Intensive Computation 2020-2021
17th March 2021

HOMEWORK 2
Linear Systems

1) Write a C program where you implement GENP (Gaussian Elimination without pivoting) and GE*P (Gaussian Elimination with either Partial, Complete or Rook pivoting – choose one) to solve linear systems.
Run the two procedures on randomly generated linear systems $Ax = b$ considering different sizes n approximately between 1000 and 10000.
For each matrix size, run a number of tests in order to obtain sufficient statistical confidence.
For each test and for each algorithm, consider the execution time and the error $\epsilon = ||A\,\tilde{x} - b||$, where $\tilde{x}$ is your computed solution.
Make two graphs using any plotting platform showing how errors and execution times change with the growth of the system size n for both methods.

Your program should look something like this:

```
runTests:
for n = 1000:1000:10000
        for ex = 1:noExperiments
                generate random A nxn and b nx1;
                run GEPP on Ax=b
                store execution time and error
                run GE*P on Ax=b
                store execution time and error
        end
end
```

2) Write a C program for the solving randomly generated linear systems using the Cholesky decomposition.
Test the algorithm over randomly generated linear systems $Ax = b$ of size n approximately between 1000 and 10000. When you generate the linear systems, make sure A **is symmetric and strictly diagonally dominant, with positive entries in its principal diagonal**. This will ensure convergence. An example of symmetric, strictly diagonally domaninat matrix with positive entries in its principal diagonal is:

```
A =  3  1  0 -1
        1  8 -4 -2
        0   -4   10    3
       -1   -2   3    8
```

Compare the results that you get in terms of computational time and solution with the output of the LAPACK function *posv. Implement a similar scheme as runTests.

For comparing the solutions obtained by your Cholesky implementation and Lapack's, compute $\epsilon = ||\tilde{x} - x_L||$, where $\tilde{x}$ is your solution and x_L is the solution obtained by Lapack.

Plot your results.

Installing LAPACK and compiling

Go to: http://www.netlib.org/lapack/
Click on software and download the library.
install with cmake and make (follow the instruction on the README). You will be needing a fortran compiler (gfortran).

To compile your project, project.c, use the following line:

```
gcc -O3 project.c -o project -lcblas -llapacke -llapack
```

LAPACK documentation: http://www.netlib.org/lapack/explore-html/

LAPACK documentation for the dposv function:
http://www.netlib.org/lapack/explore-html/dc/de9/group__double_p_osolve_ga9ce56acceb70eb6484a768eaa841f70d.html - ga9ce56acceb70eb6484a768eaa841f70d

OTHERWISE:
You might want to use the Intel MKL (Math Kernel Library).
You need to install the Intel oneAPI toolkit:
https://software.intel.com/content/www/us/en/develop/tools/oneapi/base-toolkit/download.html
And follow the installation instructions.

To compile your project, project.c, use the following lines:

```
>> source <oneapi_install_dir>/setvars.sh
>> gcc -O3 -DMKL_ILP64 project.c -o project -lmkl_intel_ilp64 -lmkl_core -lmkl_sequential -lm
```

Line advisor for linking:
https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/onemkl/link-line-advisor.html

MKL documentation:
https://software.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top.html

MKL documentation for the *posv function:
https://software.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top/lapack-routines/lapack-linear-equation-routines/lapack-linear-equation-driver-routines/posv.html

SEE ATTACHED FILES geLapack.c and geMKL.c for example including Lapack and MKL calls to the dgesv function for solving linear systems with the Gaussian Elimination.