Matrices: Type your matrix as follows:

Use , or space to separate entries, and ; or return after each row.

```
>> A = [4 5 6 -9;5 0 -3 6;7 8 5 0; -1 4 5 1]
```

or

```
>> A = [4,5,6,-9;5,0,-3,6;7,8,5,0;-1,4,5,1]
```

or

```
>> A = [ 4 5 6 -9
        5 0 -3 6
        7 8 5 0
        -1 4 5 1 ]
```

The output will be:

```
A =
   4   5   6  -9
   5   0  -3   6
   7   8   5   0
  -1   4   5   1
```

You can identify an entry of a matrix by

```
>> A(2,3)
ans =
   -3
```

A colon : indicates all entries in a row or column

```
>> A(2,:)
ans =
   5   0  -3   6
```

```
>> A(:,3)
ans =
     6
    -3
     5
     5
```

You can use these to modify entries

```
>> A(2,3) = 10
A =
   4   5   6  -9
   5   0  10   6
   7   8   5   0
  -1   4   5   1
```
or to add in rows or columns

```matlab
>> A(5,:) = [0 1 0 -1]
A =
    4  5  6  -9
    5  0 10  6
    7  8  5  0
   -1  4  5  1
    0  1  0  -1
```

or to delete them

```matlab
>> A(:,2) = []
A =
    4  6  -9
    5 10  6
    7  5  0
   -1  5  1
    0  0  -1
```

**Accessing Part of a Matrix:**

```matlab
>> A = [4,5,6,-9;5,0,-3,6;7,8,5,0;-1,4,5,1]
A =
    4  5  6  -9
    5  0 -3  6
    7  8  5  0
   -1  4  5  1
>> A([1 3],:)
ans =
    4  5  6  -9
    7  8  5  0
>> A(:,2:4)
ans =
    5  6  -9
    0 -3  6
    8  5  0
    4  5  1
>> A(2:3,1:3)
ans =
    5  0 -3
    7  8  5
Switching two rows in a matrix:

```matlab
g >> A([3 1],:) = A([1 3],:)
A =
  7  8  5  0
  5  0 -3  6
  4  5  6 -9
 -1  4  5  1
```

The Zero matrix:

```matlab
g >> zeros(2,3)
ans =
  0  0  0
  0  0  0

>> zeros(3)
ans =
  0  0  0
  0  0  0
  0  0  0
```

Identity Matrix:

```matlab
g >> eye(3)
ans =
  1  0  0
  0  1  0
  0  0  1
```

Matrix of Ones:

```matlab
g >> ones(2,3)
ans =
  1  1  1
  1  1  1
```

Random Matrix:

```matlab
g >> A = rand(2,3)
A =
  0.9501  0.4860  0.4565
  0.2311  0.8913  0.0185
```

Note that the random entries all lie between 0 and 1.
Transpose of a Matrix:

```matlab
A = [4,5,6,-9;5,0,-3,6;7,8,5,0;-1,4,5,1]
A =
    4     5     6    -9
    5     0    -3     6
    7     8     5     0
   -1     4     5     1
```

```matlab
>> transpose(A)
ans =
    4     5     7    -1
    5     0     8     4
    6    -3     5     5
   -9     6     0     1
```

```matlab
>> A'
ans =
    4     5     7    -1
    5     0     8     4
    6    -3     5     5
   -9     6     0     1
```

Diagonal of a Matrix:

```matlab
>> diag(A)
ans =
    4
    0
    5
    1
```

Row vector:

```matlab
>> v = [1 2 3 4 5]
v =
    1     2     3     4     5
```

Column vector:

```matlab
>> v = [1;2;3;4;5]
v =
    1
    2
    3
    4
    5
```
or use transpose operation \\
\[>> v = [1 2 3 4 5]'\]

\[v =
\begin{align*}
1 \\
2 \\
3 \\
4 \\
5
\end{align*}\]

**Forming Other Vectors:**

\[>> v = [1:5]\]

\[v =
\begin{align*}
1 & 2 & 3 & 4 & 5
\end{align*}\]

\[>> v = [10:-2:0]\]

\[v =
\begin{align*}
10 & 8 & 6 & 4 & 2 & 0
\end{align*}\]

\[>> v = \text{linspace}(0,1,6)\]

\[v =
\begin{align*}
0 & 0.2000 & 0.4000 & 0.6000 & 0.8000 & 1.0000
\end{align*}\]

**Important:** to avoid output, particularly of large matrices, use a semicolon \(;\) at the end of the line:

\[>> v = \text{linspace}(0,1,100);\]

gives a row vector whose entries are 100 equally spaced points from 0 to 1.

**Size of a Matrix:**

\[>> A = [4 5 6 -9 7; 5 0 -3 6 -2; 7 8 5 0 5; -1 4 5 1 -9]\]

\[A =
\begin{align*}
4 & 5 & 6 & -9 & 7 \\
5 & 0 & -3 & 6 & -2 \\
7 & 8 & 5 & 0 & 5 \\
-1 & 4 & 5 & 1 & -9
\end{align*}\]

\[>> \text{size}(A)\]

\[\text{ans} =
\begin{align*}
4 & 5
\end{align*}\]

\[>> [m,n] = \text{size}(A)\]

\[m =
\begin{align*}
4
\end{align*}\]

\[n =
\begin{align*}
5
\end{align*}\]
Output Formats

The command `format` is used to change output format. The default is

```matlab
>> format short
>> pi
ans =
   3.1416
>> format long
>> pi
ans =
   3.14159265358979
>> format rat
>> pi
ans =
   355/113
```

This allows you to work in rational arithmetic and gives the “best” rational approximation to the answer. Let’s return to the default.

