1-1 <u>Entering Matrix</u>

The best way for you to get started with MATLAB is to learn how to handle matrices. You only have to follow a few basic conventions:

- Separate the elements of a row with blanks or commas.
- Use a semicolon (;) to indicate the end of each row.
- Surround the entire list of elements with square brackets, [ ].

For Example

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

MATLAB displays the matrix you just entered.

```
A =

    16   3    2    13
    5    10   11   8
    9    6    7    12
    4    15   14   1
```

Once you have entered the matrix, it is automatically remembered in the MATLAB workspace. You can refer to it simply as **A**. Also you can enter and change the values of matrix elements by using **workspace** window.

1-2 <u>Subscripts</u>

The element in row **i** and column **j** of **A** is denoted by **`A(i,j)`**. For example, **`A(4,2)`** is the number in the fourth row and second column. For the above matrix, **`A(4,2)`** is **15**. So to compute the sum of the elements in the fourth column of **A**, type

```
>> A(1,4) + A(2,4) + A(3,4) + A(4,4)
ans =
34
```

You can do the above summation, in simple way by using **sum** command.

If you try to use the value of an element outside of the matrix, it is an error.

```
>> t = A(4,5)
??? Index exceeds matrix dimensions.
```

On the other hand, if you store a value in an element outside of the matrix, the size increases to accommodate the newcomer. The initial values of other new elements are zeros.

```
>> X = A;

>> X(4,5) = 17

X =
16   3    2    13   0
5    10   11   8    0
9    6    7    12   0
4    15   14   1    17
```

1-3 Colon Operator

The colon " **:** " is one of the most important MATLAB operators. It occurs in several different forms. The expression

```
>> 1:10
```

is a row vector containing the integers from 1 to 10

```
1 2 3 4 5 6 7 8 9 10
```

To obtain nonunit spacing, specify an increment. For example,

```
>> 100:-7:50

100 93 86 79 72 65 58 51
```

Subscript expressions involving colons refer to portions of a matrix.

```
>>A(1:k,j)
```

is the first k elements of the jth column of A.

The colon by itself refers to *all* the elements in a row or column of a matrix and the keyword **end** refers to the *last* row or column. So

```
>> A(4,:)          or    >> A(4,1:end)
```
give the same action

```
ans =
4   15   14   1

>> A(2,end)
ans =
8
```

1-4 Basic Matrix Functions

Command	Description

sum(x)
`>> x=[1 2 3`
`      4 5 6];`
`>> sum(x)`
`ans =`
`       5     7     9`
	The sum of the elements of x. For matrices, sum(x) is a row vector with the sum over each column.

`>> sum(x,2)`
`ans=`
`       6`
`       15`
	sum (x,dim) sums along the dimension dim.

`>>sum(sum(x))`
`ans =`
`    21`
	In order to find the sum of elements that are stored in matrix with *n* dimensions, you must use **sum** command *n* times in cascade form, this is also applicable for **max, min, prod, mean, median** commands.

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td><code>mean(x)</code>
<code>x=[1 2 3; 4 5 6];</code>
<code>>> mean(x)</code>
<code>ans =</code>
<code>2.5 3.5 4.5</code>

<code>>> mean(x,2)</code>
<code>ans =</code>
<code>2</code>
<code>5</code>
<code>>>mean(mean(x))</code>
<code>ans =</code>
<code>3.5000</code></td><td>The average of the elements of x. For matrices, mean(x) is a row vector with the average over each column.
mean (x,dim) averages along the dimension dim.</td></tr>
<tr><td><code>zeros(N)</code>
<code>zeros(N,M)</code>
<code>>> zeros(2,3)</code>
<code>ans =</code>
<code>0 0 0</code>
<code>0 0 0</code></td><td>Produce N by N matrix of zeros.
Produce N by M matrix of zeros.</td></tr>
<tr><td><code>ones(N)</code>
<code>ones(N,M)</code>
<code>>> ones(2,3)</code>
<code>ans =</code>
<code>1 1 1</code>
<code>1 1 1</code></td><td>Produce N by N matrix of ones.
Produce N by M matrix of ones.</td></tr>
</table>

Command	Description

`size(x)`
`>> x=[1 2 3`
`      4 5 6];`
`>> size(x)`
`ans =`
`        2     3`
	return the size (dimensions) of matrix x.

`length(v)`
`>> v=[1 2 3];`
`>> length(v)`
`ans =`
`        3`
	return the length (number of elements) of vector v.

`numel(x)`
 	
`>> v =[55 63 34];`
`>> numel(v)`
`ans =`
`     3`
 	
`>> x=[1 2`
`      4 5`
`      7 8 ];`
`>> numel(x)`
`ans =`
`     6`
	returns the number of elements in array x.

Command	Description

`single quote ( ' )`
 	
`>> x=[1 2 3`
`      4 5 6`
`      7 8 9];`
`>> x'`
`ans =`
`     1     4     7`
`     2     5     8`
`     3     6     9`
`>> v=[1 2 3];`
`>> v'`
`ans =`
`     1`
`     2`
`     3`
	Matrix transpose. It flips a matrix about its main diagonal and it turns a row vector into a column vector.

