```
>> A = [1 2; 3 4], B = [5 6; 7 8]
A =
     1     2
     3     4
B =
     5     6
     7     8
>> A + B
ans =
     6     8
    10    12
>> A - B
ans =
    -4    -4
    -4    -4
>> A * B
ans =
    19    22
    43    50
>> A .* B
ans =
     5    12
    21    32
>> A \ B
ans =
   -3.0000   -4.0000
    4.0000    5.0000
>> A .\ B
ans =
    5.0000    3.0000
    2.3333    2.0000
>> A / B
ans =
    3.0000   -2.0000
    2.0000   -1.0000
>> A ./ B
ans =
    0.2000    0.3333
    0.4286    0.5000
>> A ^ 2
ans =
     7    10
    15    22
>> A .^ 2
ans =
     1     4
     9    16
>> A ^ B
Error using  ^
Inputs must be a scalar and a square matrix.
To compute elementwise POWER, use POWER (.^) instead.
>> A .^ B
ans =
           1         64
```

```
        2187       65536
>> sum(A)
ans =
     4     6
>> sum(A')
ans =
     3     7
>> sum(A, 2)
ans =
     3
     7
>> a
Undefined function or variable 'a'.
Did you mean:
>> A
A =
     1     2
     3     4
>> diag(A)
ans =
     1
     4
>> diag(A, 2)
ans =
   Empty matrix: 0-by-1
>> fliplr(A)
ans =
     2     1
     4     3
>> flipud(A)
ans =
     3     4
     1     2
>> help sum
 sum Sum of elements.
    S = sum(X) is the sum of the elements of the vector X. If X is a
matrix,
    S is a row vector with the sum over each column. For N-D arrays,
    sum(X) operates along the first non-singleton dimension.

    S = sum(X,DIM) sums along the dimension DIM.

    S = sum(..., TYPE) specifies the type in which the
    sum is performed, and the type of S. Available options are:

    'double'    -  S has class double for any input X
    'native'    -  S has the same class as X
    'default'   -  If X is floating point, that is double or single,
                   S has the same class as X. If X is not floating
point,
                   S has class double.

    S = sum(..., MISSING) specifies how NaN (Not-A-Number) values
are
```

```
    treated. The default is 'includenan':

    'includenan' - the sum of a vector containing NaN values is also
NaN.
    'omitnan'    - the sum of a vector containing NaN values
                   is the sum of all its non-NaN elements. If all
                   elements are NaN, the result is 0.

    Examples:
    If   X = [0 1 2; 3 4 5]
    then sum(X, 1) is [3 5 7] and sum(X, 2) is [3; 12]

    If X = int8(1:20) then sum(X) accumulates in double and the
    result is double(210) while sum(X,'native') accumulates in
    int8, but overflows and saturates to int8(127).

    See also prod, cumsum, diff, accumarray, isfloat.

    Other functions named sum

    Reference page in Help browser
       doc sum

>> help diag
 diag Diagonal matrices and diagonals of a matrix.
    diag(V,K) when V is a vector with N components is a square
matrix
    of order N+ABS(K) with the elements of V on the K-th diagonal. K
= 0
    is the main diagonal, K > 0 is above the main diagonal and K < 0
    is below the main diagonal.

    diag(V) is the same as diag(V,0) and puts V on the main
diagonal.

    diag(X,K) when X is a matrix is a column vector formed from
    the elements of the K-th diagonal of X.

    diag(X) is the main diagonal of X. diag(diag(X)) is a diagonal
matrix.

    Example
       m = 5;
       diag(-m:m) + diag(ones(2*m,1),1) + diag(ones(2*m,1),-1)
    produces a tridiagonal matrix of order 2*m+1.

    See also spdiags, triu, tril, blkdiag.

    Other functions named diag

    Reference page in Help browser
       doc diag

>> help fliplr
```

```
 fliplr Flip array in left/right direction.
    Y = fliplr(X) returns X with the order of elements flipped left
to right
    along the second dimension. For example,

    X = 1 2 3     becomes  3 2 1
        4 5 6              6 5 4

    See also flipud, rot90, flip.