```matlab
>> format short
>> pi
ans =
   3.1416
```
Arithmetic operators

+ Matrix addition.
   A + B adds matrices A and B. The matrices A and B must have the same dimensions unless one is a scalar (1 \times 1 matrix). A scalar can be added to anything.

   \[
   \begin{align*}
   \gg A &= \begin{bmatrix}
   4, 5, 6, -9; 5, 0, -3, 6; 7, 8, 5, 0; -1, 4, 5, 1
   \end{bmatrix} \\
   \gg B &= \begin{bmatrix}
   9, 2, 4, -9; 1, 4, -2, -6; 8, 1, 7, 0; -3, -4, 5, 9
   \end{bmatrix}
   \end{align*}
   \]

   \[
   \gg A + B
   \]

   \[
   \begin{bmatrix}
   13, 7, 10, -18 \\
   6, 4, -5, 0 \\
   15, 9, 12, 0 \\
   -4, 0, 10, 10
   \end{bmatrix}
   \]

- Matrix subtraction.
   A - B subtracts matrix A from B. Note that A and B must have the same dimensions unless one is a scalar.

   \[
   \gg A - B
   \]

   \[
   \begin{bmatrix}
   -5, 3, 2, 0 \\
   4, -4, -1, 12 \\
   -1, 7, -2, 0 \\
   2, 8, 0, -8
   \end{bmatrix}
   \]

* Scalar multiplication

   \[
   \gg 3*A - 4*B
   \]

   \[
   \begin{bmatrix}
   -24, 7, 2, 9 \\
   11, -16, -1, 42 \\
   -11, 20, -13, 0 \\
   9, 28, -5, -33
   \end{bmatrix}
   \]
* Matrix multiplication.

A*B is the matrix product of A and B. A scalar (a 1-by-1 matrix) may multiply anything. Otherwise, the number of columns of A must equal the number of rows of B.

\[ \text{>> } A \ast B \]
\[ \text{ans =} \]
\[ \begin{array}{cccc}
116 & 70 & 3 & -147 \\
3 & -17 & 29 & 9 \\
111 & 51 & 47 & -111 \\
32 & 15 & 28 & -6 \\
\end{array} \]

Note that two matrices must be compatible before we can multiply them.

*order of multiplication is important!

\[ \text{>> } v = [1 \ 2 \ 3 \ 4] \]
\[ v = \]
\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \text{>> } w = [1;2;3;4] \]
\[ w = \]
\[ \begin{array}{cccc}
1 \\
2 \\
3 \\
4 \\
\end{array} \]

\[ \text{>> } v \ast w \]
\[ \text{ans =} \]
\[ 30 \]

\[ \text{>> } w \ast v \]
\[ \text{ans =} \]
\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
2 & 4 & 6 & 8 \\
3 & 6 & 9 & 12 \\
4 & 8 & 12 & 16 \\
\end{array} \]

.* Array multiplication

A.*B denotes element-by-element multiplication. A and B must have the same dimensions unless one is a scalar.

A scalar can be multiplied into anything.

\[ \text{>> } a = [3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9] \]
\[ a = \]
\[ \begin{array}{cccccccc}
3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array} \]

\[ \text{>> } b = [8 \ 6 \ 2 \ 4 \ 5 \ 6 \ -1] \]
\[ b = \]
\[ \begin{array}{cccccccc}
8 & 6 & 2 & 4 & 5 & 6 & -1 \\
\end{array} \]
\[ \text{ans} = \\
\begin{array}{cccccc}
24 & 24 & 10 & 24 & 35 & 48 & -9
\end{array}
\]

\text{Matrix power.}

\[ C = A^n \] is \( A \) to the \( n \)-th power if \( n \) is a scalar and \( A \) is square. If \( n \) is an integer greater than one, the power is computed by repeated multiplication.

\[ \text{>> A = } \begin{bmatrix} 4 & 5 & 6 & -9 \\ 5 & 0 & -3 & 6 \\ 7 & 8 & 5 & 0 \\ -1 & 4 & 5 & 1 \end{bmatrix} \]
\[
A = \\
\begin{array}{cccc}
4 & 5 & 6 & -9 \\
5 & 0 & -3 & 6 \\
7 & 8 & 5 & 0 \\
-1 & 4 & 5 & 1
\end{array}
\]

\[ \text{>> A}^3 \]
\[
\text{ans} = \\
\begin{array}{cccccc}
501 & 352 & 351 & -651 \\
451 & 169 & -87 & 174 \\
1103 & 799 & 533 & -492 \\
445 & 482 & 413 & -182
\end{array}
\]

\text{Array power.}

\[ C = A \cdot B \] denotes element-by-element powers. \( A \) and \( B \) must have the same dimensions unless one is a scalar. A scalar can go in either position.

\[ \text{>> A = } \begin{bmatrix} 8 & 6 & 2 & 4 & 5 & 6 & -1 \end{bmatrix} \]
\[
A = \\
\begin{array}{cccccc}
8 & 6 & 2 & 4 & 5 & 6 & -1
\end{array}
\]

\[ \text{>> A}^3 \]
\[
\text{ans} = \\
\begin{array}{cccccccc}
512 & 216 & 8 & 64 & 125 & 216 & -1
\end{array}
\]

\text{Length of a Vector, Norm of a Vector, Dot Product}

\[ \text{>> u = } \begin{bmatrix} 8 & -7 & 6 & 5 & 4 & -3 & 2 & 1 & 9 \end{bmatrix} \]
\[
u = \\
\begin{array}{cccccccccccc}
8 & -7 & 6 & 5 & 4 & -3 & 2 & 1 & 9
\end{array}
\]

\[ \text{>> length(u)} \]
\[
\text{ans} = \\
9
\]
\textgreater \textgreater \text{norm}(u) \\
\text{ans} = \\
16.8819  \\
\textgreater \textgreater v = [9 \ -8 \ 7 \ 6 \ -4 \ 5 \ 0 \ 2 \ -4]  \\
v = \\
9 \ -8 \ 7 \ 6 \ -4 \ 5 \ 0 \ 2 \ -4  \\
\textgreater \text{dot}(u,v)  \\
\text{ans} = \\
135  \\
\textgreater u'^{*}v  \\
\text{ans} = \\
135  \\
\textbf{Complex vectors:}  \\
\textgreater u = [2-3i, \ 4+6i, \ -3, \ +2i]  \\
u = \\
2.0000- \ 3.0000i \ \ 4.0000+ \ 6.0000i \ \ -3.0000 \ \ 0+ \ 2.0000i  \\
\textgreater \text{conj}(u)  \\
\text{ans} = \\
2.0000+ \ 3.0000i \ \ 4.0000- \ 6.0000i \ \ -3.0000 \ \ 0- \ 2.0000i  \\
\textbf{Hermitian transpose:}  \\
\textgreater u'  \\
\text{ans} = \\
2.0000+ \ 3.0000i \ \ 4.0000- \ 6.0000i \ \ -3.0000 \ \ 0- \ 2.0000i  \\
\textgreater \text{norm}(u)  \\
\text{ans} = \\
8.8318  \\
\textgreater \text{dot}(u,u)  \\
\text{ans} = \\
78  \\
\textgreater \text{sqrt}(\text{ans})  \\
\text{ans} = \\
8.8318  \\
\textgreater u'^{*}u  \\
\text{ans} = \\
78
Solving Systems of Linear Equations

The best way of solving a system of linear equations

\[ A x = b \]

in MATLAB is to use the backslash operation \ (backwards division)

