VICTORIA UNIVERSITY
MELBOURNE AUSTRALIA

Arrays in Matlab

- NEF1104 – Problem Solving for Engineers
  - Week 2 Session 4

VICTORIA UNIVERSITY
MELBOURNE AUSTRALIA

Operations on matrices and arrays

Matlab supports the standard operations of addition, subtraction, transposition and multiplication.

Recall that these operations are defined only when the matrices involved satisfy certain properties:

A + B	A and B must be the same size (same numbers of rows and columns).
A − B	A and B must be the same size.
AB	The number of columns in A must be equal to the number of rows of B.
A^2	A must be a square matrix.
A^T	No restrictions.

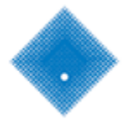
Operations on matrices and arrays

A + B	A and B must be the same size (same numbers of rows and columns).
A − B	A and B must be the same size.
AB	The number of columns in A must be equal to the number of rows of B.
A^2	A must be a square matrix.
A^T	No restrictions.

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

A =

    1    2    3    4
    5    6    7    8
    9   10   11   12

>> B = magic(3)

B =

    8    1    6
    3    5    7
    4    9    2

>> C = [10 11 12;13 14 15;16 17 18]

C =

   10   11   12
   13   14   15
   16   17   18
```

```
>> B+C

ans =

   18   12   18
   16   19   22
   20   26   20

>> B*A

ans =

   67   82   97  112
   91  106  121  136
   67   82   97  112

>> A'*C

ans =

  219  234  249
  258  276  294
  297  318  339
  336  360  384
```

```
>> B^2

ans =

   91   67   67
   67   91   67
   67   67   91

>> C^4

ans =

   782730   842292   901854
   993492  1069092  1144692
  1204254  1295892  1387530
```



VICTORIA UNIVERSITY
MELBOURNE AUSTRALIA

Dot operators

The “dot” operators (“array operators”) are a special class of commands which operate on every element of an array independently.

A*B: multiply A by B using the usual matrix definition
A.*B: multiply every element of *A* by the corresponding element of *B*

So this command is defined when *A* and *B* are the same size

```
>> A = [1 2 3 4;5 6 7 8]

A =

     1     2     3     4
     5     6     7     8


>> B = [1 1 -1 -1;2 -2 3 -3]

B =

     1     1    -1    -1
     2    -2     3    -3

>> A.*B

ans =

     1     2    -3    -4
    10   -12    21   -24
```

Dot operators

Dot division (divides corresponding elements) and Dot powers:

```
>> A = [1 2 3 4;5 6 7 8]

A =

    1    2    3    4
    5    6    7    8

>> B = [1 1 -1 -1;2 -2 3 -3]

B =

    1    1   -1   -1
    2   -2    3   -3
```

```
>> A./B

ans =

   1.0000   2.0000  -3.0000  -4.0000
   2.5000  -3.0000   2.3333  -2.6667

>> A.^B

ans =

   1.0000   2.0000   0.3333   0.2500
  25.0000   0.0278 343.0000   0.0020
```

Some other operations:

```
>> 1./A

ans =

   1.0000   0.5000   0.3333   0.2500
   0.2000   0.1667   0.1429   0.1250

>> A.^3

ans =

     1     8    27    64
   125   216   343   512
```

If you enter these commands without a dot you'll get an error message: try them and see!

Relational operators

Much of Matlab usage is concerned with testing values.

There are six “relational operators”:

>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal
~=	Not equal

Relational operators

```
>> A = [1 2 3 4;5 6 7 8]

A =

    1    2    3    4
    5    6    7    8

>> B = [1 1 -1 -1;2 -2 3 -3]

B =

    1    1   -1   -1
    2   -2    3   -3
```

```
>> A>B

ans =

  2×4 logical array

   0   1   1   1
   1   1   1   1

>> A<=B+4

ans =

  2×4 logical array

   1   1   1   0
   1   0   1   0

>> A==abs(B)+3

ans =

  2×4 logical array

   0   0   0   1
   1   0   0   0
```

If you type in those at this point, you'll find that *ans* is a matrix of type "Logical".

In Matlab, a *logical* matrix consists of two values: true and false, which are printed as 1 and 0 respectively.

Logical operators

Results of testing can be joined together using “logical operators”:

x & y	x and y: true if both x and y are true, and false if at least one of x and y are false.
x || y	x or y: true if either or both of x and y are true, and false only when both of x and y are false.
xor(x,y)	exclusive or of x and y: true if one of x and y is true and the other false, and false if both x and y are true, or both are false.
~x	not x: true if x is false, and false if x is true.

Here’s how we might use these operators to implement the function:

$$f(x, y) = \begin{cases} x + y, & \text{if } x \geq 0 \text{ or } y \geq 0 \\ x^2 + y^2, & \text{if } x < 0 \text{ and } y < 0 \\ 0, & \text{other} \end{cases}$$

```
function out = f(x,y)

if x>=0 || y>=0
    out = x+y;

else if x<0 & y<0
        out = x^2+y^2;

    else
        out = 0;
    end

end
end
```

The commands *any*, *all* and *find*

Results of testing can be joined together using “logical operators”:

any(test(v))	Returns 1 if *any* of the elements of the vector v are true for the test
all(test(v))	Returns 1 if *all* of the elements of the vector v are true for the test
find(test(v))	Returns the list of indices of v for which the test is true.

```
>> A = 1:8

A =

    1    2    3    4    5    6    7    8
```

```
>> any(A==5)

ans =

  logical

   1

>> any(A>5)

ans =

  logical

   1

>> any(A>9)

ans =

  logical

   0
```

```
>> all(A>3)

ans =

  logical

   0

>> all(A<9)

ans =

  logical

   1
```

VICTORIA UNIVERSITY
MELBOURNE AUSTRALIA

The commands *any, all* and *find*

To apply these operators to an array, you first need to make the array into one big vector.

The colon in brackets takes all the elements of A, and lists them as one long column vector.

```
>> A=magic(4)

A =

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

```
>> any(A(:)>10)

ans =

  logical

   1

>> all(A(:)<20)

ans =

  logical

   1
```

The commands *any*, *all* and *find*

The results of find will be given as single numbers, indicating the position using the single number indexing system.

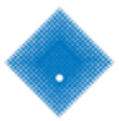
```
>> A=magic(4)

A =

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

```
>> find(A>10)

ans =

     1
     6
     8
    12
    13
    15

>> find(A>4 & A<8)

ans =

     2
     7
    11
```

```
>> X = find(A > 10)

X =

     1
     6
     8
    12
    13
    15

>> A(X) = A(X)*100

A =

        1600           2           3        1300
           5        1100          10           8
           9           7           6        1200
           4        1400        1500           1
```

Find can be used to change all values of an array satisfying some criterion.

i.e. to multiply all values of A which are greater than 10, by 100



VICTORIA UNIVERSITY
MELBOURNE AUSTRALIA