    Other functions named fliplr

    Reference page in Help browser
       doc fliplr

>> help flipud
 flipud Flip array in up/down direction.
    Y = flipud(X) returns X with the order of elements flipped
upside down
    along the first dimension.  For example,

    X = 1 4      becomes  3 6
        2 5               2 5
        3 6               1 4

    See also fliplr, rot90, flip.

    Other functions named flipud

    Reference page in Help browser
       doc flipud

>> cat(1, A, B)
ans =
     1     2
     3     4
     5     6
     7     8
>> A
A =
     1     2
     3     4
>> B
B =
     5     6
     7     8
>> help cat
 cat Concatenate arrays.
    cat(DIM,A,B) concatenates the arrays A and B along
    the dimension DIM.
    cat(2,A,B) is the same as [A,B].
    cat(1,A,B) is the same as [A;B].

    B = cat(DIM,A1,A2,A3,A4,...) concatenates the input
```

```
    arrays A1, A2, etc. along the dimension DIM.

    When used with comma separated list syntax, cat(DIM,C{:}) or
    cat(DIM,C.FIELD) is a convenient way to concatenate a cell or
    structure array containing numeric matrices into a single
matrix.

    Examples:
      a = magic(3); b = pascal(3);
      c = cat(4,a,b)
    produces a 3-by-3-by-1-by-2 result and
      s = {a b};
      for i=1:length(s),
        siz{i} = size(s{i});
      end
      sizes = cat(1,siz{:})
    produces a 2-by-2 array of size vectors.

    See also num2cell.

    Other functions named cat

    Reference page in Help browser
       doc cat

>> cat(2, A, B)
ans =
     1     2     5     6
     3     4     7     8
>> cat(1, A', B')
ans =
     1     3
     2     4
     5     7
     6     8
>> cat(1, A, B')
ans =
     1     2
     3     4
     5     7
     6     8
>> vertcat(A,B)
ans =
     1     2
     3     4
     5     6
     7     8
>> horzcat(A,B)
ans =
     1     2     5     6
     3     4     7     8
>> rot90(A)
ans =
     2     4
```

```
     1     3
>> rot90(A')
ans =
     3     4
     1     2
>> help rot

rot 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.

>> help rot90
 rot90  Rotate array 90 degrees.
    B = rot90(A) is the 90 degree counterclockwise rotation of
matrix A. If
    A is an N-D array, rot90(A) rotates in the plane formed by the
first and
    second dimensions.

    rot90(A,K) is the K*90 degree rotation of A, K = +-1,+-2,...

    Example,
        A = [1 2 3      B = rot90(A) = [ 3 6
             4 5 6 ]                     2 5
                                         1 4 ]

    See also flipud, fliplr, flip.

    Other functions named rot90

    Reference page in Help browser
       doc rot90
>> M
M =
     1     4     7    10
     2     5     8    11
     3     6     9    12
>> B = reshape(A, 2, 6)
Error using reshape
To RESHAPE the number of elements must not change.
>> shape(A)
Undefined function or variable 'shape'.
Did you mean:
>> B = reshape(A, 4, 3)
Error using reshape
To RESHAPE the number of elements must not change.
>> size(A)
ans =
     2     2
>> size(B)
ans =
     2     2
>> B = reshape(M, 4, 3)
```

```
B =
     1     5     9
     2     6    10
     3     7    11
     4     8    12
>> B = reshape(M, 2, 6)
B =
     1     3     5     7     9    11
     2     4     6     8    10    12
>> help reshape
 reshape Reshape array.
    reshape(X,M,N) or reshape(X,[M,N]) returns the M-by-N matrix
    whose elements are taken columnwise from X. An error results
    if X does not have M*N elements.

    reshape(X,M,N,P,...) or reshape(X,[M,N,P,...]) returns an
    N-D array with the same elements as X but reshaped to have
    the size M-by-N-by-P-by-.... The product of the specified
    dimensions, M*N*P*..., must be the same as NUMEL(X).

