DAY II-TUTORIALS

MORE ON MATRIX

Finding the size of a matrix

The following syntax finds the size of a matrix:

```
 d = size(X)
[m,n] = size(X)
 m = size(X,dim)
```

Where m=number of rows and n=number of columns.

e.g

```
q=[2 3 4 1;6 3 8 9;1 5 3 6]

q =

     2     3     4     1

     6     3     8     9

     1     5     3     6

>> size(q)

ans =

     3     4

>> [m n]=size(q)

m =

     3

n =

     4

>> m=size(q,1)

m =

     3

>> n=size(q,2)
```

```
n  =4
```

Matrix Concatenation

```
>>a=[1 2 3], b=[7 9 4], c=[1 3 5 8]
```

then

```
>>d=[a b c]

D =

1     2     3     7     9     4     1     3     5

>> d=[a;b;c]

d =

     1     2     3

     7     9     4

     1     3     5
```

- B = [1 2 3;4 5 6;7 8 9]

```
B =

     1     2     3

     4     5     6

     7     8     9
```

- >> W=B([1 3],:)   % selects the first and third rows, all the columns.

```
W=

     1       2     3

      7     8     9
```

Deleting Rows or Columns

To get rid of arrow or column set it equal to the empty matrix [].

```
a=[1 2 3;4 5 6;7 8 9]
```

```
a =

    1     2     3

    4     5     6

    7     8     9


>> a(:,2)=[]                    % deletes the all the rows the second column.

a =

    1     3

    4     6

    7     9
```

Try doing more

Using MATLAB built-in functions to Concatenation matrices

The functions to be used are vertcat and horzcat.

1. vertcat

Concatenate arrays vertically

Syntax

```
C = vertcat(A1, A2, ...)
```

Description

`C = vertcat (A1, A2 ...)` vertically concatenates matrices `A1`, `A2`, and so on. All matrices in the argument list must have the same number of columns.

Examples

Create a 5-by-3 matrix, `A`, and a 3-by-3 matrix, `B`. Then vertically concatenate `A` and `B`.

```
A = magic (5);              % Create 5-by-3 matrix, A
A(:, 4:5) = []
```

```
A =

    17    24     1
    23     5     7
     4     6    13
    10    12    19
    11    18    25


B = magic(3)*100         % Create 3-by-3 matrix, B

B =

   800   100   600
   300   500   700
   400   900   200


C = vertcat(A,B)         % Vertically concatenate A and B

C =

    17    24     1
    23     5     7
     4     6    13
    10    12    19
    11    18    25
   800   100   600
   300   500   700
   400   900   200
```

2. **horzcat**

Concatenate arrays horizontally

Syntax

```
C = horzcat(A1, A2, ...)
```

Description

`C = horzcat(A1, A2, ...)` horizontally concatenates matrices `A1`, `A2`, and so on. All matrices in the argument list must have the same number of rows.

Examples

Create a 3-by-5 matrix, `A`, and a 3-by-3 matrix, `B`. Then horizontally concatenate `A` and `B`.

```
A = magic(5);            % Create 3-by-5 matrix, A
A(4:5,:) = []

A =

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22


B = magic(3)*100       % Create 3-by-3 matrix, B

B =

   800   100   600
   300   500   700
   400   900   200


C = horzcat(A, B)      % Horizontally concatenate A and B

C =

    17    24     1     8    15   800   100   600
    23     5     7    14    16   300   500   700
     4     6    13    20    22   400   900   200
```

EXERCISE

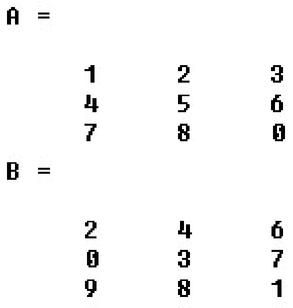
Create

1. 5-by-4 matrix, X
2. 7-by-6 matrix, Y
3. Delete all the roles, the second column of matrix X
4. Delete all the columns, the last row of matrix Y
5. Find the size MATRIX X and Y
6. concatenate X and Y. Vertically and Horizontally using vertcat, horzcat and the normal way.

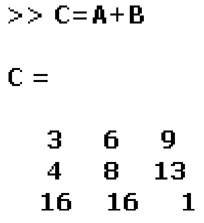
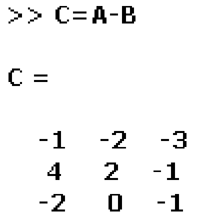
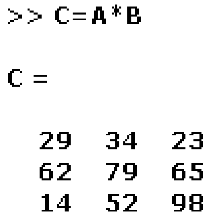
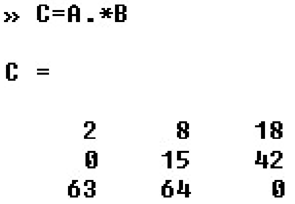
MATRIX ARITHMETIC

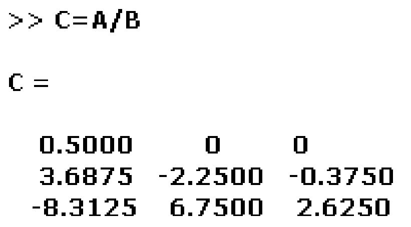
MATLAB treats arithmetic operations on arrays in an element-by-element manner. This operation is accomplished by including a dot or a period before the arithmetic operator such as multiplication, division, etc. The table below gives a list of such operators with examples of how these operators are being used.

