Play with C-Matrix

1 Introduction

This tiny program is a simple implementation for matrix calculation. I had ever been an employee of Oracle CDC as a JDBC/UCP QA tester for 21 months which, although a very short time, is the most peaceful and happy days during my career life. This little program is my tribute to Oracle whose greatness goes far beyond its products.

2 General information

C-Matrix is written in C++ using a few C++ 11 features and it needs a compiler supporting these features, I use gcc 10.2.1 on Fedora 32 to compile it. Also it needs flex and bison to generate a simple scanner and an even more simple parser.

C-Matrix mostly contains a API along with a shared library and an executable program. The former is two head files in the include folder and a so file generated in the bin folder. The latter is a interactive command line tool to run some matrix calculation. Just running make will get these files.

The test folder include a sample program to illustrate how to use the API, covering most of the interfaces.

3 Interactive tool

The interactive tool xinyuw generated in the bin folder depends on the previously mentioned so file. Before starting it I always run the env.sh to set the $LD_LIBRARY_PATH:

```
[wxy@localhost matrix]$ . env.sh
[wxy@localhost matrix]$ bin/xinyuw
>
```

3.1 Vector scalar computation

```
>a=<345,-5465454543,32344,3345>
>a
 345  -5465454543  32344  3345
>a=<1,2,3,-6,2.7E5>
>a
 1.000000  2.000000  3.000000  -6.000000  270000.000000
>b=a*2; c=a/2
>b
 2.000000  4.000000  6.000000  -12.000000  540000.000000
>c
 0.500000  1.000000  1.500000  -3.000000  135000.000000
```

3.2 Vector plus/minus

```
>a=<1,2,3,-6,2.7E5>
>b=<7,99, -100, -5456.35, 735>
```

```
>c=a + b;  d = a-b;
>c
 8.000000  101.000000  -97.000000  -5462.350000  270735.000000
>d
 -6.000000  -97.000000  103.000000  5450.350000  269265.000000
```

3.3 Matrix zeros/ones/eye/rand

```
>zeros(2,4)
2 X 4
 0.000000  0.000000  0.000000  0.000000
 0.000000  0.000000  0.000000  0.000000
>a=zeros(3, 'int64') ; a
3 X 3
 0  0  0
 0  0  0
 0  0  0
>b=ones(3,2,'single'); b
3 X 2
 1.000000  1.000000
 1.000000  1.000000
 1.000000  1.000000
>c=eye(3, 'int16'); c
3 X 3
 1  0  0
 0  1  0
 0  0  1
>d=rand(3,5)
>d
3 X 5
 0.968504  0.977058  0.256706  0.106121  0.017222
 0.251612  0.639529  0.468474  0.352592  0.614893
 0.784169  0.179397  0.355693  0.198715  0.942123
```

The last parameter is a optional string, it may be single|double|int8|int16|int32|int64|uint8|uint16|uint32|uint64. The default value is “double”.

3.4 Matrix diag

```
>a=<10, -20, 30, -40, 50>
>b=diag(a); c=diag(a,2); d=diag(c, 2)
>a
 10  -20  30  -40  50
>b
5 X 5
 10    0   0    0   0
  0  -20   0    0   0
  0    0  30    0   0
  0    0   0  -40   0
  0    0   0    0  50
>c
```

```
7 X 7
 0  0  10    0   0    0   0
 0  0   0  -20   0    0   0
 0  0   0    0  30    0   0
 0  0   0    0   0  -40   0
 0  0   0    0   0    0  50
 0  0   0    0   0    0   0
 0  0   0    0   0    0   0
>d
 10  -20  30  -40  50
```

3.5 Matrix blkdiag

```
>a=[1,2,3;-4,-5,-6]; b=rand(3,4); c=eye(2,'int64')
>b
3 X 4
 0.575080  0.877710  0.921801  0.814696
 0.588162  0.066109  0.398154  0.852708
 0.971620  0.856323  0.596885  0.516978
>d=blkdiag(c,b*1000,a,'int32')
>d
7 X 9
 1  0    0    0    0    0   0   0   0
 0  1    0    0    0    0   0   0   0
 0  0  575  877  921  814   0   0   0
 0  0  588   66  398  852   0   0   0
 0  0  971  856  596  516   0   0   0
 0  0    0    0    0    0   1   2   3
 0  0    0    0    0    0  -4  -5  -6
```

