Linear Algebra in MATLAB

Jiseok Chae

Department of Mathematical Sciences
KAIST

Week 12

Note

We already have done some linear algebra in MATLAB, but there are still a lot more features provided by MATLAB we have not yet seen.

This time, we will see some of the builtin MATLAB functions related to linear algebra, focusing on the ones that are more widely used.

Contents

A matrix in MATLAB is not only a two dimensional array, but also can be seen as a long column vector that is “folded”.

Treating a matrix as a “folded” vector, something strange happens:

```
>> A = zeros(3, 4);
>> for i = 1 : 12
A(i) = i;
end
>> A

A =

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

The command `reshape` reshapes the matrix by changing the way it is "folded".

```
>> B = reshape(A, [4, 3])

B =

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

Concatenating matrices can be done by writing the block form of a matrix, as if matrices are scalars.

```
>> A = [1, 2, 3; 4, 5, 6]; B = [10, 20; 30, 40]; C = [100; 200];
>> [A, B, C]

ans =

     1     2     3    10    20   100
     4     5     6    30    40   200

>> [A; B, C]

ans =

     1     2     3
     4     5     6
    10    20   100
    30    40   200
```

The function `diag` can be used in two ways. It can either extract the main diagonal of a matrix to construct a vector, or create a diagonal matrix with diagonal entries being the entries of the given vector.

```
>> C = [1, 2, 3; 4, 5, 6; 7, 8, 9];
>> d = diag(C)

d =

     1
     5
     9

>> D = diag(d)

D =

     1     0     0
     0     5     0
     0     0     9
```

A backslash(\) operator denotes a “reverse” division. In many Korean keyboards, a backslash is replaced by a won sign(₩), so you may have to type this key instead.

```
>> 4/2

ans =

     2

>> 4\2

ans =

    0.5000
```

Note that for two scalars a and b, the reverse division $\mathtt{x} = \mathtt{a} \setminus \mathtt{b}$ computes x such that $\mathtt{ax} = \mathtt{b}$. This extends to the expression $\mathtt{A} \setminus \mathtt{b}$ where A is a matrix and b is a vector.

In particular, if A is invertible, then A\b is equal to $\mathtt{A}^{-1}\mathtt{b}$.

If the system $\mathtt{Ax} = \mathtt{b}$ does not have a solution but A has full row rank, then A\b computes the *least squares solution* of the system $\mathtt{Ax} = \mathtt{b}$.

In other cases, analogous to the situation of dividing by zero, you will get a warning message.

In many cases, the names of the builtin functions are straightforward abbreviations of the mathematical terms. Some examples are:

function	what it computes
`det(A)`	the determinant of `A`
`rank(A)`	the rank of `A`
`rref(A)`	the reduced row echelon form of `A`
`adjoint(A)`	the adjoint of `A`

Contents

A decomposition of a matrix is representing a matrix into a product of matrices that has a special structure.

The LU decomposition is one example of a decomposition.

```
>> A = magic(3); % A 3x3 magic square
>> [L, U] = lu(A);

L =

    1.0000         0         0
    0.3750    0.5441    1.0000
    0.5000    1.0000         0

U =

    8.0000    1.0000    6.0000
         0    8.5000   -1.0000
         0         0    5.2941
```

To get the LU decomposition of the form $PA = LU$ for a permutation matrix P, the output format should be changed as below.

```
>> [L, U, P] = lu(A)

L =

    1.0000         0         0
    0.5000    1.0000         0
    0.3750    0.5441    1.0000

U =

    8.0000    1.0000    6.0000
         0    8.5000   -1.0000
         0         0    5.2941

P =

     1     0     0
     0     0     1
     0     1     0
```

The QR decomposition can be done by `qr`.

```
>> A = magic(4); % A 4x4 magic square
>> [Q, R] = qr(A);

Q =

   -0.8230    0.4186    0.3123   -0.2236
   -0.2572   -0.5155   -0.4671   -0.6708
   -0.4629   -0.1305   -0.5645    0.6708
   -0.2057   -0.7363    0.6046    0.2236

R =

  -19.4422  -10.5955  -10.9041  -18.5164
         0  -16.0541  -15.7259   -0.9848
         0         0    1.9486   -5.8458
         0         0         0    0.0000
```

Diagonalization can be done by eig, but this function behaves slightly differently to other functions.

```
>> A = magic(3);
>> [P, D] = eig(A);
>> [P, D]

ans =

  -0.5774   -0.8131   -0.3416    15.0000          0          0
  -0.5774    0.4714   -0.4714          0     4.8990          0
  -0.5774    0.3416    0.8131          0          0    -4.8990

>> P * D * inv(P)

ans =

    8.0000     1.0000     6.0000
    3.0000     5.0000     7.0000
    4.0000     9.0000     2.0000
```

The command `d = eig(A)` does not store the diagonal matrix of eigenvalues nor the matrix of eigenvectors to `d`. Instead, `d` becomes the *vector* of eigenvalues.

```
>> d = eig(A)

d =

   15.0000
    4.8990
   -4.8990
```

To compute the singluar value decomposition, use `svd`.
Be careful of the matrix `V` of right singluar vectors.

```
>> A = magic(4); % A 4x4 magic square, but...
>> [U, S, V] = svd(A(1:3)); % we use only the first 3 rows.
>> U * S * V

ans =

   14.4844    5.1688   10.9411    9.0431
    6.3537   13.3857    9.1054   -2.7469
    9.4926   10.8477    9.2152    4.1592

>> U * S * V'

ans =

   16.0000    2.0000    3.0000   13.0000
    5.0000   11.0000   10.0000    8.0000
    9.0000    7.0000    6.0000   12.0000
```

Thank you!