Math-c Documentation

Linear, vector and matrix functions

Make range

To make a vector from range A to B is possible with the range notation A:inc:B where inc=1 by default in A:B notation, including A and B in the range.

To exclude A or B can be set by the character $ in A side or B side.

example:

```
>>> a = 1:7

a = [1 2 3 4 5 6 7]

>>> a = 1:2:7     //increment 2

a = [1  3  5  7]

>>> a = 1:$7

a = [1 2 3 4 5 6]

>>> a = 1$:$7

a = [2 3 4 5 6]
```

Using index and delete rows and columns of matrix

To access matrix elements the index notation can be used, the first element, is the row and the second is the column starting the index in 0. to select all elements in column or row use ":". Note in index is not used the character "$" like in range.

To delete rows just assign the row or column to [];

examples:

```
//notice a is a nested matrix containing in a vector
```

```
// in element (0,2)
 a  = [6  5  [-1  -3  [first  second]]  0  12 ;
      9  17  10  1  20 ;
      12  2  18  11  2 ;
      0  13  16  4  8 ;
      15  16  5  4  13]
>>>  a(0,2)(2)
  ans = [first  second]
>>> a(0,2)(2)(0)
  ans = second
>>> a(7)        //select the element (2,1)
  ans = 2
>>> a(0,2)(2)=88        //you can assign using index
  ans = [88]
>>> a
 a  = [6  5  [-1  -3  88]  0  12 ;
      9  17  10  1  20 ;
      12  2  18  11 2 ;
      0   13  16  4  8 ;
      15  16  5  4  13]
>>> a(1,:)     //all columns of row 1
   ans = [9  17  10  1  20]
>>> a(1,1:)   //from column 1 to end
    ans = [17  10  1  20]
>>> a(1,0:2:)      //from column 0 increment 2 to end
     ans = [9  10  20]
>>> a(1:2,:) = []    //delete rows 1 and 2
    a = [6  5  [-1  -3  [first  second]]  0  12 ;
        0  13  16  4  8 ;
```

```
15  16  5  4  13]
```

Multiple assign

You can assign multiples values in one line.

examples:

```
>>> [a b c] = [3 4 [7 8]]
  ans = [3 4 [7 8]]
>>> a
  ans = 3
>>> b
  ans = 4
>>> c
  ans = [7 8]
```

Merge operators

A <-> B

Join two matrices horizontally, must have the same number of rows.

examples:

```
>>> M = [7 5 4;2 4 5]
  M = [7 5 4 ;
       2 4 5]
>>> D = [8;9]
  D = [8 ;
       9]
>>> M <-> D
  ans = [7 5 4 8 ;
         2 4 5 9]
>>> a = 7;        //scalar and vector
```

```
>>> g = [1 9 4];
>>> a <-> g
  ans = [7 1 9 4];
```

A <|> B

Join two matrices vertically, must have the same number of cols. B will be added to bottom A

examples:

```
>>> M = [7 5 4;2 4 5]
  M = [7 5 4 ;
       2 4 5]
>>> D = [2 9 6]
>>> M <-> D
  ans = [7 5 4 ;
         2 4 5 ;
         2 9 6]
>>> a = 7;
>>> g = [5; 2; 9];          //scalar and vector
>>> a <|> g
  ans = [7 ;
         5
         2;
         9]
```

Matrix functions

y = conv(a,b)

convolution of vector a and b

a -> vector

b -> vector

returns

y -> convolution of a and b; example:

>>> u = [3 3 0 3 14];

>>> v = [15 5 5 13 14];

>>> conv(u,v)

ans = [45 60 30 99 306 127 109 224 196]

y = cross(a,b)

cross product between a and vectors

a -> vector of size 3 (any orientation)

b -> vector of size 3 (any orientation)

returns

y -> cross product

y = det(x)

determinant of square matrix x

x -> square matrix

returns

y -> determinant of x

y = dot(a,b)

dot product

a -> value must be 1xM or Mx1 size

b -> value must be 1xM or Mx1 size

returns

y -> dot product

1 = eye()

y = eye(a)

y = eye(a,b)

Identity matrix

a -> integer value

b -> integer value (default b=a)

returns

y -> identity matrix of size a,b; example:

>>> eye()

ans = 1

>>> eye(3)

ans = [1 0 0 ;

0 1 0 ;

0 0 1]

>>> eye(3,4)

ans = [1 0 0 0 ;

0 1 0 0 ;

0 0 1 0]

y = filteri(num,den,sig)

y = filteri(num,den,sig,yi)

y = filteri(num,den,sig,yi,xi)

filter signal with the transfer function.

num -> vector, numerator

den -> vector, denominator

sig -> vector, signal input

yi -> vector, output initial conditions (yi=0)

xi -> vector, input initial conditions (default xi=0)

returns

y -> filter signal with the transfer function with numerator num and denominator den.

