HPC Spring 2023: Advanced Topics in Numerical Analysis:

Assignment 1

Keigo Ando (ka2705)

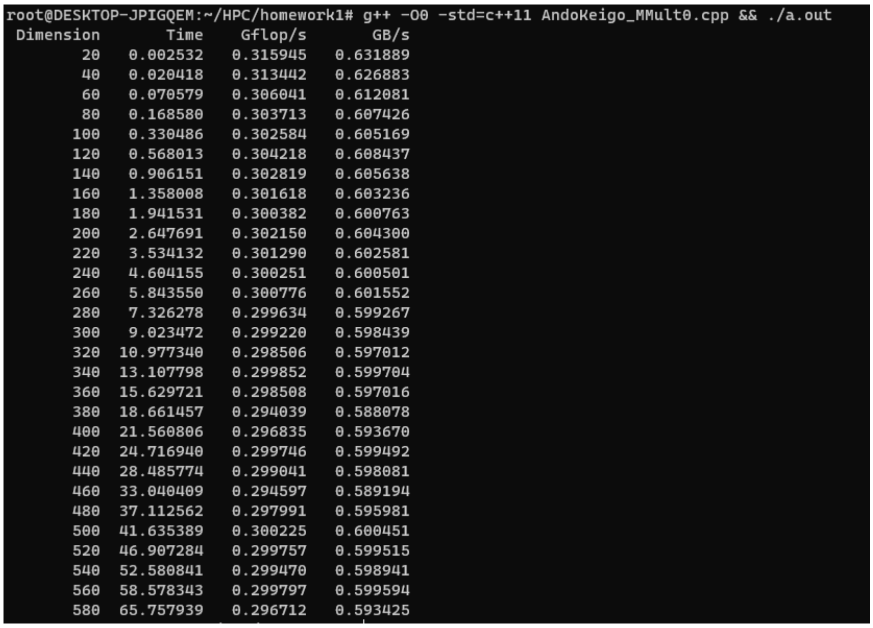
Problem2

My implementation can be seen in the source file `AndoKeigo_MMult0.cpp`

The processor I used for thsi assignment is an 8 core AMD Ryzen 7 4700 with Radeon Graphics. Cpu MHz is 1996.193 and cache size is 512 KB. And the program is executed on the WSL2 enviroment.

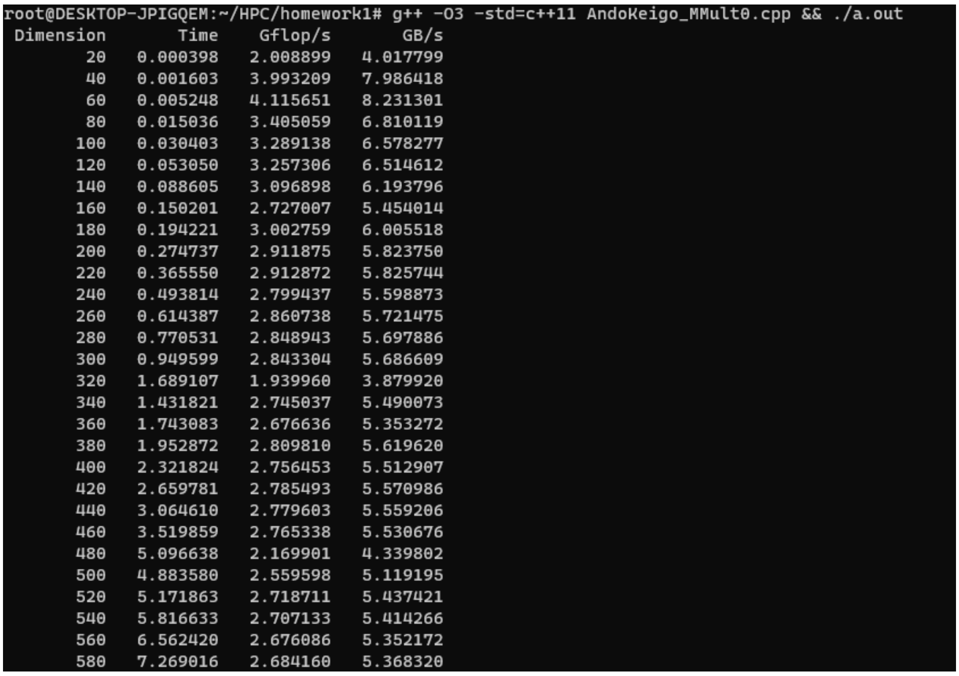
The flop rate and bandwidth for each optimizaion flag are as follows:

- -O0



```
root@DESKTOP-JPIGQEM:~/HPC/homework1# g++ -O0 -std=c++11 AndoKeigo_MMult0.cpp && ./a.out
 Dimension       Time    Gflop/s       GB/s
        20   0.002532   0.315945   0.631889
        40   0.020418   0.313442   0.626883
        60   0.070579   0.306041   0.612081
        80   0.168580   0.303713   0.607426
       100   0.330486   0.302584   0.605169
       120   0.568013   0.304218   0.608437
       140   0.906151   0.302819   0.605638
       160   1.358008   0.301618   0.603236
       180   1.941531   0.300382   0.600763
       200   2.647691   0.302150   0.604300
       220   3.534132   0.301290   0.602581
       240   4.604155   0.300251   0.600501
       260   5.843550   0.300776   0.601552
       280   7.326278   0.299634   0.599267
       300   9.023472   0.299220   0.598439
       320  10.977340   0.298506   0.597012
       340  13.107798   0.299852   0.599704
       360  15.629721   0.298508   0.597016
       380  18.661457   0.294039   0.588078
       400  21.560806   0.296835   0.593670
       420  24.716940   0.299746   0.599492
       440  28.485774   0.299041   0.598081
       460  33.040409   0.294597   0.589194
       480  37.112562   0.297991   0.595981
       500  41.635389   0.300225   0.600451
       520  46.907284   0.299757   0.599515
       540  52.580841   0.299470   0.598941
       560  58.578343   0.299797   0.599594
       580  65.757939   0.296712   0.593425
```

- -O3



```
root@DESKTOP-JPIGQEM:~/HPC/homework1# g++ -O3 -std=c++11 AndoKeigo_MMult0.cpp && ./a.out
 Dimension       Time    Gflop/s       GB/s
        20   0.000398   2.008899   4.017799
        40   0.001603   3.993209   7.986418
        60   0.005248   4.115651   8.231301
        80   0.015036   3.405059   6.810119
       100   0.030403   3.289138   6.578277
       120   0.053050   3.257306   6.514612
       140   0.088605   3.096898   6.193796
       160   0.150201   2.727007   5.454014
       180   0.194221   3.002759   6.005518
       200   0.274737   2.911875   5.823750
       220   0.365550   2.912872   5.825744
       240   0.493814   2.799437   5.598873
       260   0.614387   2.860738   5.721475
       280   0.770531   2.848943   5.697886
       300   0.949599   2.843304   5.686609
       320   1.689107   1.939960   3.879920
       340   1.431821   2.745037   5.490073
       360   1.743083   2.676636   5.353272
       380   1.952872   2.809810   5.619620
       400   2.321824   2.756453   5.512907
       420   2.659781   2.785493   5.570986
       440   3.064610   2.779603   5.559206
       460   3.519859   2.765338   5.530676
       480   5.096638   2.169901   4.339802
       500   4.883580   2.559598   5.119195
       520   5.171863   2.718711   5.437421
       540   5.816633   2.707133   5.414266
       560   6.562420   2.676086   5.352172
       580   7.269016   2.684160   5.368320
```

Problem3

(a) (b)

See the source file `AndoKeigo_LaplaceEq.cpp`

(c)

The number of iterations for each methods for different numbers can be seen as follows:

N	Jacobi Method	Gauss-Seidel Method
10	222	112
1000	1848917	924460
100000	Do not converge after 1000000 interations	Do not converge after 1000000 interations

The following table describes run times for N = 100000 for 50 iterations with Gauss Seidel method using different compiler optimization flags. (It took an inordinately long time to run the program with optimization flag -O0, I did that with 50 instead of 100 for the maximum iteration.)

Opt flag	Run times (s)
-O0	2276.006599
-O3	627.933401

I also implement the program `AndoKeigo_LaplaceEq_modi.cpp` for faster computation to avoide a large number of iterations in matrix computation (not paralrellizing but just reducing the number of operations in loop). However, I guess this implementation is out of the objectives of this assignement. So I include this file just for reference.

The computer architecture I used for this experiment is the same as in Problem 2.