Matlab/Freemat/Octave/Scilab: Arrays – Matrices and Vectors[1]

The array[2] is a fundamental data structure in computer programming. The array or matrix is the most important construct in *Matlab*, *Freemat, Octave* and *Scilab*. Hence the name 'Matlab' or 'Matrix Laboratory'. In this handout we look at methods for setting up arrays in Matlab/Freemat/Octave/Scilab. An array with a single row or a single column is known as a line array or vector.

Line Array

A line array – a row of numbers can be defined by simply listing the numbers, separated by spaces or commas, within square brackets. For example numbers=[1 2 3 4 5] or numbers=[1,2,3,4,5] defines the line array of numbers 1 2 3 4 5 . numbers can be also be defined using a counter numbers = 1: 1: 5, meaning the numbers from one to maximum five in steps of one. These and other examples are given here. Finally the element with any index i can be found by the name of the array followed by i in brackets. For example halves(5) returns the fifth element of the array halves. The function length can be used to find the number of elements in the array.

```
numbers=[1 2 3 4 5]
numbers =
1 2 3 4 5
--> numbers=[1,2,3,4,5]
numbers =
1 2 3 4 5
--> numbers = 1: 1: 5
numbers =
1 2 3 4 5
--> length(numbers)
ans =
5
--> odd=[1 3 5 7 9]
odd =
1 3 5 7 9
--> odd= 1: 2: 9
odd =
1 3 5 7 9
fibonacci=[1 1 2 3 5]
fibonacci =
1 1 2 3 5
reverse_numbers =
5 4 3 2 1
```

[1] Matrix Definitions
[2] Arrays

```
--> halves=1: 0.5: 5
halves =
1.0000   1.5000   2.0000   2.5000   3.0000   3.5000   4.0000   4.5000   5.0000
--> halves(4)
ans =
2.5000
--> length(numbers)
ans =
5
```

Column Array

A column array can be defined in a similar way, but with the elements each being followed by a semicolon or return. Also a column array can be achieved by transposing a line array using a prime (‘). Following on from the examples above, the column $\begin{bmatrix}1\\2\\3\\4\\5\end{bmatrix}$ can be obtained by the following methods.

```
--> numbersT=[1
2
3
4
5
]
numbersT =
1
2
3
4
5
--> numbersT=[1; 2; 3; 4; 5]
numbersT =
1
2
3
4
5
--> numbersT=numbers'
numbersT =
1
2
3
4
5
```

Matrices

A matrix or rectangular array can be defined by using spaces (or commas) to separate the elements of each row and semicolons (or newline) to separate the rows. Any element of the array can be identified by its row and column. The function size can be used to determine the dimensions of an array.

```
--> A=[1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
A =
1  2  3  4
5  6  7  8
9 10 11 12
13 14 15 16
--> A(3,2)
ans =
10
 --> size(A)
ans =
4 4
```

In practice, matrices are often square, but they do need to be. In the following a non-square matrix is defined.

```
--> B = [0 1; 2 3; 4 5; 6 7]
B =
0 1
2 3
4 5
6 7
--> size(B)
ans =
4 2
```

A matrix can also be transposed.

```
--> B'
ans =
0 2 4 6
1 3 5 7
```

We can augment two matrices side-by-side if they have the same number of rows or one above the other if they have the same number of columns as follows.

```
--> [A B]
ans =
1  2  3  4  0  1
5  6  7  8  2  3
9 10 11 12  4  5
13 14 15 16  6  7
--> [A; B']
ans =
1  2  3  4
5  6  7  8
9 10 11 12
13 14 15 16
0  2  4  6
1  3  5  7
```

A particular row or column of a matrix can be extracted. For example A(2, :) is the second row of the matrix A and A(:,3) is the third column of the matrix A.

```
--> A(2, :)
ans =
5 6 7 8
--> A(:,3)
ans =
3
7
11
15
```

A sub-matrix can be extracted. For example A(2:3,2;4) represents the matrix that is composed of columns 2..3 and rows 2..4 of the matrix A.

```
--> A(2:3, 2:4)
ans =
6  7  8
10 11 12
```

We may also have arrays of complex numbers[3].

```
--> C=[1+i, -1-2i; 3, 2-i]
C =
1.0000 +  1.0000i   -1.0000 -  2.0000i
3.0000 +  0.0000i    2.0000 -  1.0000i
```

Special Matrices

Matlab/Freemat/Octave/Scilab include commands for setting up special matrices. For example zeros(*m*,*n*) sets up a m by n matrix of zeros, ones(*m*,*n*) sets up a m by n matrix of ones.

```
--> zeros(3,4)
ans =
0 0 0 0
0 0 0 0
0 0 0 0
--> ones(2,3)
ans =
1 1 1
1 1 1
```

The same functions with one argument set up the corresponding square matrices. The command eye(m) sets up a m by n matrix of zeros, but with ones down the main diagonal; an identity matrix.

```
 --> zeros(4)
ans =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
--> ones(4)
ans =
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
--> eye(4)
ans =
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

[3] Complex Numbers