example:

for step input with a transfer function of:

the filteri function will be

>>> y=filteri([0 0.5 -0.11 -.3],[1 -1.9 1.6 -0.6 0.05],zeros(1,30)+1);

and if the initial conditions are y(-1)=1.5,y(-2)=2; in x(-1)=0 x(-2)=1.2

>>> y=filteri([0 0.5 -0.11 -.3],[1 -1.9 1.6 -0.6 0.05],zeros(1,30)+1,[1.5 2],[0 1.2]);

y = fft(x)

Fast fourier transform

x -> vector size of power 2, and some other sizes.

returns

y -> return the vector of the Discrete Fourier Transform or return 0, if the size of x is not implemented to apply the algorithm.

y = ifft(x)

Inverse Fast fourier transform

x -> vector size of power 2, and some other sizes.

returns

y -> return the vector of the Inverse Discrete Fourier Transform or return 0, if the size of x is not implemented to apply the algorithm.

y = inv(x)

inverse matrix

x -> square matrix

returns

y -> inverse matrix

a = matjoin(a)

y = matjoin(a,b)

join two matrix elements (useful with images)

a -> matrix value

b -> matrix value (same size of a)

returns

y -> a new matrix with vectors containing the elements of each matrix, example:

a = [7 9 8 0 ;

2 0 9 4 ;

9 1 1 0]

b = [0 6 1 9 ;

9 10 6 5 ;

7 3 10 2]

>>> y = matjoin(a,b)

y = [[7 0] [9 6] [8 1] [0 9] ;

[2 9] [0 10] [9 6] [4 5] ;

[9 7] [1 3] [1 10] [0 2]]

y = matsplit(a)

split in matrices the sub elements of a (useful with images)

a -> matrix value with sub elements(must have the same number of sub-elements in the matrix), example a = [[5 6 7] [8 5 5] [4 3 5]]

returns

y -> a nested matrix a containing an array of matrices containing the elements, example with two sub-elements:

y = [[7 0] [9 6] [8 1] [0 9] ;

[2 9] [0 10] [9 6] [4 5] ;

[9 7] [1 3] [1 10] [0 2]]

>>> [a b] = matsplit(y);

>>> a

a = [7 9 8 0 ;

2 0 9 4 ;

9 1 1 0]

>>> b

b = [0 6 1 9 ;

9 10 6 5 ;

7 3 10 2]

y = max(x)

Maximum value of x

x -> value, vector or matrix

returns

y -> the maximum value of x and the index, if x is a matrix get the maximum of each column and its row position; example.

a = [14 4 15 2 ;

10 10 0 14 ;

0 1 11 3 ;

6 1 2 3]

>>> max(a)

ans = [[14 10 15 14] [0 1 0 1]]

y = mean(x)

Average value of x

x -> value, vector or matrix

returns

y -> the average value of x, if x is a matrix get the average of each column and returns as a vector; example.

a = [14 4 15 2 ;

10 10 0 14 ;

0 1 11 3 ;

6 1 2 3]

>>> mean(a)

ans = [7.5 4 7 5.5]

y = min(x)

Minimum value of x

x -> value, vector or matrix

returns

y -> the minimum value of x and the index, if x is a matrix get the minimum of each column and its row position; example.

```
a = [14  4  15  2 ;
   10  10  0  14 ;
    0  1  11  3 ;
    6  1  2  3]
>>> min(a)
ans = [[0  1  0  2]  [2  2  1  0]]
```

y = size(x)

Size of x

x -> value, scalar, vector or matrix

returns

y -> a 1,2 vector containing in index 0 the number of rows and index 1 the number of columns.

y = sort(a)

y = sort(a,b)

Sort a

a -> value, vector or matrix

b ->string, must be "ascend" or "descend"

returns

y -> the sorted matrix, sorting each column, if a is a vector, sort the vector in any orientation.

y = sort(a)

y = sort(a,b)

Sort a

a -> value, vector or matrix

b -> string, must be "ascend" or "descend" ("ascend" is the default value)

returns

y -> the sorted matrix, sorting each column, if a is a vector, sort the vector in any orientation.

y = sum(x)

Summation of x

x -> value, vector or matrix

returns

y -> if x is a matrix, return a vector with the summation of each column, if a is a vector, return a scalar.

y = sumsq(x)

Summation of square x

x -> value, vector or matrix

returns

y -> returns the following operation sum(x.*x)

y = resize(x,[height width])

resize matrix x

a -> matrix value

height -> integer value

width -> integer value

returns

y -> a matrix with the dimension of height and width, if the matrix is bigger, then is filled with 0

y = xcorr(a)

y = xcorr(a,b)

cross correlation

a -> vector value

returns

y -> returns cross correlation of a and b, if b is not defined, then return the autocorrelation of a.

y = zeros(a)

y = zeros(a,b)

matrix of zeros

a -> integer value

b -> integer value (default b=a)

returns

y -> matrix of zeros of size a,b