How to run code

$ bash hw3_1.sh

$ bash hw3_2.sh

$ bash hw3_3.sh

$ ./hw3_1

$ ./hw3_2

$ ./hw3_3

Results

```
(base) qmin@rose:~/na/Homework #3/Code$ ./hw3_1
########## Using Gauss-Jordan Elimination ##########
Answers for A2
-2.873567 -0.612357 0.976277 0.635819 -0.553441
Answers for A3
-0.326608 1.532293 -1.044825 -1.587447 2.928480 -2.218931
total_time 0.000048

########## Using LU decomposition ##########
Answers for A1
1.000000 -3.000000 2.000000 0.000000
Answers for A2
-2.873566 -0.612357 0.976277 0.635819 -0.553441
Answers for A3
-0.326608 1.532292 -1.044826 -1.587447 2.928480 -2.218930
total_time 0.000044

########## Using svd ##########
Answers for A1
1.293531 -2.412938 1.119406 -0.293531
Answers for A2
-2.873566 -0.612357 0.976278 0.635819 -0.553441
Answers for A3
-0.326609 1.532292 -1.044825 -1.587447 2.928479 -2.218929
total_time 0.000071
```

```
(base) qmin@rose:~/na/Homework #3/Code$ ./hw3_2
-2.873566389083862304687500000 -0.612356722354888916015625000 0.976277351379394531250000000 0.635818660259246826171875000 -0.553441107273101806640625000
Is it improved??
-2.873566150665283203125000000 -0.612356543540954589843750000 0.976277351379394531250000000 0.635818481445312500000000000 -0.553441047668457031250000000
-2.873566389083862304687500000 -0.612356662750244140625000000 0.976277351379394531250000000 0.635818600654602050781250000 -0.553441107273101806640625000
-2.873566150665283203125000000 -0.612356603145599365234375000 0.976277351379394531250000000 0.635818481445312500000000000 -0.553441047668457031250000000
-2.873566389083862304687500000 -0.612356662750244140625000000 0.976277351379394531250000000 0.635818600654602050781250000 -0.553441107273101806640625000
-2.873566150665283203125000000 -0.612356603145599365234375000 0.976277351379394531250000000 0.635818481445312500000000000 -0.553441047668457031250000000

=======
-0.326607882976531982421875000 1.532292485237121582031250000 -1.044825673103332519531250000 -1.587447047233581542968750000 2.928479671478271484375000000 -2.218930006027221679687500000
Is it improved??
-0.326607853174209594726562500 1.532292366027832031250000000 -1.044825196266174316406250000 -1.587447643280029296875000000 2.928480148315429687500000000 -2.218930482864379882812500000
-0.326607942581176757812500000 1.532293200492858886718750000 -1.044825315475463867187500000 -1.587447762489318847656250000 2.928480386734008789062500000 -2.218931674957275390625000000
-0.326607942581176757812500000 1.532292842864990234375000000 -1.044825553894042968750000000 -1.587447524070739746093750000 2.928480148315429687500000000 -2.218930959701538085937500000
-0.326607853174209594726562500 1.532292366027832031250000000 -1.044824957847595214843750000 -1.587448000907897949218750000 2.928480148315429687500000000 -2.218930721282958984375000000
-0.326608002185821533203125000 1.532292962074279785156250000 -1.044825434684753417968750000 -1.587447643280029296875000000 2.928480386734008789062500000 -2.218931198120117187500000000
```

```
(base) qmin@rose:~/na/Homework #3/Code$ ./hw3_3
Answers for A1
inverse matrix
-100000002004087734272.000000   -37500000751532900352.000000    37500000751532900352.000000     12500001350022594560.000000
-200000056784733601792.000000   -75000019095251845120.000000    75000010299158822912.000000     25000004899068444672.000000
300000041196635291648.000000    112500015448738234368.000000    -112500006652645212160.000000   -37500005149579411456.000000
100000002004087734272.000000    37500000751532900352.000000     -37499996353486389248.000000    -12500001350022594560.000000
determinant
-0.000000
Answers for A2
inverse matrix
0.354536        0.766945        0.207769        -0.595412       0.253128
0.035454        0.126695        0.195777        -0.159541       0.050313
-0.138686       -0.098540       -0.096715       0.124088        0.016423
-0.052138       -0.303962       -0.023201       0.234619        -0.044578
0.149114        0.459333        0.051356        -0.171011       0.042492
determinant
3835.999512
Answers for A3
inverse matrix
-0.162205       0.122801        0.024068        -0.016431       -0.022840       0.046132
0.169407        -0.041117       0.228313        -0.087624       0.180306        -0.395655
-0.011636       0.122745        -0.117407       -0.180981       0.015910        0.186766
0.105669        -0.051726       -0.108916       0.299774        0.000859        -0.190541
-0.053026       -0.042362       0.160508        -0.224034       0.161811        0.015024
-0.062341       -0.064694       -0.234216       0.351126        -0.364828       0.434633
determinant
16178.401367
```

Discussion

1.

총 걸린 시간은 SVD >>>> Gauss-Jordan > LU 이다.

SVD는 3개의 matrix로 분해하고 다시 역행렬을 취하고 계산해야 하기 때문에 직관적으로 다른 2가지 방법보다 계산량이 많아 시간이 오래걸린 것으로 생각된다. Gauss-Jordan ,LU는 시간이 걸린 시간이 거의 비슷하다.

LU와 SVD는 singular한 matrix lineq1에 대해서도 근사적으로 해를 구할 수 있다.

하지만 Gauss-jordan은 해를 구하지 못하고 종료된다.

2.

mprove 함수를 적용하면 solution값이 바뀐다. 하지만 원래 정확한 solution을 모르기 때문에 실제로 더 정확한 solution으로 바뀌었는지는 실험결과 값을 통해 확인은 불가능하다.

Numerical recipe 교재에서는, mprove를 1~2번 call 하는 것만으로도 충분히 수렴한다고 언급되어 있다. ./hw3_2에서 mprove를 5번 call한 결과값을 출력하고 있는데, 값이 크게 변하지 않는 점을 보아, 교재의 말을 실험결과값을 통해 확인 할 수 있었다.

3.

LU decomposition을 하고나면, back subsitituion을 통해 쉽게 역행렬을 구할 수 있음을 확인했다. 또한, determinant도, 대각행렬의 값을 곱 하면 되기 때문에, 간단하게 구할 수 있다.