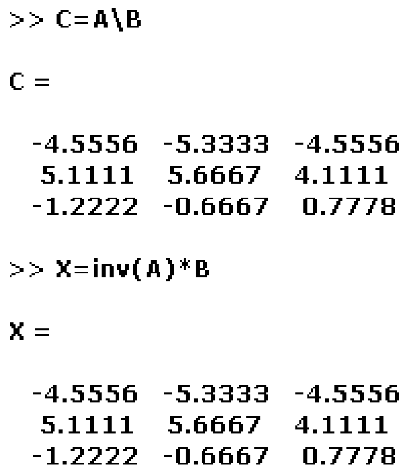
For examples in the table below, the following matrices are used:

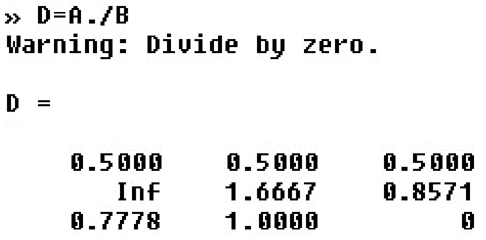


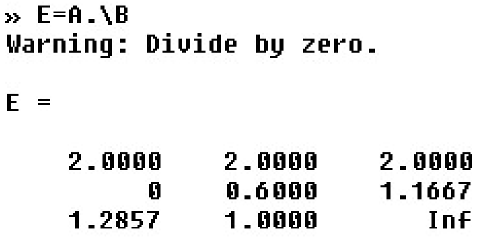
```
A =

      1      2      3
      4      5      6
      7      8      0
B =

      2      4      6
      0      3      7
      9      8      1
```

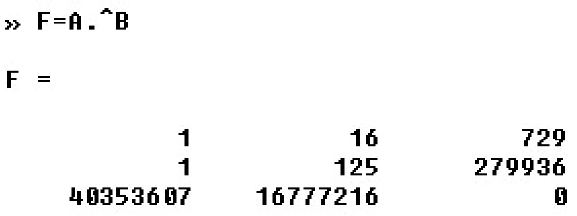
Operation	Description	Example (MATLAB actual inputs and outputs)
+	Addition	 >> C=A+B C = 3 6 9 4 8 13 16 16 1
-	Substraction	 >> C=A-B C = -1 -2 -3 4 2 -1 -2 0 -1
*	Multiplication	 >> C=A*B C = 29 34 23 62 79 65 14 52 98
.*	Element-by-element multiplication. Note that this is different from multiplication of two matrices.	 » C=A.*B C = 2 8 18 0 15 42 63 64 0 Note that this is not the same as **C = A*B**

/	Right matrix division.
	Dividing matrix **B** into matrix **A**
		
		`>> C=A/B`
		`C =`
		`   0.5000        0        0`
		`   3.6875  -2.2500  -0.3750`
		`  -8.3125   6.7500   2.6250`

\	Left matrix division.
	Dividing **A** into **B**. This is equivalent to **inv(A)*B**. Note that **X** = **C** is the solution to **A*X=B**
		
		`>> C=A\B`
		`C =`
		`  -4.5556  -5.3333  -4.5556`
		`   5.1111   5.6667   4.1111`
		`  -1.2222  -0.6667   0.7778`
		`>> X=inv(A)*B`
		`X =`
		`  -4.5556  -5.3333  -4.5556`
		`   5.1111   5.6667   4.1111`
		`  -1.2222  -0.6667   0.7778`

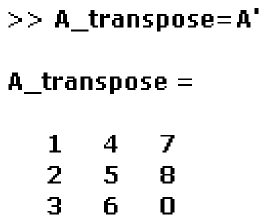
./	Element-by-element division note that **D**(2,1) is undefined due to the zero at **B**(2,1)	
		`» D=A./B`
		`Warning: Divide by zero.`
		`D =`
		`    0.5000    0.5000    0.5000`
		`       Inf    1.6667    0.8571`
		`    0.7778    1.0000         0`

.\	Element-by-element left division.Note that left division in this particular example means elements of B divided by the corresponding elements of A.	
		`» E=A.\B`
		`Warning: Divide by zero.`
		`E =`
		`    2.0000    2.0000    2.0000`
		`         0    0.6000    1.1667`
		`    1.2857    1.0000       Inf`

.^	Element-by-element power	
		`» F=A.^B`
		`F =`
		`           1          16         729`
		`           1         125      279936`
		`    40353607    16777216           0`

Transposing a matrix in MATLAB involves a simple prime notation (') after the defined matrix or the use of the `transpose` function (transp).as shown below:

Example:



```
>> A_transpose=A'

A_transpose =

     1     4     7
     2     5     8
     3     6     0
```

Diagonal of a matrix

The command diag has two uses.

