


MATLAB Course
For Engineers
lecture 2


Additional Array Functions

max(A)

Returns the algebraically largest element in A; if A is a vector. Returns a row vector containing the largest elements in each column if A is a matrix.

[x,k]=max(A)

Similar to max(A) but stores the maximum values in the row vector x and their indices in the row vector k.

min(A)	**Like max but returns minimum values.**
[x,k]=min(A)	**Similar to max(A) but stores the minimum values in the row vector x and their indices in the row vector k.**
sum (A)	**Sums the elements in each column of the array A and returns a row vector containing the sums.**

find (x)	**Computes an array containing the indices of the nonzero elements of the arrays x.**
[u,v,w]=find (A)	**Computes the arrays u and v, containing the row and column indices of the nonzero elements of the matrix A, and the array w, containing the values of the *nonzero* elements. The array w may be omitted.**
Sort (A)	**Sorts each column of the array A in ascending order and returns an array the same size as A.**

Mathematical operations with arrays

Array Addition and Subtraction

Array addition; For example:

$$\begin{bmatrix} 6 & -2 \\ 10 & 3 \end{bmatrix} + \begin{bmatrix} 9 & 8 \\ -12 & 14 \end{bmatrix} = \begin{bmatrix} 15 & 6 \\ -2 & 17 \end{bmatrix}$$

Array subtraction is performed in a similar way.

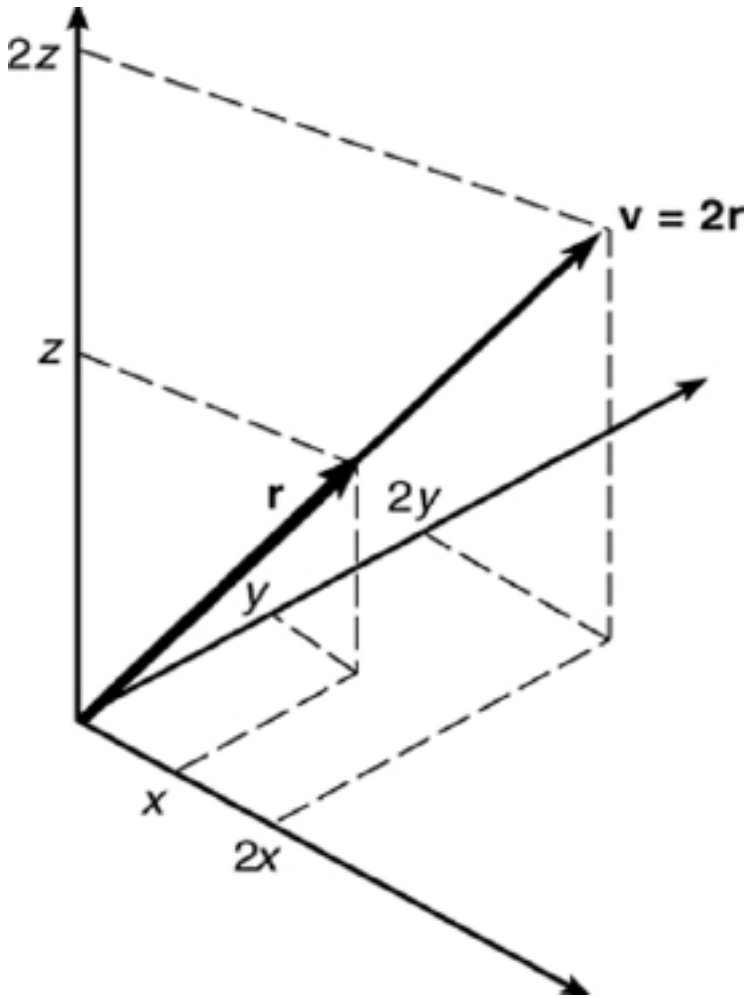
The addition shown is performed in Matlab as following:

```
>>A = [6,-2;10,3];
>>B = [9,8;-12,14];
>>A+B
ans =
        15     6
        -2     17
```

Geometric interpretation of scalar multiplication of a vector

If **r** =[x, y, z], then

v = 2**r**
= 2[x, y, z]
= [2x, 2y, 2z].



Multiplying a matrix A by a scalar *w* produces a matrix whose elements are the elements of A multiplied by *w*. For example:

$$3\begin{bmatrix} 2 & 9 \\ 5 & -7 \end{bmatrix} = \begin{bmatrix} 6 & 27 \\ 15 & -21 \end{bmatrix}$$

This multiplication is performed in Matlab as follows:

```
>>A = [2, 9 ; 5, -7];
>>3*A
ans =
        6      27
       15     -21
```

Multiplication of an array by a scalar is easily defined and easily carried out.

However, multiplication of two *arrays* is not so straight forward.

MATLAB uses two definitions of multiplication:

***(1) array* multiplication (also called *element-by-element* multiplication)**

***(2) matrix* multiplication.**

Division and exponentiation must also be carefully defined when you are dealing with operations between two arrays.

