Matlab/Freemat/Octave/Scilab : Array Operations

- Matrix and Vector Arithmetic[1]

Once we have covered the techniques for defining and accessing arrays (matrices or vectors)[2] we can look at the operations that are available. There are a wider range of operations availabl than there are in normal matrix-vector arithmetic[1].

Addition and Subtraction

Addition and subtraction of arrays (matrices and vectors) is only possible when the array operands have the same dimension(s) (length for line arrays (vectors) and size for rectangular arrays (matrices)). The result has the same dimensions as the operands and is equivalent to an element-by-element addition or subtraction. This is equivalent to matrix or vector addition or subtraction.

For example for line arrays (vectors) and rectangular arrays (matrices)[1]

```
--> x=[1 2 3 4 5]
 x =
1 2 3 4 5
 --> y=[0 -1 4 -5 2]
y =
r0 -1  4 -5  2
 --> x+y
ans =
1  1  7 -1  7
--> x-y
ans =
```

```
--> A=[2 -1; 1 3; 4 -2]
A =
2 -1
1  3
4 -2
--> B=[1 2; -2 1; 0 5]
B =
1  2
-2  1
0  5
--> A+B
ans =
3  1
-1  4
4  3
--> A-B
ans =
1 -3
3  2
4 -7
```

[1] Matrix Arithmetic
[2] Arrays - Matrices and Vectors

Matrix multiplication

For matrix multiplication (or matrix-vector multiplication) the second matrix must have the same number of rows as the first matrix has columns. The operator for standard matrix multiplication is the '* '. Following the examples in reference 1, we have the following.

```
--> A=[2 -1; 1 3; 4 -2]
A =
2 -1
1  3
4 -2
--> C=[1 3; -2 4]
C =
1  3
-2  4
--> A*C
ans =
4  2
-5 15
8  4
--> C*C
ans =
-5  15
-10  10
--> C^2
ans =
Columns 1 to 1
-5.0000 +  0.0000i                -10.0000 +  0.0000i
Columns 2 to 2
15.0000 +  0.0000i                 10.0000 +  0.0000i
```

Note that C^2 is computed by finding C*C and C^2 .

Matrix *Division*

Consider a system of the form

$$AX = B$$

where *A* and *B* are given matrices and *X is* a matrix to be determined. If *A, B* and *X* were numbers then we would be tempted to write

$$X = B/A\ ,$$

and this is similar to the notation used in Matlab.

In mathematics we would rather write

$$X = A^{-1}B \ .$$

In reality, we are solving linear systems of equations in carrying out an operation of this nature and this will be considered in a separate document.

Array Multiplication, Division and Power

Matlab/Freemat/Octave/Scilab support an alternative form of array multiplication. This requires that the two arrays to be multiplied have the same dimensions (size), the operation is denoted '.*' and the result is an array for which every element is equal to the product of the two corresponding elements of the array operands. Array division (operation is denoted './') is defined in a similar way. The elements of an array can be individually raised to a power, with the operator denoted '.^'.

```
 --> A=[ 1 2 3; 4 5 6]
A =
1 2 3
4 5 6
--> B=[2 4 6; 4 5 6]
B =
2 4 6
4 5 6
--> A.*B
ans =
 2  8 18
16 25 36
--> A./B
ans =
0.5000    0.5000    0.5000
1.0000    1.0000    1.0000
--> A.^2
ans =
1  4  9
16 25 36
```

Sum of the elements of an array

The elements of a rectangular array can be summed by using the sum function. For a matrix A, sum(A) returns the sums of the columns of A. The sums of the rows can be found by transposing A and applying the sum function; sum(A'). The total sum of the whole array A can be found by the command sum(sum(A)).

```
A =
1 2 3
4 5 6
--> sum(A)
ans =
5 7 9
--> sum(A')
ans =
6 15
--> sum(sum(A))
ans =
21
```

The sum of a line array can be determined in a similar way.

```
--> x=[1 2 3 4 5]
x =
1 2 3 4 5
--> sum(x)
ans =
15
```

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
```

[3] Complex Numbers