Introduction to MATLAB: Part II

Vector and Matrices Operations with MATLAB

In the previous introduction section, how to input vectors and matrices has been explained briefly. In this section, vector and matrix operations will be discussed within the limited scope of physic laboratory experiments.

Vectors Operation

In MATLAB a vector is a matrix with either one row or one column. In two dimensional system, a vector is usually represented by 1 × 2 matrix. For example, a vector, B in Figure 1 is 6***i*** + 3***j***, where ***i*** and ***j*** are unit vectors in the positive direction for x and y axes, respectively in the Cartesian coordinate system. For three dimensional system, the unit vectors in x, y, and z axes are labeled ***i***, ***j***, and ***k***, respectively.

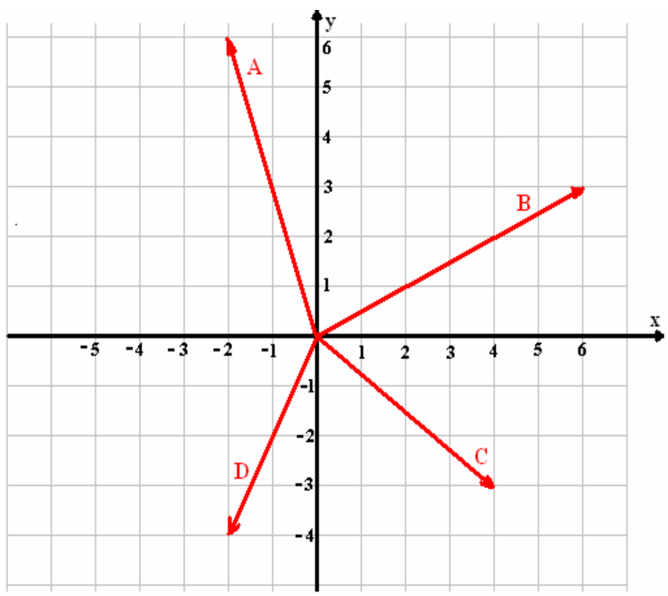

Figure 1. Two dimensional vectors, A, B, C and D plotted in the Cartesian coordinate system

As mentioned earlier in the section of inputting vectors and matrices, MATLAB uses square brackets, [ ] to create a vector. For example, to create the vectors A = -2***i*** + 6***j***, B = 6***i*** + 3***j***, C = 4***i*** – 3***j***, and D = -2***i*** – 4***j*** shown in Figure 10, type in the MATLAB command window.

```
>> A = [-2, 6]
>> B = [6, 3]
>> C = [4, -3]
>> D = [-2, -4]
```

To create a vector **v** = 3*i* + 2*j* + 5*k*, type `v = [3, 2, 5]`.

The basic operations that can be performed with vectors are addition, subtraction, and scalar multiplication.
For example, to add A and B vectors (-2***i*** + 6***j***) + (6***i*** + 3***j***),

```
>> [-2, 6] + [6, 3]
ans =
     4     9
```

To perform the scalar multiplication 2·(3*i* + 2*j* +5*k*), type `2*[3, 2, 5]` or `[3, 2, 5]*2` :

```
>> [3, 2, 5]*2
ans =
     6     4    10
```

To find the magnitude of a vector in MATLAB, you can use a command, **norm**. In mathematics, the magnitude of a vector (xi + yj) is defined as $\sqrt{x^2 + y^2}$. For example, the magnitude of the vector C (= 4*i* – 3*j*) is 5 (=$\sqrt{4^2 + 3^2}$)). In MATLAB,

```
>> norm([4, -3])
ans =
     5
```

or, since you already defined the vector C (C = [4, -3]),

```
>> norm(C)
ans =
     5
```

To find the angle with x axis, use the **`atan2(y, x)`** command, which is the four quadrant arctangent of the real parts of the elements of x and y. Note that the *y* value must be entered before *x*. This command will return an angle between π and -π. To get a value in degrees, multiply the answer by 180/π. For example, to find the angles in degree for the vectors A = -2***i*** + 6***j***, B = 6***i*** + 3***j***, C = 4***i*** – 3***j***, and D = -2***i*** – 4***j***:

```
>> atan2(6, -2)*180/pi
ans =
  108.4349

>> atan2(3, 6)*180/pi
ans =
   26.5651

>> atan2(-3, 4)*180/pi
ans =
  -36.8699

>> atan2(-4, -2)*180/pi
ans =
 -116.5651
```