MATLAB has two forms of arithmetic operations on arrays. Next we introduce one form, called *array* operations, which are also called *element-by-element* operations. Then we will introduce *matrix* operations. Each form has its own applications.

Element-by-element (.) operations:

Symbol	Operation	Form	Examples
+	Scalar-array addition	A + b	[6,3]+2=[8,5]
-	Scalar-array subtraction	A – b	[8,3]-5=[3,-2]
+	Array addition	A + B	[6,5]+[4,8]=[10,13]
-	Array subtraction	A – B	[6,5]-[4,8]=[2,-3]
.*	Array multiplication	A.*B	[3,5].*[4,8]=[12,40]
./	Array right division	A./B	[2,5]./[4,8]=[2/4,5/8]
.\	Array left division	A.\B	[2,5].\[4,8]=[2\4,5\8]
.^	Array exponentiation	A.^B	[3,5].^2=[3^2,5^2]
			2.^[3,5]=[2^3,2^5]
			[3,5].^[2,4]=[3^2,5^4]

***Array* or *Element-by-element* multiplication (.*)**

is defined only for arrays having the same size. The definition of the product x.*y, where x and y each have *n* elements, is

x.*y = [x(1)y(1), x(2)y(2), ... , x(n)y(n)]

If x and y are row vectors. For example, if

x = [2, 4, – 5], y = [– 7, 3, – 8];

then >>z = x.*y gives:

```
ans =
        –14,  12,  40
```

z = [2(– 7), 4 (3), –5(–8)]

If x and y are column vectors, the result of x.*y is a column vector. For example:

z = (x').*(y') gives

$$z = \begin{bmatrix} 2(-7) \\ 4(3) \\ -5(-8) \end{bmatrix} = \begin{bmatrix} -14 \\ 12 \\ 40 \end{bmatrix}$$

Note **that x' is a column vector with size 3 × 1 and thus does not have the same size as y, whose size is 1 × 3.**

Thus for the vectors x and y the operations x'.*y and y.*x' are not defined in MATLAB and will generate an error message.

The array operations are performed between the elements in corresponding locations in the arrays. For example, the array multiplication operation A.*B results in a matrix C that has the same size as A and B and has the elements $c_{ij} = a_{ij} b_{ij}$. For example, if

$$\mathbf{A} = \begin{bmatrix} 11 & 5 \\ -9 & 4 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} -7 & 8 \\ 6 & 2 \end{bmatrix}$$

then C = A.*B gives this result:

$$\mathbf{C} = \begin{bmatrix} 11(-7) & 5(8) \\ -9(6) & 4(2) \end{bmatrix} = \begin{bmatrix} -77 & 40 \\ -54 & 8 \end{bmatrix}$$

The built-in MATLAB functions such as sqrt(x) and exp(x) automatically operate on array arguments to produce an array result the same size as the array argument x.

Thus these functions are said to be *vectorized* functions.

For example, in the following session the result y has the same size as the argument x.

```
>>x = [4, 16, 25];
>>y = sqrt(x)
y =
   2  4  5
```

However, when multiplying or dividing these functions, or when raising them to a power, you must use element-by-element (.) operations if the arguments are arrays.

For example, to compute

$z = (e^y \sin x) \cos^2 x$**, you must type**

```
z = exp(y).*sin(x).*(cos(x)).^2.
```

You will get an error message if the size of x is not the same as the size of y. The result z will have the same size as x and y.

Array Division (./)

The definition of array division is similar to the definition of array multiplication except that the elements of one array are divided by the elements of the other array. Both arrays must have the same size. The symbol for array right division is (./)
For example, if

x = [8, 12, 15] y = [–2, 6, 5]

then >>z = x./y

Gives

z = [8/(–2), 12/6, 15/5] = [–4, 2, 3]

Also, if

$$\mathbf{A}=\begin{bmatrix}24 & 20\\ -9 & 4\end{bmatrix} \qquad \mathbf{B}=\begin{bmatrix}-4 & 5\\ 3 & 2\end{bmatrix}$$

Then

```
>>C=A./B
```

Gives

$$\mathbf{C}=\begin{bmatrix}24/(-4) & 20/5\\ -9/3 & 4/2\end{bmatrix}=\begin{bmatrix}-6 & 4\\ -3 & 2\end{bmatrix}$$

Array Exponentiation (.^)

MATLAB enables us not only to raise arrays to powers but also to raise scalars and arrays to *array* powers.

To perform exponentiation on an element-by-element basis, we must use the .^ symbol.

For example, if x = [3, 5, 8], then typing >>x.^3 produces the array

$$[3^3, 5^3, 8^3] = [27, 125, 512].$$

We can raise a scalar to an array power.
For example, if p = [2, 4, 5], then typing 3.^p produces the array $[3^2, 3^4, 3^5] = [9, 81, 243]$.