```matlab
>> A = [1 2 3; -1 0 2; 1 3 1]
A =
    1  2  3
   -1  0  2
   1  3  1

>> b = [1; 0; 0]
b =
    1
    0
    0

>> x = A \ b
x =
    0.6667
   -0.3333
    0.3333
```

The backslash is implemented by using Gaussian elimination with partial pivoting.

An alternative, but less accurate, method is to compute inverses:

```matlab
>> B = inv(A)
B =
    0.6667   -0.7778   -0.4444
   -0.3333    0.2222    0.5556
    0.3333    0.1111   -0.2222

or
>> B = A \ (-1)
B =
    0.6667   -0.7778   -0.4444
   -0.3333    0.2222    0.5556
    0.3333    0.1111   -0.2222

>> x = B * b
x =
    0.6667
   -0.3333
    0.3333
Another method is to use the command \texttt{rref}:

To solve the following system of linear equations:
\[
\begin{align*}
  x_1 + 4x_2 - 2x_3 + x_4 &= 2 \\
  2x_1 + 9x_2 - 3x_3 - 2x_4 &= 5 \\
  x_1 + 5x_2 - x_4 &= 3 \\
  3x_1 + 14x_2 + 7x_3 - 2x_4 &= 6
\end{align*}
\]
we form the augmented matrix:
\[
\begin{bmatrix}
  1 & 4 & -2 & 3 & 2 \\
  2 & 9 & -3 & -2 & 5 \\
  1 & 5 & 0 & -1 & 3 \\
  3 & 14 & 7 & -2 & 6
\end{bmatrix}
\]
\[
\texttt{>> A = [1,4,-2,3,2; 2,9,-3,-2,5; 1,5,0,-1,3; 3,14,7,-2,6]}
\]
\[
\texttt{A} = \\
\begin{bmatrix}
  1 & 4 & -2 & 3 & 2 \\
  2 & 9 & -3 & -2 & 5 \\
  1 & 5 & 0 & -1 & 3 \\
  3 & 14 & 7 & -2 & 6
\end{bmatrix}
\]
\[
\texttt{>> rref(A)}
\]
\[
\texttt{ans} = \\
\begin{bmatrix}
  1.0000 & 0 & 0 & 0 & -5.0256 \\
  0 & 1.0000 & 0 & 0 & 1.6154 \\
  0 & 0 & 1.0000 & 0 & -0.2051 \\
  0 & 0 & 0 & 1.0000 & 0.0513
\end{bmatrix}
\]

The solution is: \( x_1 = -5.0256, \ x_2 = 1.6154, \ x_3 = -0.2051, \ x_4 = 0.0513 \).

\textbf{Case 1: Infinitely many solutions:}

\[
\begin{bmatrix}
  -2 & 2 & -2; 1 & -1 & 1; 2 & -2 & 2
\end{bmatrix}
\]
\[
\texttt{A} = \\
\begin{bmatrix}
  -2 & 2 & -2 \\
  1 & -1 & 1 \\
  2 & -2 & 2
\end{bmatrix}
\]
\[
\texttt{>> b = [-8; 4; 8]}
\]
\[
\texttt{b} = \\
\begin{bmatrix}
  -8 \\
  4 \\
  8
\end{bmatrix}
\]
\[
\texttt{>> A\backslash b}
\]
\texttt{Warning: Matrix is singular to working precision.}
\[
\texttt{ans} = \\
\begin{bmatrix}
  \infty \\
  \infty \\
  \infty
\end{bmatrix}
\]
\texttt{MATLAB} is unable to find the solutions;
In this case, we can apply \texttt{rref} to the augmented matrix.

\begin{verbatim}
>> C = [A b]
C =
   -2  2  -2 -8
   1 -1  1  4
   2 -2  2  8

>> rref(C)
ans =
   1  -1  1  4
   0   0  0  0
   0   0  0  0
\end{verbatim}

You can use \texttt{rrefmovie} to see each step of Gaussian elimination.

\begin{verbatim}
>> rrefmovie(C)

Original matrix
C =
   -2  2  -2 -8
   1 -1  1  4
   2 -2  2  8

Press any key to continue. . .

pivot = C(1,1)
C =
   1  -1  1  4
   1 -1  1  4
   2 -2  2  8

Press any key to continue. . .

eliminate in column 1
C =
   1  -1  1  4
   1 -1  1  4
   2 -2  2  8

Press any key to continue. . .

