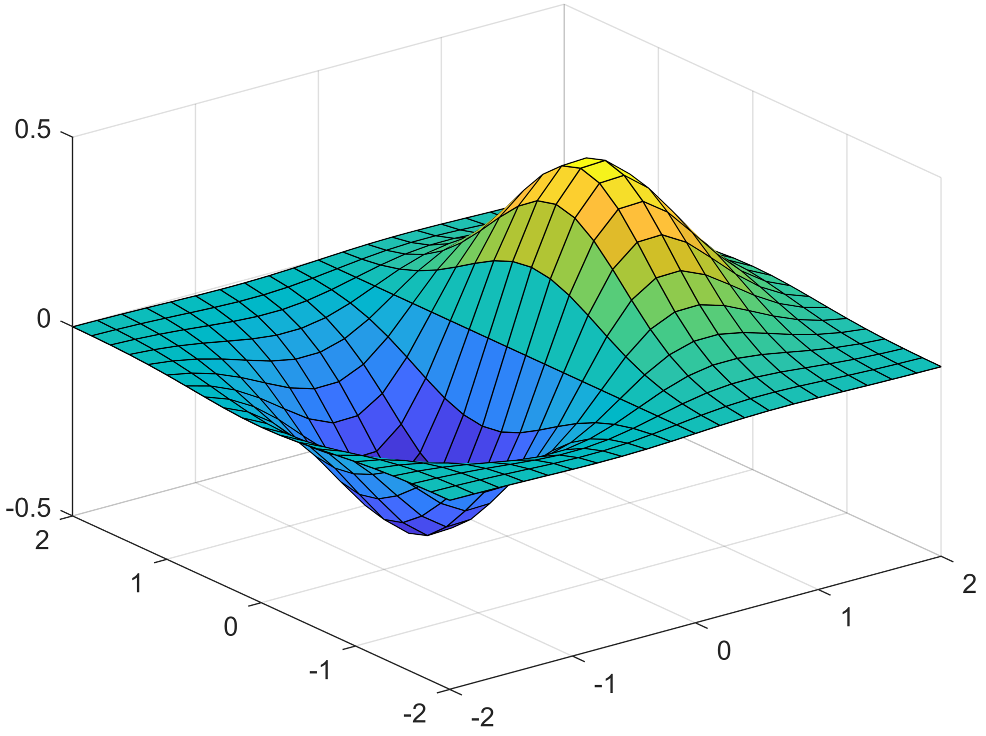
```
plot(x,f3)
```


×10^8
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
-8
-6
-4
-2
0
2
4
6
8

