MATLAB
Scalars, vectors and matrixes

Edited by Péter Vass

Matrix

The fundamental data structure of MATLAB is the matrix, that is a two-dimensional set of numbers arranged in a tabular form.

It has two dimensional parameters:

- the number of rows (n),
- the number of columns (m).

The size of a matrix can be given by these values (n x m).

While a row vector is treated as a 1-by m matrix, a column vector is an n-by-1 matrix.

That a scalar is a 1-by-1 matrix, results from the previous ones.

<u>Example:</u>

```
>> a=10;
>> size(a)
```

Matrix

Creating a matrix:

```
>> A=[5 -1 3;7 1 4;-2 0 6]
```

or

```
>> A=[5,-1,3;7,1,4;-2,0,6]
```

where

[ ]	encloses the matrix elements,
, or space	the column separator,
;	the row separator.

Check the size of the matrix and the number of its elements.

```
>> size(A)
>> numel(A)
```

Vector

Creating a row vector:

```
>>  b=[-2 a+1 0]
          or
>>  b=[-2, a+1, 0]
>> size(b)
>> length(b)
```

Creating a column vector:

```
>> c=[a/3; 2; a*0.2]
>> size(c)
>> length(c)
```

Empty matrix

Empty matrix:

```
>>  d=[]
```

Checking the type of a matrix

by mean of calling the following built-in functions:
isempty, isscalar, isvector, ismatrix

<u>Examples:</u>

```
>>isempty(a)
>>isempty(d)
>> isscalar(a)
>> isscalar(d)
```

Try out the others.

Other ways of creating row vectors

Colon operator

We can create equally spaced elements between two values.

start:stop (the increment is one)

start:increment or decrement:stop

<u>Examples</u>

```
>>x=0:10
>>y=0:-0.2:-4
>>z=2:pi/10:5
```

Calling the linspace function

linspace(*start*, *stop*) (creates 100 equally spaced elements)

linspace(*start*, *stop*, *number of elements*)

<u>Examples</u>

```
>>x=linespace(0,10)
>>y=linespace(0,2,10)
```

Other ways of creating row vectors

Calling the logspace function

We can create logarithmically spaced elements between two values

logspace(*start*, *stop*) (creates 50 logarithmically spaced elements between 10^*start* and 10^*stop*)

logspace(*start*, *stop*, *number of elements*)

<u>Examples</u>

```
>>x=logspace(0,2)
>>y=logspace(0,2,10)
```

Finding the minimum and maximum value of a matrix or vector

```
>>min(x)
```

```
>>max(x)
```

Creating special matrixes

Creating a matrix of ones

```
>>X=ones(2,3)
>>x=ones(3,1)
```

Creating a matrix of zeros

```
>>Y=zeros(2,3)
>>y=zeros(1,3)
```

Creating a matrix with uniformly distributed random elements (between 0 and 1)

```
>>X=rand(3,2)
```

Creating a matrix with normally distributed random elements (drawn from standard nomal distribution)

```
>>Y=randn(2,3)
```

Creating special matrixes

How we can create a row vector of random elements drawn from a normal distribution with mean 2 and standard deviation 5 ?

```
>>x=2+5*randn(1,10)
```

Creating an identity matrix of zeros

```
>>Y=eye(4,4)
```

Creating a diagonal matrix with specified values

```
>>X=diag([2, -5, 0.5])
```

Extracting the diagonal elements of a matrix

```
>>Y=randn(3,3)
>>y=diag(Y)
```

Concatenating matrixes

We can merge submatrixes but must take care to fit the dimensions

Examples

```
>>Z=[X,Y]
>> V=[X;Y]
>> V=[V; [0, 0, 0]]
>> V=[V [10; 10; 10; 10; 10; 10; 10]]
```

We can also replicate a submatrix.

Example

```
Z=repmat(X,2,3)
```

Referencing the elements

We can modify or use in expressions a single or a range of vector elements by means of proper referencing.

Referencing a single element of a vector

variable_name(*i*)

Examples

```
>> x=10*rand(1,9)
>> x(3)
>> x(3)=x(3)+3
```

Referencing a range of elements from a vector

variable_name(*start_index* : *stop_index*)

Examples

```
>> y=x(3:6)
>> x(5)=sum(2:4)
```

Referencing the elements

Referencing a range of elements from a vector

Examples

```
>> y=x(3:end)
>> mean(y)
>> std(y)
```

mean, *std* and *median* are statistical functions for computing the arithmetical mean, the standard deviation and the median of a data series. (See MATLAB help).

Referencing a single element of a matrix

variable_name(*i* , *j*)

Examples

```
>> X=10*rand(3,4)
>>X(2,3)
```

Referencing the elements

Referencing a row or a column of a matrix

variable_name(i , :)

variable_name(: , j)

Examples

```
>> X(1, :)
>>X(:, 1)
```

Referencing a submatrix of a matrix

variable_name($i_1 : i_2$, $j_1 : j_2$)

Examples

```
>>X(2:3,1:3)
>>X(:, 1:3)
>> X(:, 2:end)
>>X(2:3, :)
```

Deleting rows or column

Deleting rows or columns from a matrix

by means of assigning an empty matrix to the range of rows or columns of a matrix.