`max (x)`
`>> x=[1 2 3`
`      4 5 6];`
`>> max (x)`
`ans =`
`        4     5     6`
`>> max(max(x))`
`ans =`
`     6`
	Find the largest element in a matrix or a vector.

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td><pre>min (x)
>> x=[1 2 3
      4 5 6];
>> min (x)
ans =
        1   2   3
>> min(min(x))
ans =
        1</pre></td><td>Find the smallest element in a matrix or a vector.</td></tr>
<tr><td><pre>magic(N)
>> magic(3)
ans =
        8   1   6
        3   5   7
        4   9   2</pre></td><td>produce N Magic square. This command produces valid magic squares for all N>0 except N=2.</td></tr>
<tr><td><pre>inv(x)
>> x=[1 4;
      5 8];
>> inv(x)
ans =
   -0.6667   0.3333
    0.4167   -0.0833</pre></td><td>produce the inverse of matrix x.</td></tr>
</table>

Command	Description

diag(x)
`>> x=[1   2   3`
`      4   5   6`
`      7   8   9];`
`>> diag(x)`
`ans =`
`     1`
`     5`
`     9`
`>> v=[1 2 3];`
`>> diag(v)`
`ans =`
`     1     0     0`
`     0     2     0`
`     0     0     3`
	Return the diagonal of matrix x. if x is a vector then this command produce a diagonal matrix with diagonal x.

prod(x)
`>> x=[1 2 3`
`          4 5 6];`
`>> prod(x)`
`ans =  4    10    18`
`>> prod(prod(x))`
`ans =`
`   720`
	Product of the elements of x. For matrices, Prod(x) is a row vector with the product over each column.

Command	**Description**
median(x) x=[4 6 8 10 9 1 8 2 5]; >> median(x) ans = 8 6 5 >> median(x,2) ans = 6 9 5 >> median(median(x)) ans = 6	The median value of the elements of x. For matrices, median (x) is a row vector with the median value for each column. median(x,dim) takes the median along the dimension dim of x.
sort(x,DIM,MODE) >> x = [3 7 5 0 4 2]; >> sort(x,1) ans = 0 4 2 3 7 5 >> sort(x,2) ans = 3 5 7 0 2 4 >> sort(x,2,'descend') ans = 7 5 3 4 2 0	Sort in ascending or descending order. - For vectors, sort(x) sorts the elements of x in ascending order. For matrices, sort(x) sorts each column of x in ascending order. DIM= 1 by default MODE= 'ascend' by default

Command	**Description**

`det(x)`
`>> x=[5 1 8`
`      4 7 3`
`      2 5 6];`
`>> det(x)`
`ans =`
`   165`
	Det is the determinant of the square matrix x.

`tril(x)`
`>> x=[5 1 8`
`      4 7 3`
`      2 5 6];`
`>> tril(x)`
`ans =`
`     5     0     0`
`     4     7     0`
`     2     5     6`
	Extract lower triangular part of matrix x.

`triu(x)`
`>> x=[5 1 8`
`      4 7 3`
`      2 5 6];`
`>> triu(x)`
`ans =`
`     5     1     8`
`     0     7     3`
`     0     0     6`
	Extract upper triangular part of matrix x.

Note

When we are taken away from the world of linear algebra, matrices become two-dimensional numeric arrays. Arithmetic operations on arrays are done element-by-element. This means that addition and subtraction are the same for arrays and matrices, but **that multiplicative operations are different**. MATLAB uses a **dot (.), or decimal point, as part of the notation for multiplicative array operations**.

Example: Find the factorial of 5

```
>> x=2:5;
>> prod(x)
```

Example: if x = [1,5,7,9,13,20,6,7,8], then

a) replace the first five elements of vector x with its maximum value.
b) reshape this vector into a 3 x 3 matrix.

solution

```
a)
 >> x(1:5)=max(x)

b)
 >> y(1,:)=x(1:3);
 >> y(2,:)=x(4:6);
 >> y(3,:)=x(7:9);
 >> y
```

Example: Generate the following row vector b=[1, 2, 3, 4, 5, ...9,10], then transpose it to column vector.

solution

```
>> b=1:10
b =
     1     2     3     4     5     6     7     8     9    10
>> b=b';
```

Exercises

1- If `x=[1 4; 8 3]`, find :
   a) the inverse matrix of x .
   b) the diagonal of x.
   c) the sum of each column and the sum of whole matrix x.
   d) the transpose of x.

2- If `x= [2 8 5; 9 7 1],  b=[2 4 5]` find:
   a) find the maximum and minimum of x.
   b) find median value over each row of x.
   c) add the vector **b** as a third row to x.

3- If `x=[ 2 6 12; 15 6 3; 10 11 1]`, then

   a) replace the first row elements of matrix x with its average value.
   b) reshape this matrix into row vector.

4- Generate a 4 x 4 Identity matrix.

5- Generate the following row vector b=[5, 10, 15, 20 . . . . 95, 100], then find the number of elements in this vector.