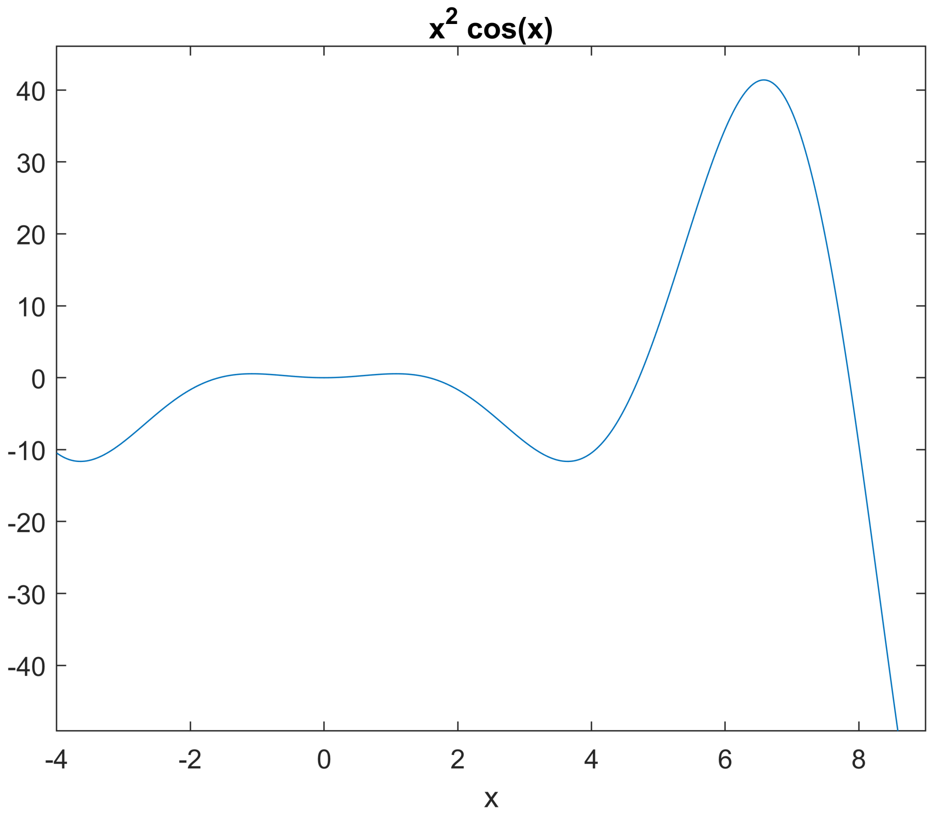
```
plot(x,f4)
```


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-8
-6
-4
-2
0
2
4
6
8

```
[x,y] = meshgrid(-2:.2:2);
g = x .* exp(-x.^2 - y.^2);
surf(x, y,g)
```



```
syms x
f = x^2*cos(x);
ezplot(f, [-4,9])
```



```
a = int(f, -4, 9)
```

a = $8\cos(4)+18\cos(9)+14\sin(4)+79\sin(9)$

```
disp('Area: '), disp(double(a));
```

```
Area: 
    0.3326
```