Remember that (.^) is a *single* symbol. The dot in 3.^p is *not* a *decimal point* associated with the number 3. The following operations, with the value of p given here, are equivalent and give the correct answer:

```
>>3.^p
>>3.0.^p
>>3..^p
>>(3).^p
>>3.^[2,4,5]
```

Matrix-Matrix Multiplication (*)

In the product of two matrices AB, the number of *columns* in A must equal the number of *rows* in B. The row-column multiplications form column vectors, and these column vectors form the matrix result. The product AB has the same number of *rows* as A and the same number of *columns* as B. For example,

$$\begin{bmatrix} 6 & -2 \\ 10 & 3 \\ 4 & 7 \end{bmatrix} \begin{bmatrix} 9 & 8 \\ -5 & 12 \end{bmatrix} = \begin{bmatrix} (6)(9)+(-2)(-5) & (6)(8)+(-2)(12) \\ (10)(9)+(3)(-5) & (10)(8)+(3)(12) \\ (4)(9)+(7)(-5) & (4)(8)+(7)(12) \end{bmatrix}$$

$$= \begin{bmatrix} 64 & 24 \\ 75 & 116 \\ 1 & 116 \end{bmatrix}$$

Use the operator (*) to perform matrix multiplication in MATLAB. The following MATLAB session shows how to perform the matrix multiplication.

```
>>A = [6,-2;10,3;4,7];
>>B = [9,8;-5,12];
>>A*B
ans =
    64   24
    75   116
    1    116
```

Matrix multiplication does not have the commutative property; that is, in general, $AB \neq BA$.
A simple example will demonstrate this fact:

$$\mathbf{AB} = \begin{bmatrix} 6 & -2 \\ 10 & 3 \end{bmatrix} \begin{bmatrix} 9 & 8 \\ -12 & 14 \end{bmatrix} = \begin{bmatrix} 78 & 20 \\ 54 & 122 \end{bmatrix}$$

Whereas

$$\mathbf{BA} = \begin{bmatrix} 9 & 8 \\ -12 & 14 \end{bmatrix} \begin{bmatrix} 6 & -2 \\ 10 & 3 \end{bmatrix} = \begin{bmatrix} 134 & 6 \\ 68 & 65 \end{bmatrix}$$

Reversing the order of matrix multiplication is a common and easily made mistake!!!!.

Special Matrices

Two exceptions to the noncommutative property are the *null* or *zero* matrix, denoted by 0 and the *identity*, or *unity*, matrix, denoted by I.

The null matrix contains all zeros and is not the same as the *empty* matrix [ ], which has no elements.

These matrices have the following properties:

$$0A = A0 = 0$$

$$IA = AI = A$$

Polynomial Operations Using Arrays

Polynomial Multiplication and Division

The function convolution conv(a,b) computes the product of the two polynomials described by the coefficient arrays a and b. The two polynomials need not be the same degree. The result is the coefficient array of the product polynomial.

The function deconvolve [q,r] = deconv(num,den) computes the result of dividing a numerator polynomial, whose coefficient array is num, by a denominator polynomial represented by the coefficient array den. The quotient polynomial is given by the coefficient array q, and the remainder polynomial is given by the coefficient array r.

Polynomial Multiplication and Division

Examples:

```
>>a = [9,-5,3,7];
>>b = [6,-1,2];
>>product = conv(a,b)
product =
   54   -39   41   29   -1   14
>>[quotient, remainder] = deconv(a,b)
quotient =
   1.5   -0.5833
remainder =
   0   0   -0.5833   8.1667
```

Polynomial Roots

The function roots(a) computes the roots of a polynomial specified by the coefficient array a. The result is a *column vector* that contains the polynomial's roots.

For example:

```
>>r = roots([2, 14, 20])
r =
   -2
   -5
```

Polynomial Coefficients

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

The result is a *row* vector that contains the polynomial's coefficients arranged in descending order of power.

For example:

```
>>c = poly([-2, -7])
c =
   1    9    14
```

The function polyval(a,x) evaluates a polynomial at specified values of its independent variable x, which can be a matrix or a vector. The polynomial's coefficients of descending powers are stored in the array a. The result is the same size as x.

Example:

$f(x) = 9x^3 - 5x^2 + 3x + 7$ for $-2 \le x \le 5$, you type

```
>>a = [9,-5,3,7];
>>x = [-2:1:5];
>>f = polyval(a,x);
```

Polynomial Derivatives

polyder(p)	Returns a vector b containing the coefficients of the derivative of the polynomial represented by the vector p.
polyder(p1,p2)	Returns a vector b containing the coefficients of the polynomial that is the derivative of the product of the polynomials represented by p1 and p2.
[num, den] = polyder(p2,p1)	Returns the vectors num and den containing the coefficients of the numerator and denominator polynomials of the derivative of the quotient p2/p1, where p1 and p2 are polynomials.