3.6 Matrix cat

```
>a=rand(2,4,'int8'); b=ones(2,4,'int8');
>c=cat(1,a,b)
>c
4 X 4
  102.000000  -116.000000   8.000000  -37.000000
  -97.000000    -2.000000  97.000000  116.000000
    1.000000     1.000000   1.000000    1.000000
    1.000000     1.000000   1.000000    1.000000
>a=rand(3,2,'int8'); b=eye(3,'int8')
>c=cat(2,a,b,'int32')
>a
3 X 2
  78  -51
  98   27
  50   43
>b
3 X 3
  1  0  0
  0  1  0
  0  0  1
>c
3 X 5
  78  -51  1  0  0
  98   27  0  1  0
  50   43  0  0  1
```

The first parameter of cat is a integer, 1 means 1st-dimension, 2 means 2nd-dimension.

3.7 Matrix plus/minus/multiply/inverse/left-divide/right-divide

```
>a=[1,2,3;-4,-5,-6;7,8,9]; b=[-9,-8,-7;3,2,1;-6,-5,-4]
>a+b
3 X 3
  -8  -6  -4
  -1  -3  -5
   1   3   5
>a-b
3 X 3
  10  10  10
  -7  -7  -7
  13  13  13
>a*b
3 X 3
  -21  -19  -17
   57   52   47
  -93  -85  -77
>inv(a)
3 X 3
   1.000000   2.000000   1.000000
  -2.000000  -1.000000   0.000000
   1.666667   0.666667  -0.000000
>a/b
3 X 3
   0.000000  -2.500000  -1.000000
  -1.000000   2.000000   0.000000
   0.000000  -3.000000  -2.000000
>a\b
3 X 3
   -9.000000   -9.000000   -9.000000
   15.000000   14.000000   13.000000
  -13.000000  -12.000000  -11.000000
```

3.8 Matrix scalar computation

```
>a=rand(3)
>a
3 X 3
  0.752818  0.110893  0.456079
  0.018749  0.092940  0.757419
  0.409571  0.948294  0.310282
```

```
>a+5; a-5; a*5; a/5
3 X 3
  5.752818  5.110893  5.456079
  5.018749  5.092940  5.757419
  5.409571  5.948294  5.310282
3 X 3
  -4.247182  -4.889107  -4.543921
  -4.981251  -4.907060  -4.242581
  -4.590429  -4.051706  -4.689718
3 X 3
  3.764089  0.554465  2.280397
  0.093746  0.464699  3.787095
  2.047853  4.741471  1.551411
3 X 3
  0.150564  0.022179  0.091216
  0.003750  0.018588  0.151484
  0.081914  0.189659  0.062056
```

3.9 Matrix transpose

```
>a=rand(3,3,'int8')
>a
3 X 3
  -113  -126   34
   -86   -69  -29
  -126    69   85
>transpose(a)
3 X 3
  -113  -86  -126
  -126  -69    69
    34  -29    85
```

3.10 Matrix determinant

```
>a=rand(5)
>a
5 X 5
  0.885460  0.849160  0.229303  0.671743  0.274362
  0.255110  0.967034  0.259178  0.199648  0.003610
  0.156689  0.125241  0.793604  0.728754  0.619799
  0.453210  0.226598  0.532940  0.606014  0.209812
  0.962982  0.731073  0.022883  0.297523  0.808525
>det(a)
-0.077645
```

This function use a recursion algorithm which takes a O(n!) complexity, tremendous sub-matrix are generated and destroyed during it's execution. The max scale I ever tried is 13, it seems never return when the size goes up to 14.

3.11 quit the program

Both Ctrl-D and quit will terminate the program.