<u>Examples</u>

```
>> X(:, 3:4)=[ ]
>>X(2, :)=[ ]
```

Vector operations

addition and subtraction
multiplications
transpose a vector
appending vectors
magnitude of a vector

Vector operations

Addition and subtraction of two vectors

Condition: both vectors must be row or column with the same length (number of element).

<u>Examples</u>

```
>> a=[5, 7, -1, 2]; b=[2, 3, 0, -4];
>>c=a+b;
>>disp('a+b=');disp(c);
>>d=a-b;
>> disp('a-b=');disp(d);
>>e=b-a;
>> disp('b-a=');disp(e);
```

Vector operations

Scalar multiplication of vectors

```
>>a=3*a
>>b=-5*[3, 2, -8, 0]
```

Transpose of a vector

```
>>bT=b'
```

Scalar product of two vectors (or dot product)

Condition: the operands must be a row and a column vector in sequence, and they have the same length (number of elements).

```
>> c=a*bT
```

Dyadic multiplication

Condition: the operands must be a column and a row vector in sequence, and they have the same length.

```
>>d=bT*a
```

Vector operations

Element-wise multiplication

```
>>c=a.*b
```

Element-wise raising to power

```
>>c=a.^2
```

Magnitude of a vector

```
>>cmag=sqrt(sum(c.*c))
```

Appending vectors

```
>>c=[a, b]
>>c=[a; b]
```

Matrix operations

Matrix operations

addition and subtraction
scalar operations of matrixes
matrix multiplication
transpose of a matrix
division of matrixes
determinant of a matrix
inverse of a matrix
powers of a matrix

Addition and subtraction of matrixes

Condition:

both of the matrixes must have the same size (number of rows and columns).

Matrix operations

Example:

```
>>A = [ -1 2 3 ; 4 -5 6; 7 8 9];
>>B = [ 7 5 -6 ; 2 0 8; 5 -7 1];
>>C = A + B
>>D = A – B
```

Scalar operations of matrixes

A scalar (number) can be added to, subtracted from, multiplied by a matrix, and a matrix can be divided by it.

The result is a new matrix which has the same size as the operand matrix.

Its elements are the increased, decreased, multiplied and divided elements of the operand matrix.

Matrix operations

Example

```
>>A = [ 8 20 23 ; 14 8 6; 27 8 9];
>>s=4;
>>B=A+s
>>C=A-s
>>D=A*s
>>F=A/s
```

Multiplication of two matrixes

Condition:

the number of columns of the first operand must be equal to the number of rows of the second operand.

The result is a matrix which has the same number of rows as that of the first operand and the same number of columns as that of the second operand.

Matrix operations

Example

```
>>A = [ 1, 2, 3; 4, 5, 6];
>>B = [2, 4; 6, 8; 10, 12];
>>C = A*B
```

Element-wise multiplication of two matrix

Condition:

The operand matrixes must have the same size.

```
>>A = [ -1 2 3 ; 4 -5 6; 7 8 9];
>>B = [ 7 5 -6 ; 2 0 8; 5 -7 1];
>>C = A.*B
```

Transpose of a matrix

The transpose operation flips a matrix over its diagonal.

```
>>CT=C'
```

Matrix operations

Division of two matrixes

There are two kinds of matrix division.

Backslash character (\) symbolizes the *left division* and the slash character (/) is used for the notation of *righ division*.

Condition:

Both of the operands (matrixes) must have the same size.

Let A and B denote the two matrixes.

The left division of A by B is X,

A\B=X

and X fulfils the following equation

A*X=B

The right division of A by B is Y,

A/B=Y

and Y fulfils the following equation

Y*B=A

Matrix operations

X and Y do not exist in all cases.
For some matrixes, the divisions can not be performed or the result is uncertain.
The relationship between the left and right divisions is the following:
B/A=A'\B'

Example

```
>> A =[1, 1; -2, 6];
>> B=[3, -5; 7, 2];
>> X=A\B
>>A*X
>>Y=A/B
>>Y*B
>>B'\A'
```

Matrix operations

Element-wise division of two matrix

Condition:

The operand matrixes must have the same size.

Example

```
>> A =[1, 1; -2, 6];
>> B=[3, -5; 7, 2];
>> C=A.*B
```

Determinant of a matrix

Condition:

The matrix must be square.

Example:

```
>>A = [ 1 2 3; 4 3 2; 0 3 5];
>>det(A)
```

Matrix operations

Inverse of a matrix

Condition:

The matrix must be square and its determinant can not be equal to zero.

If the determinant of the matrix is zero, the inverse does not exist and the matrix is singular.

The product of the matrix and its inverse is an identity matrix.

Example:

```
>>Ai=inv(A)
>>A*Ai
```

Powers of a matrix

Condition:

The matrix must be square and the exponent must be an integer number.

Matrix operations

If the exponent is a positive integer, the matrix is multiplied by itself as many times as the value of the exponent.
If the exponent is zero, the result is an identity matrix.
If the exponent is -1, the result is the inverse of the matrix (if it exists).
If the exponent is another negative integer, the inverse matrix is multiplied by itself as many times as the absolute value of the exponent.
Example:

```
>>A2=A^2
>>A*A
>>A3=A^3
>>A*A*A
>>A^-1
>>inv(A)
>>A_4=A^-4
>>inv(A)^4
```

Matrix operations

Element-wise power

Condition:

The two matrix operands must have the same size.

Each element of the first matrix is raised to the power of the correspondent element in the other matrix.

Example:

```
>>A.^A2
```