- **The first use is to extract a diagonal of a matrix.**

syntax

`v = diag(X)` returns the main diagonal of `X`.

Example

```
A = [1 2 3; 4 5 6; 7 8 9]
```

D=diag(A)

>> **D=diag(A)**

D =

1

5

9

- **The second use is to create diagonal matrices.**

  Example

  d=diag([2;1;3;4])

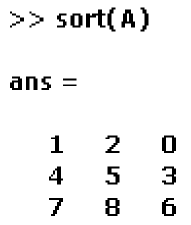
  d =

2 0 0 0
0 1 0 0
0 0 3 0
0 0 0 4

Sorting columns and rows follow the syntax: ***B=sort(****A*,*dim****)***, where *dim* is the dimension of the matrix with the value 1 for column; 2 for row. Matrix A is the variable specified by the user.

Example:

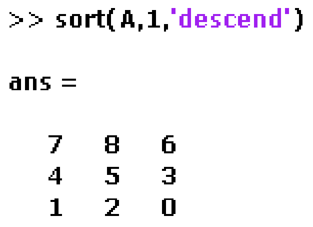
Sorting columns:



```
>> sort(A)

ans =

     1     2     0
     4     5     3
     7     8     6
```

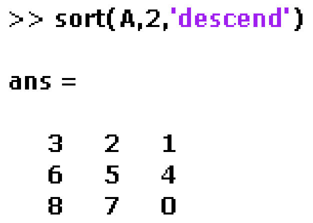
Note that without *dim* being specified, the default value is 1. The default setting is ascending order. The variable name of the sorted matrix can be omitted if no needed.

Sorting column in descending order:



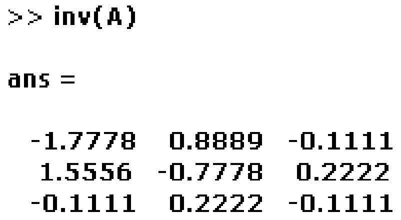
```
>> sort(A,1,'descend')

ans =

     7     8     6
     4     5     3
     1     2     0
```

Sorting row in descending order



```
>> sort(A,2,'descend')

ans =

     3     2     1
     6     5     4
     8     7     0
```

The inverse of matrix **A** can be obtained with the command:



```
>> inv(A)

ans =

  -1.7778   0.8889  -0.1111
   1.5556  -0.7778   0.2222
  -0.1111   0.2222  -0.1111
```

Determinat of a matrix

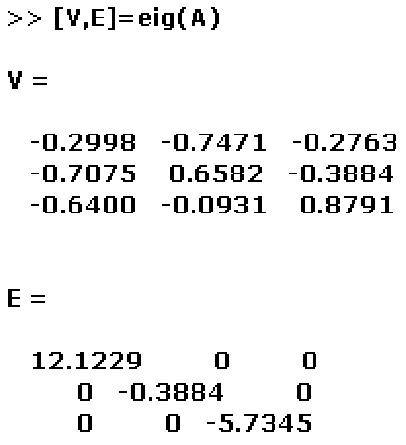
Syntax

```
d = det(X)
```

Examples

```
 A = [1 2 3; 4 5 6; 7 8 9]

d = det(A)

d=0
```

Eigenvalues and Eigenvectors can easily be obtained with the command **[V,E]=eig(***matrix name***)**:



```
>> [V,E]=eig(A)

V =

  -0.2998  -0.7471  -0.2763
  -0.7075   0.6582  -0.3884
  -0.6400  -0.0931   0.8791


E =

  12.1229        0        0
        0  -0.3884        0
        0        0  -5.7345
```

Special Matrices in Matlab

1. eye

Identity matrix

Syntax

```
Y = eye(n)
Y = eye(m,n)
eye([m n])
```

Example:

- eye (3)

  ans =

  ```
     1    0    0

     0    1    0

     0    0    1
  ```

  Try eye(3,4)

2. zeros

Create array of all zeros

Syntax

```
B = zeros(n)
B = zeros(m,n)
B = zeros([m n])
```

Example

```
s=zeros(3)          % 3x3 matrix

s =

   0    0    0

   0    0    0

   0    0    0
```

3. ones

Create array of all ones

Syntax

```
Y = ones(n)
Y = ones(m,n)
Y = ones([m n])
```

Example

- ones(2,4)

  ans =

  ```
     1    1    1    1

     1    1    1    1
  ```

- **Others are:**
  - **rand(3)**
  - **rand(2,3)**

    magic(3)
  - **randn(2,3)**

Syntax

```
B = reshape(A,m,n)
```

`B = reshape(A,m,n)` returns the `m`-by-`n` matrix `B` whose elements are taken column-wise from `A`. An error results if `A` does not have `m*n` elements.

Examples

Reshape a `3`-by-`4` matrix into a `2`-by-`6` matrix.

```
A =
    1     4     7    10
    2     5     8    11
```

```
    3    6    9    12

B = reshape(A,2,6)

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