    reshape(X,...,[],...) calculates the length of the dimension
    represented by [], such that the product of the dimensions
    equals NUMEL(X). The value of NUMEL(X) must be evenly divisible
    by the product of the specified dimensions. You can use only one
    occurrence of [].

    See also squeeze, shiftdim, colon.

    Other functions named reshape

    Reference page in Help browser
       doc reshape

>> x = 1
x =
     1
>> y = 1.0
y =
     1
>> z = [1]
z =
     1
>> a = '1'
a =
1
>> b = ['1']
b =
1
>> x == y
ans =
     1
>> x == z
ans =
     1
```

```
>> y == z
ans =
     1
>> x == a
ans =
     0
>> b
b =
1
>> x == b
ans =
     0
>> z == b
ans =
     0
>> str2num(a)==x
ans =
     1
>> int2str(x)==b
ans =
     1
>> help str2num
 str2num Convert string matrix to numeric array.
    X = str2num(S) converts a character array representation of a
matrix of
    numbers to a numeric matrix. For example,

         S = ['1 2'         str2num(S) => [1 2;3 4]
              '3 4']

    The numbers in the string matrix S should be ASCII character
    representations of a numeric values.  Each number may contain
digits,
    a decimal point, a leading + or - sign, an 'e' or 'd' preceding
a
    power of 10 scale factor, and an 'i' or 'j' for a complex unit.

    If the string S does not represent a valid number or matrix,
    str2num(S) returns the empty matrix.  [X,OK]=str2num(S) will
    return OK=0 if the conversion failed.

    CAUTION: str2num uses EVAL to convert the input argument, so
side
    effects can occur if the string contains calls to functions.
Use
    STR2DOUBLE to avoid such side effects or when S contains a
single
    number.

    Also spaces can be significant.  For instance, str2num('1+2i')
and
    str2num('1 + 2i') produce x = 1+2i while str2num('1 +2i')
produces
    x = [1 2i].  These problems are also avoided when you use
```

```
STR2DOUBLE.

    See also str2double, num2str, hex2num, char.

    Other functions named str2num

    Reference page in Help browser
       doc str2num

>> help int2str
 int2str Convert integer to string.
    S = int2str(X) rounds the elements of the matrix X to integers
and
    converts the result into a string matrix.
    Return NaN and Inf elements as strings 'NaN' and 'Inf',
respectively.

    See also num2str, sprintf, fprintf, mat2str.

    Other functions named int2str

    Reference page in Help browser
       doc int2str

>> a = [1 2 3; 3 2 1;1 2 4]
a =
     1     2     3
     3     2     1
     1     2     4
>> a > 2
ans =
     0     0     1
     1     0     0
     0     0     1
>> a == 1
ans =
     1     0     0
     0     0     1
     1     0     0
>> (a>1) & (a<3)
ans =
     0     1     0
     0     1     0
     0     1     0
>> a(a>2)
ans =
     3
     3
     4
>> a(a>2) = 5
a =
     1     2     5
     5     2     1
     1     2     5
```

```
>> a = linspace(1, 100, 20)
a =
  Columns 1 through 11
    1.0000    6.2105   11.4211   16.6316   21.8421   27.0526
32.2632   37.4737
42.6842   47.8947   53.1053
  Columns 12 through 20
   58.3158   63.5263   68.7368   73.9474   79.1579   84.3684
89.5789   94.7895
100.0000
>> help linspace
 linspace Linearly spaced vector.
    linspace(X1, X2) generates a row vector of 100 linearly
    equally spaced points between X1 and X2.

    linspace(X1, X2, N) generates N points between X1 and X2.
    For N = 1, linspace returns X2.

    Class support for inputs X1,X2:
       float: double, single

    See also logspace, colon.

    Other functions named linspace

    Reference page in Help browser
       doc linspace

>> ele6 = a[6]
 ele6 = a[6]
          |
Error: Unbalanced or unexpected parenthesis or bracket.

>> ele6 = a(6)
ele6 =
   27.0526
>> b = [123;456;789]
b =
   123
   456
   789
>> b = [1 2 3;4 5 6;7 8 9]
b =
     1     2     3
     4     5     6
     7     8     9
>> x = b(2, 2)
x =
     5
>> help sub2ind
 sub2ind Linear index from multiple subscripts.
    sub2ind is used to determine the equivalent single index
    corresponding to a given set of subscript values.