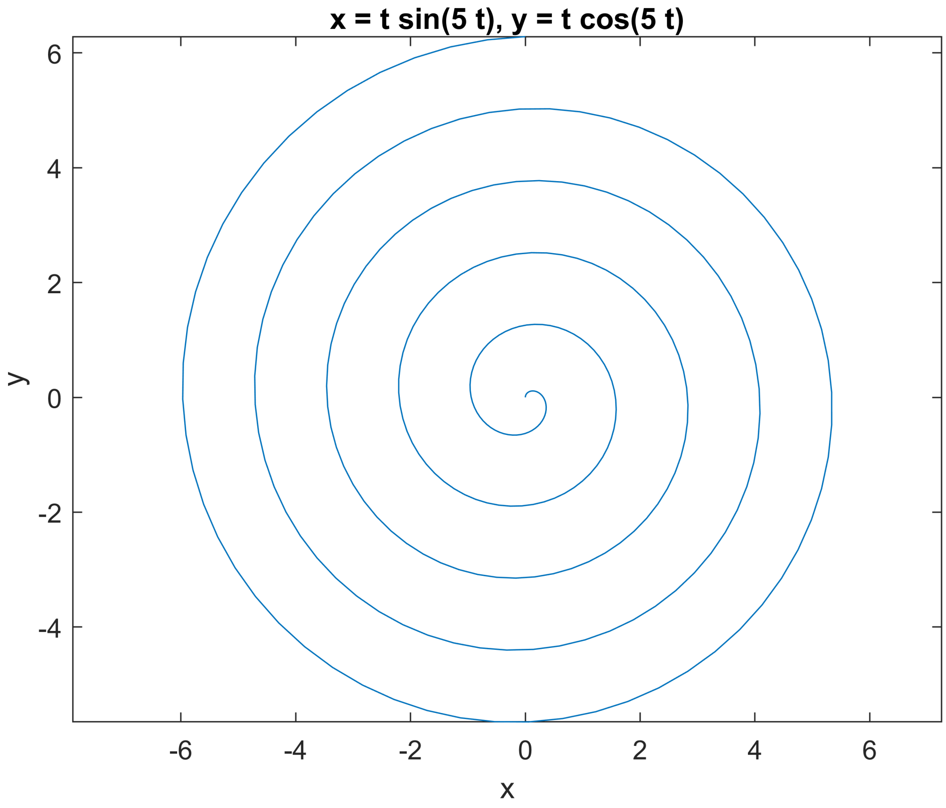
```

syms t
x = t*sin(5*t)
```

x = $t\sin(5\,t)$

```
y = t*cos(5*t)
```

y = $t\cos(5\,t)$

```
ezplot(x, y)
```



```
syms x
x
```

x = x

```
int(x)
```

ans =

$$\frac{x^2}{2}$$

```
x^3-2*x+5
```

ans = $x^3 - 2\,x + 5$

```
int(x^3-2*x+5, x)
```

ans =

$$\frac{x\,\left(x^3 - 4\,x + 20\right)}{4}$$

```
cos(x)
```

ans = $\cos(x)$

```
int(cos(x))
```

ans = $\sin(x)$

```
sin(x)
```

ans = $\sin(x)$

```
int(sin(x))
```

ans = $-\cos(x)$

```
tan(x)
```

ans = $\tan(x)$

```
int(tan(x))
```

ans = $-\log(\cos(x))$

```
asin(x)
```

ans = $\mathrm{asin}(x)$

```
int(asin(x))
```

ans = $x\,\mathrm{asin}(x) + \sqrt{1 - x^2}$

```
acos(x)
```

ans = $\mathrm{acos}(x)$

```
int(acos(x))
```

ans = $x\,\mathrm{acos}(x) - \sqrt{1 - x^2}$

```
atan(x)
```

ans = $\mathrm{atan}(x)$

```
int(atan(x))
```

ans =

$$x\,\mathrm{atan}(x) - \frac{\log(x^2 + 1)}{2}$$

```
sec(x)
```

ans =

$$\frac{1}{\cos(x)}$$

```
int(sec(x))
```

ans =

$\log\left(\frac{1}{\cos(x)}\right)+\log(\sin(x)+1)$

```
cot(x)
```

ans = $\cot(x)$

```
int(cot(x))
```

ans = $\log(\sin(x))$

```

disp("differentiation")
```

```
differentiation
```

```

syms x y z
y = exp(x)
```

y = e^x

```

diff(y,x)
```

ans = e^x

```

clear y

disp("Limits")
```

```
Limits
```

```
fu = (x-3) / (x-1)
```

fu =

$$\frac{x-3}{x-1}$$

```
limit(fu, 1000000000000000000)
```

ans =

$$\frac{999999999999999997}{999999999999999999}$$

```

syms x y
eq = x^3 -3*x^2 + 3*x - 1 == 0
```

eq = $x^3-3\,x^2+3\,x-1=0$

```
solve(eq)
```

ans =

$$\begin{pmatrix}1\\1\\1\end{pmatrix}$$

```
a = 5*x+ 9*y==5
```

a = $5\,x+9\,y=5$

```
b = 3*x-6*y==4
```

b = $3\,x-6\,y=4$

```

s= solve(a,b);
s.x
```

ans =

$$\frac{22}{19}$$

```
s.y
```

ans =

$$-\frac{5}{57}$$

```

clear a b

a = 2*x + 3*y + 2*z == 70
```

a = $2\,x+3\,y+2\,z=70$

```
b = 3*x + 3*y + 4*z == 95
```

b = $3\,x+3\,y+4\,z=95$

```
c = x + y + z == 30
```

c = $x+y+z=30$

```

s = solve(a,b,c);
disp("x =")
```

```
x =
```

```
s.x
```

ans = 15

```
disp("y = ")
```

```
y =
s.y
```

```
ans = 10
```

```
disp("z = ")
```

```
z =
```

```
s.z
```

```
ans = 5
```

```

n = input ( 'What\" s your \"favorite\" number?' )
```

```
n = 15
```

```
twiceYourFavoritePlusOne = 2*n + 1
```

```
twiceYourFavoritePlusOne = 31
```

```
myvector = [2 ,8 ,3]
```

```
myvector = 1×3
     2     8     3
```

```
myvectort = [ 2 : 3 : 6 ]
```

```
myvectort = 1×2
     2     5
```

```
if (rem(n,2) == 0 )
    disp('even' )
else
    disp ('odd')
end
```

```
odd
```

```
age4review = [19 7 4 2; 1 14 89 62; 2 3 2 12]
```

```
age4review = 3×4
    19     7     4     2
     1    14    89    62
     2     3     2    12
```