C =
   1  -1  1  4
   0  0  0  0
   2 -2  2  8

Press any key to continue. . .

C =
   1  -1  1  4
   0  0  0  0
   0  0  0  0

Press any key to continue. . .

column 2 is negligible
\end{verbatim}
\[
\begin{bmatrix}
1 & -1 & 1 & 4 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

Press any key to continue. . .

column 3 is negligible

\[
\begin{bmatrix}
1 & -1 & 1 & 4 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

Press any key to continue. . .

column 4 is negligible

\[
\begin{bmatrix}
1 & -1 & 1 & 4 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

Conclusion: There are infinitely many solutions since row 2 and row 3 are all zeros.

Case 2: No solutions:

```matlab
>> A = [-2 1; 4 -2]
A =
   -2    1
   4   -2

>> b = [5; -1]
b =
    5
   -1

>> A \ b
Warning: Matrix is singular to working precision.
ans =
   Inf
   Inf

>> C = [A b]
C =
   -2    1    5    4   -2   -1

>> rref(C)
ans =
   1.0000  -0.5000    0
   0         1.0000

Conclusion: Row 2 is not all zeros, and the system is incompatible.
**Important:** If the coefficient matrix $A$ is rectangular (not square) then $A \backslash b$ gives the least squares solution (relative to the Euclidean norm) to the system $Ax = b$. If the solution is not unique, it gives the least squares solution $x$ with minimal Euclidean norm.

```matlab
>> A = [1 1; 2 1; -5, -1]
A =
    1    1
    2    1
   -5   -1
>> b = [1; 1; 1]
b =
    1
    1
    1
>> A \ b
ans =
   -0.5385
    1.7692
```

If you want the least squares solution in the square case, one trick is to add an extra equation $0 = 0$ to make the coefficient matrix rectangular:

```matlab
>> A = [-2 2 -2; 1 -1 1; 2 -2 2]
A =
   -2    2   -2
    1   -1    1
    2   -2    2
>> b = [-8; 4; 8]
b =
   -8
    4
    8
>> A \ b
Warning: Matrix is singular to working precision.
ans =
   Inf
   Inf
   Inf
```

```matlab
>> A(4,:) = 0
A =
   -2    2   -2
    1   -1    1
    2   -2    2
    0    0    0
```
>> b(4) = 0
b =
    -8
    4
    8
    0

>> A \ b
Warning: Rank deficient, rank = 1  tol = 2.6645e-15.
ans =
    4.0000
    0
    0
    0
Functions

Functions are vectors! Namely, a vector $x$ and a vector $y$ of the same length correspond to the sampled function values $(x_i, y_i)$.

To plot the function $y = x^2 - 0.5x$ first enter an array of independent variables:

```matlab
>> x = linspace(0,1,25)
>> y = x.^2 - .5*x;
>> plot(x,y)
```

The plot shows up in a new window. To plot in a different color, use

```matlab
>> plot(x,y,'r')
```

where the character string ‘r’ means red. Use the help window to see other options.

To plot graphs on top of each other, use `hold on`.

```matlab
>> hold on
>> z = exp(x);
>> plot(x,z)
>> plot(x,z,'g')
hold off will stop simultaneous plotting. Alternatively, use

```matlab
>> plot(x,y,'r',x,z,'g')
```

Surface Plots

Here $x$ and $y$ must give a rectangular array, and $z$ is a matrix whose entries are the values of the function at the array points.

```matlab
>> x = linspace(-1,1,40); y = x;
>> z = x’ * (y.^2);
>> surf(x,y,z)
```

Typing the command

```matlab
>> rotate3d
```

will allow you to use the mouse interactively to rotate the graph to view it from other angles.