```

```
    IND = sub2ind(SIZ,I,J) returns the linear index equivalent to
the
    row and column subscripts in the arrays I and J for a matrix of
    size SIZ.

    IND = sub2ind(SIZ,I1,I2,...,IN) returns the linear index
    equivalent to the N subscripts in the arrays I1,I2,...,IN for an
    array of size SIZ.

    I1,I2,...,IN must have the same size, and IND will have the same
size
    as I1,I2,...,IN. For an array A, if IND =
sub2ind(SIZE(A),I1,...,IN)),
    then A(IND(k))=A(I1(k),...,IN(k)) for all k.

    Class support for inputs I,J:
       float: double, single
       integer: uint8, int8, uint16, int16, uint32, int32, uint64,
int64

    See also ind2sub.

    Other functions named sub2ind

    Reference page in Help browser
       doc sub2ind

>> help ind2sub
 ind2sub Multiple subscripts from linear index.
    ind2sub is used to determine the equivalent subscript values
    corresponding to a given single index into an array.

    [I,J] = ind2sub(SIZ,IND) returns the arrays I and J containing
the
    equivalent row and column subscripts corresponding to the index
    matrix IND for a matrix of size SIZ.
    For matrices, [I,J] = ind2sub(SIZE(A),FIND(A>5)) returns the
same
    values as [I,J] = FIND(A>5).

    [I1,I2,I3,...,In] = ind2sub(SIZ,IND) returns N subscript arrays
    I1,I2,..,In containing the equivalent N-D array subscripts
    equivalent to IND for an array of size SIZ.

    Class support for input IND:
       float: double, single
       integer: uint8, int8, uint16, int16, uint32, int32, uint64,
int64

    See also sub2ind, find.

    Other functions named ind2sub

    Reference page in Help browser
```

```
       doc ind2sub

>> sub2ind(size(b), 2, 2)
ans =
     5
>> [R C] = ind2sub([3,3],5)
R =
     2
C =
     2
>> a = [1, 2, 3, 4, 5]
a =
     1     2     3     4     5
>> a(3) = []
a =
     1     2     4     5
>> a = [2, 3, 1, 4, 5]
a =
     2     3     1     4     5
>> a(a==3) = []
a =
     2     1     4     5


>> load results
>> whos
  Name       Size            Bytes  Class     Attributes

  A          2x2                32  double
  B          2x6                96  double
  C          1x1                 8  double
  M          3x4                96  double
  R          1x1                 8  double
  a          1x4                32  double
  all       38x9             41276  cell
  ans        1x69               138  char
  b          3x3                72  double
  ele6       1x1                 8  double
  i          1x1                16  double    complex
  num       37x6              1776  double
  text      38x7             30372  cell
  x          1x1                 8  double
  y          1x1                 8  double
  z          1x1                 8  double

>> x = [1:10]
x =
     1     2     3     4     5     6     7     8     9    10
>> y = [x.^2]
y =
     1     4     9    16    25    36    49    64    81   100
>> plot(x, y)
>> plot(x)
>> plot(y)
```

```
>> plot(x, y)
>> help plot
 plot   Linear plot.
    plot(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
    then the vector is plotted versus the rows or columns of the
matrix,
    whichever line up.  If X is a scalar and Y is a vector,
disconnected
    line objects are created and plotted as discrete points
vertically at
    X.

    plot(Y) plots the columns of Y versus their index.
    If Y is complex, plot(Y) is equivalent to plot(real(Y),imag(Y)).
    In all other uses of plot, the imaginary part is ignored.

