Matrices are fundamental to MATLAB; they can be visualized as tables of values. The size of a matrix is $m \times n$, where m is the number of rows and n is the number of columns.

You need to become familiar with matrix generation and manipulation. To type a matrix into MATLAB you must

- begin with a square bracket, [
- separate elements in a row with commas or spaces
- use semicolons to separate rows
- end the matrix with another square bracket,]

For example, to generate the matrix

$$A = \begin{bmatrix} 1 & 1 & 5 \\ 7 & 12 & 2 \\ 32 & 16 & 8 \end{bmatrix}$$

in MATLAB, type
`A = [1 1 5; 7 12 2; 32 16 8]` or `A = [1,1,5; 7,12,2; 32,16,8]`
A (column) vector can be entered in the same way as a matrix. For example, the vector

$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

can be entered as `x = [1; 2; 3; 4]`
`linspace(a,b,n)` Generates a row vector of n equally spaced points between a and b.

The transpose of a matrix A, denoted by A^T, is obtained by reversing the rows and columns of A. That is, if $A = (a_{ij})$ then $A^T = (a_{ji})$. In MATLAB A^T is calculated by $A.'$ (i.e., a period followed by a single quote mark) or, simpler, A' (for matrices with real entries). Thus, the vector x is usually entered as

```
x=[1 2 3 4]'
```

Special Matrices:

- `zeros(n)` Generates an $n \times n$ matrix with all elements being 0.
- `zeros(m, n)` Generates an $m \times n$ matrix with 0 entries.
- `zeros(size(A))` Generates a zero matrix of the size of A.
- `ones` Generates a matrix with all elements being 1. The arguments are the same as for `zeros`.
- `eye` Generates the identity matrix. The arguments are the same as for `zeros`.

- `rand` Generates a matrix whose elements are uniformly distributed random numbers in the interval (0, 1). The arguments are the same as for `zeros`.
- `randi([MIN,MAX],n,m)` Generates an $n \times m$ matrix with integer entries drawn from MIN:MAX.
- `magic(n)` Generates an $n \times n$ matrix constructed from the integers 1 through n^2 with equal row, column, and diagonal sums.
- The `gallery` function can return many different types of test matrices for applications.

- `dot(x,y)` Dot product of vectors x and y.
- `cross(x,y)` Cross product of 3-D vectors x and y.
- `A + B` Matrix addition.
- `A - B` Matrix subtraction.
- `A*B` Matrix multiplication.
- `A^n` Matrix exponentiation.
- `A\b` The solution to Ax=b by Gaussian elimination when A is a square nonsingular matrix.
- `A.*B` Elementwise multiplication.
- `A.^p` Elementwise exponentiation.
- `A./B` Elementwise division.

- `max(x)` The maximum element of a real vector.
  `[m, i] = max(x)` also returns the element which contains the maximum value in i.
- `max(A)` A row vector containing the maximum element in each column of a matrix.
  `[m, i] = max(A)` also returns the element in each column which contains the maximum value in i.
- `min(x)` The minimum element of a real vector x.
- `min(A)` A row vector containing the minimum of the elements in each column in a matrix A.
- `mean(x)` The mean, or average, of the elements of a vector.
- `mean(A)` A row vector containing the mean of the elements in each column in a matrix.
- `norm(x)` The Euclidean length of a vector.
- `norm(A)` The matrix norm of A. Note: the norm of a matrix is not the Euclidean length of each column in the matrix.

- `prod(x)` The product of the elements of a vector.
- `prod(A)` A row vector containing the product of the elements in each column in a matrix.
- `sort(x)` Sorts the elements in increasing order of a real vector.
- `sort(A)` Sorts the elements in increasing order in each column of a real matrix.
- `sum(x)` The sum of the elements of a vector.
- `sum(A)` A row vector containing the sums of the elements in each column in a matrix.
- `range(x)` Returns the range of the values in x. For a vector input, it is the difference between the maximum and minimum values.
- `range(A)` A row vector containing the range for each column.

For a 4x3 matrix *A*:

`diag(A)` extracts the diagonal of matrix *A* as a vector.

`A(3:4,2:3)` gets those elements of *A* that are located in rows 3 to 4 and columns 2 to 3.

`A(:,4)=A(:,1)` adds a fourth column to *A* and set it equal to the first column of *A*.

`A(2:4,2:4)=eye(3)` replaces the last 3x3 submatrix of *A* by a 3x3 identity matrix.

`A([1 3],:)=[ ]` deletes the first and third rows of *A*.

`round(A)` rounds off all entries of *A*.

`A(:)` strings out all elements of *A* in a column.

`reshape(A, 2,6)` transforms *A* into a 2x6 matrix.

`rot90(A)` rotates *A* by 90^o.

`tril(A)` extracts the lower triangular part of *A*.

`triu(A)` extracts the upper triangular part of *A*.

For an nxn matrix A:

`det(A)` is the determinant of A.

`inv(A)` is the inverse of A.

`poly(A)` is a row vector with n+1 elements which are the coefficients of the characteristic polynomial.

`[V,D]=eig(A)` produces a diagonal matrix D of eigenvalues and a full matrix V whose columns are the corresponding eigenvectors.

Appendix: Polynomials

`p=[5 -3 0 2 1]` specifies the polynomial $p(x) = 5x^4 - 3x^3 + 2x + 1$.

`polyval(p,3)` gives $p(3)$.

`roots(p)` finds the roots of p.

`poly(r)` computes the coefficients of the polynomial whose roots are specified by the vector `r`.

`conv(p,q)` computes the product of the polynomials `p` and `q`.

`[q,r]=deconv(num,den)` computes the result (i.e., *quotient* and *remainder*) of dividing polynomial `num` by polynomial `den`.