There are other inverse tangent functions **`atand( )`** and **`atan( ),`** the first giving a degree answer and the second giving a radian answer. For the vector of D = -2***i*** – 4***j***, examples are:

```
>> atand(-4/-2)
ans =
   63.4349

>> atan(-4/-2)
ans =
    1.1071
```

However, they do not return the angle in the proper quadrant. As shown in the above examples, **`atand(-4/-2)`** returns an answer of 63.4 degrees instead of the third quadrant angle of 243.4 (= 360 + (-**116.6**)) degrees.

Converting Rectangular (Cartesian coordinate) to Polar Coordinate

Polar coordinate system in two-dimension is composed of two coordinates, r (radius) and θ (angle). To convert the vector B = 6***i*** + 3***j***) into polar coordinates, find the radius using the norm command `norm([6,3])` and the angle using `atan2` command, `atan2(3, 6)`. The results are:

```
>> r = norm([6, 3])
r =
    6.7082

>> theta = atan2(3,6)*180/pi
theta =
   26.5651
```

Therefore, the polar coordinate for the B vector is r= 6.71 (radius) and θ = 26.57 degree (angle).

Polar to Rectangular Conversions

To convert a vector with magnitude 5 and an angle of -36.8699 degrees into rectangular, see the following example (Notice that the "d" at the end of cosine and sine). The result gives us the vector 4***i*** – 3***j***.

```
> 5*[cosd(-36.8699), sind(-36.8699)]
ans =
    4.0000   -3.0000
```

To convert a vector with magnitude 5 and an angle of -0.6435 radians into rectangular, type

```
>> 5*[cos(-0.6435), sin(-0.6435)]
ans =
    4.0000   -3.0000
```

Dot and Cross Products

The **dot product** (also called scalar product) of the vectors ***a*** and ***b*** is written as ***a · b*** and defined to be ***a · b*** = abcosϕ, where a and b is the magnitudes of the two vectors and ϕ is the angle between the two vectors. When two vectors are in unit-vector notation, their dot product is written as: $\boldsymbol{a} \cdot \boldsymbol{b} = (a_1\boldsymbol{i} + a_2\boldsymbol{j} + a_3\boldsymbol{k})\cdot(b_1\boldsymbol{i} + b_2\boldsymbol{j} + b_3\boldsymbol{k})$ and the result is $\boldsymbol{a} \cdot \boldsymbol{b} = a_1b_1 + a_2b_2 + a_3b_3$.

In MATLAB, you can compute the dot product using **dot** command. For example, to find the dot product of two vectors, A = -2***i*** + 6***j*** and B = 6***i*** + 3***j***,

```
>> dot([-2, 6],[6,3])
ans =
     6

>> clear all
>> A = [-2, 6];
>> B = [6, 3];
>> dot (A, B)
ans =
     6

>>clear all
>> syms x y z
>> v = [x, y, z];
>> r = [2, 3, 5];
>> dot(r, v)
```

```
 ans =
 2*x+3*y+5*z
```

We remark that the MATLAB's symbolic dot product assumes that its arguments may be complex and takes the complex conjugates of the components of its first argument. To see the effect of this, we compute instead:

```
>> dot(v,r)
 ans =
2* conj(x)+3*conj(y)+5*conj(z)
```

Since in this introduction we only want dot products of real-valued vectors, it helps to define as below:

```
>> realdot = @(u, v) u*transpose(v);
>> realdot(v,r)
 ans =
 2*x+3*y+5*z
```

The function of "realdot" works in the same manner as dot command for numerical arguments. For example,

```
>> a = [3,4];
>> b = [1,1];
>> dot(a,b)
ans =
     7

>> realdot(a,b)
ans =
    7
```