    Various line types, plot symbols and colors may be obtained with
    plot(X,Y,S) where S is a character string made from one element
    from any or all the following 3 columns:

           b     blue          .     point              -     solid
           g     green         o     circle             :     dotted
           r     red           x     x-mark             -.
dashdot
           c     cyan          +     plus               --    dashed
           m     magenta       *     star             (none)  no
line
           y     yellow        s     square
           k     black         d     diamond
           w     white         v     triangle (down)
                               ^     triangle (up)
                               <     triangle (left)
                               >     triangle (right)
                               p     pentagram
                               h     hexagram

    For example, plot(X,Y,'c+:') plots a cyan dotted line with a
plus
    at each data point; plot(X,Y,'bd') plots blue diamond at each
data
    point but does not draw any line.

    plot(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...) combines the plots defined
by
    the (X,Y,S) triples, where the X's and Y's are vectors or
matrices
    and the S's are strings.

    For example, plot(X,Y,'y-',X,Y,'go') plots the data twice, with
a
    solid yellow line interpolating green circles at the data
points.

    The plot command, if no color is specified, makes automatic use
```

```
of
    the colors specified by the axes ColorOrder property.  By
default,
    plot cycles through the colors in the ColorOrder property.  For
    monochrome systems, plot cycles over the axes LineStyleOrder
property.

    Note that RGB colors in the ColorOrder property may differ from
    similarly-named colors in the (X,Y,S) triples.  For example, the
    second axes ColorOrder property is medium green with RGB [0 .5
0],
    while plot(X,Y,'g') plots a green line with RGB [0 1 0].

    If you do not specify a marker type, plot uses no marker.
    If you do not specify a line style, plot uses a solid line.

    plot(AX,...) plots into the axes with handle AX.

    plot returns a column vector of handles to lineseries objects,
one
    handle per plotted line.

    The X,Y pairs, or X,Y,S triples, can be followed by
    parameter/value pairs to specify additional properties
    of the lines. For example, plot(X,Y,'LineWidth',2,'Color',[.6 0
0])
    will create a plot with a dark red line width of 2 points.

    Example
       x = -pi:pi/10:pi;
       y = tan(sin(x)) - sin(tan(x));
       plot(x,y,'--rs','LineWidth',2,...
                       'MarkerEdgeColor','k',...
                       'MarkerFaceColor','g',...
                       'MarkerSize',10)

    See also plottools, semilogx, semilogy, loglog, plotyy, plot3,
grid,
    title, xlabel, ylabel, axis, axes, hold, legend, subplot,
scatter.

    Other functions named plot

    Reference page in Help browser
       doc plot

 title  Graph title.
    title('text') adds text at the top of the current axis.


title('text','Property1',PropertyValue1,'Property2',PropertyValue2,.
..)
    sets the values of the specified properties of the title.
```

```
    title(AX,...) adds the title to the specified axes.

    H = title(...) returns the handle to the text object used as the
title.

    See also xlabel, ylabel, zlabel, text.

    Reference page in Help browser
       doc title

>> plot(x, y)
>> title('Simple x vs y plot')
>> xlabel('x')
>> ylabel('y')
>> plot(x, y, 'b--')
>> x
x =
     1     2     3     4     5     6     7     8     9    10
>> y
y =
     1     4     9    16    25    36    49    64    81   100
>> z = [x.^3]
z =
  Columns 1 through 9
           1           8          27          64         125
216
343         512         729
  Column 10
        1000
>> plot(x, z, 'rs')
>> plot(x, y, 'b--')
>> hold on
>> plot(x, z, 'rs')
>> hold off
>> plot(x, y, 'b--')
>> hold on
>> plot(x, z, 'rs')

>> a = [x.^4]
a =
  Columns 1 through 9
           1          16          81         256         625
1296
2401        4096        6561
  Column 10
       10000
>> b = [x.^5]
b =
  Columns 1 through 9
           1          32         243        1024        3125
7776
16807       32768       59049
  Column 10
      100000
```

```
>> subplot(2, 2, 1)
>> subplot(2, 2, 1);
>> subplot(2, 2, 1)
plot(x, y)
>> subplot(2, 2, 2)
plot(x, z)
subplot(2,2,3)
plot(x, a)
subplot(2, 2, 4)
plot(x, b)
>> linkaxes([subplot(2, 2, 1), subplot(2, 2, 2), subplot(2, 2, 3),
subplot(2, 2,
4)],'xy')
>>
```