The **cross product** (also called vector product) of vectors ***a*** and ***b***, written as $\boldsymbol{a} \times \boldsymbol{b}$, produces a third vector c whose magnitude is c = ab*sin*ϕ where ϕ is the smaller of the two angles between vectors ***a*** and ***b***. In unit-vector notation, the cross product of vectors ***a*** and ***b*** is written as $\boldsymbol{a} \times \boldsymbol{b} = (a_1\boldsymbol{i} + a_2\boldsymbol{j} + a_3\boldsymbol{k}) \times (b_1\boldsymbol{i} + b_2\boldsymbol{j} + b_3\boldsymbol{k})$ and the results is $\boldsymbol{a} \times \boldsymbol{b} = (a_2b_3 - a_3b_2)\boldsymbol{i} + (a_3b_1 - a_1b_3)\boldsymbol{j} + (a_1b_2 - a_2b_1)\boldsymbol{k}$.

In MATLAB, you can compute the cross product using **cross** command. For example, to find the cross product of $(3i + 2j + 5k)$ and $(7i - 2j + 8k)$,

```
>> clear all
>> a = [3, 2, 7];
>> b = [7, -2, 8];
>> cross (a, b)
ans =
    30    25   -20
```

or

```
>> cross ([3, 2, 7], [7, -2, 8])
ans =
    30    25   -20
```

Therefore, the resultant cross product is $30\boldsymbol{i} + 25\boldsymbol{j} - 20\boldsymbol{k}$ vector.

Matrix Operation

The following table shows the basic matrix operations in MATLAB for addition, substraction, multiplying all elements of matrix by a scalar, and dividing all elements of a matrix by a scalar.

Matrix A	Matrix B

`>> A = [2, 4; 6, 8]`
`A =`
`2 4`
`6 8`
	`>> B = [1, 2; 3, 4]`
	`B =`
	`1 2`
	`3 4`

Addition, A + B	Subtraction, A – B

`>> A + B`
`ans =`
`3 6`
`9 12`
	`>> A – B`
	`ans =`
	`1 2`
	`3 4`

Multiplication of all elements of matrix A by 2	Division of all elements of matrix A by 2

`>> Z = 2*A`
`Z =`
`4 8`
`12 16`
	`>> X = A/2`
	`X =`
	`1 2`
	`3 4`

Multiplication	Element-by-element multiplication. Note that this is different from multiplication of two matrices.

`>> A*B`
`ans =`
`14 20`
`30 44`
	`>> A.*B`
	`ans =`
	`2 8`
	`18 32`

Element-by-element power	

`>> A.^B`
`ans =`
`2 16`
`216 4096`	

In addition to the matrix operations mentioned in the above table, there are matrix divisions including left and right matrix division. Here, only left matrix division will be discussed.

The MATLAB command for left matrix division is **`mldivide`** or "\" (back slash). For example, **`X = A \ B`** in MATLAB command means dividing A into B. This is equivalent to **`inv(A)*B`**, where inv(A) is the inversion of matrix A. Basically, the resultant value, X is the solution to **`A*X = B`**, which is expressed as inv(A)*A*X = inv(A)*B. The product of inv(A) and A gives you identity matrix. For example,

```
>> A = [1, 2; 3, 4]
A =
     1     2
     3     4
```

```
>> inv(A)
ans =

   -2.0000    1.0000
    1.5000   -0.5000

>> inv(A)*A
ans =
    1.0000         0
    0.0000    1.0000
```

Let’s try to solve the following linear equations using left matrix division. First thing is to create matrix form.

$$3x + 4y + 5z = 2$$
$$2x - 3y + 7z = -1$$
$$x - 6y + z = 3$$

$$\begin{pmatrix} 3 & 4 & 5 \\ 2 & -3 & 7 \\ 1 & -6 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} := \begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix}$$

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

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

>> X = mldivide(A,B)
X =
    2.6196
   -0.2283
   -0.9891

>> format rat, X
X =
     241/92
     -21/92
     -91/92
```

Therefore the answer is x = 241/92, y = -21/92, and z = -91/92.