Assignment II

This is the last of the two assignments in this course. Like in the first assignment, you will write a short program and a short report. The purpose of this assignment is to train and assess your ability to:

- Design, implement, and document a program that solves a mathematical problem.
- Debug and test mathematical software.
- Call external (third party) programs and libraries.

Please note that this assignment will count towards your final grade (approximately 10% of final grade).

Practical information

You may work on this assignment alone or in small groups of 2-3 students, but you must hand in an **individual report** regardless of whether you choose to work alone or as part of a small group. You may hand in your group's code if the code was written collaboratively by you and your group.

Deadline

Your **report** and **code** must be submitted via CampusNet no later than on **November 22, 2017**.

Report

Your report should be short and concise and **no longer than one A4 page**. If necessary, you may include a one-page appendix with code snippets. Your report should also include either a front page or a header/footer with the following information:

- course number and date
- name and student number
- group members (if applicable).

You must submit your report as a **PDF file**.

Code

You must submit your code as a single **zip file** that contains all **source files and a makefile**. You are encouraged to test your code and your makefile on the DTU Unix system (e.g. via ThinLinc) to make sure that there are no compilation errors.

Your code will be evaluated based on:

- program structure
- readability (use comments in the code, indentation, etc.)
- correctness and error checking.

Questions

Please post any questions that you may have about the assignment on Piazza.

Background

The method of least-squares is commonly used for data fitting. Given a matrix $A \in \mathbb{R}^{m\times n}$ (i.e., a matrix with m rows and n columns) and a vector $b \in \mathbb{R}^m$, the method of least-squares seeks a vector $x \in \mathbb{R}^n$ that minimizes the sum of squared residuals

$$f(x) = \sum_{i=1}^{m} (a_i^T x - b_i)^2$$

where a_i^T denotes the ith row of A and b_i is the ith element of b. Equivalently, the function $f(x)$ can be expressed as $f(x) = \|Ax - b\|_2^2$ (i.e., the squared Euclidean norm of the difference between the vector Ax and the vector b).

There are many different methods for solving least-squares problems of the form

$$\text{minimize } \|Ax - b\|_2^2.$$

Clearly, if the system $Ax = b$ is consistent, then the minimum of $\|Ax - b\|_2^2$ is zero, and the least-squares problem is equivalent to the problem of solving a system of m linear equations with n unknowns. However, this is generally not the case when $m > n$ and when the vector b consists of measurements that may be contaminated by noise. We will therefore limit our attention to the case where $m > n$.

Given a solution $x^\star$ to a least-squares problem, it is natural to ask whether there could be other solutions. This question of uniqueness can be answered using linear algebra: $x^\star$ is the unique solution if and only if the matrix A has rank n, or equivalently, the solution is unique if and only if the nullspace of A is trivial (i.e., the nullspace contains only the zero-vector). Thus, to keep things simple, we will henceforth assume that A has rank n.

When $m > n$ and A has full rank (i.e., rank n), the least-squares problem can be solved by means of a *QR factorization* of A (one can show that if $A = QR$ with $Q \in \mathbb{R}^{m\times n}$ and $R \in \mathbb{R}^{n\times n}$, then $x^\star = R^{-1}Q^T b$). Implementing a QR factorization from scratch is tedious, so instead of implementing our own QR factorization, we will use an external library called LAPACK (Linear Algebra PACKage) to solve the least-squares problem. LAPACK is originally written in Fortran, but we can link to a compiled LAPACK library when we compiler our C code. We will use a routine called **dgels** which has the following C prototype:

```
/* C prototype for LAPACK routine DGELS */
void dgels_(
    const char * trans,  /* 'N' or 'T'             */
    const int * m,       /* rows in A               */
    const int * n,       /* cols in A               */
    const int * nrhs,    /* cols in B               */
    double * A,          /* array A                 */
    const int * lda,     /* leading dimension of A */
    double * B,           /* array B                 */
    const int * ldb,      /* leading dimension of B */
    double * work,        /* workspace array         */
    int * lwork,          /* workspace size          */
    int * info            /* status code             */
);
```

Remarks: Notice that all inputs are pointers. This is because Fortran uses *call-by-reference* to pass function arguments (as opposed to *call-by-value*). Notice also the underscore after the function name `dgels`. This is added by some Fortran compilers, and it is (typically) required when using a routine from the LAPACK library in a C program. Also, it is important to note that the LAPACK library assumes column-major ordering of the elements in a matrix. The **dgels** routine is documented on the LAPACK website. Note that the routine needs some "workspace" storage for intermediate computations (the input argument `work`). According to the documentation, the length of the `work` array (i.e., the input argument `lwork`) should satisfy

$$\texttt{lwork} \geq \max(1, \min(m,n) + \max(\min(m,n), \texttt{nrhs}))$$

where the input argument `nrhs` is the "number of right-hand sides" (which should be equal to 1 when the input argument `B` represents a vector).

Assignment

Code

Write a non-interactive program (i.e., the user should not be prompted for input) that

1. reads a matrix A and a vector b from two separate text files,
2. solves the least-squares problem

$$\text{minimize } \|Ax - b\|_2^2$$

   using the `dgels` routine from the LAPACK library,
3. writes the solution $x^\star$ to a file,
4. and prints out the relative residual norm $\|Ax^\star - b\|_2 / \|b\|_2$.

The user should be able to specify the data files (the matrix A and the vector b) and the output file using command line arguments, e.g.,

```
$ ./lssolve A.txt b.txt solution.txt
```

Your program should work for problems of different sizes. However, you may assume that $m > n$ and that the matrix A has full rank (i.e., rank n).

The `source` directory included in the zip file contains a template `lssolve.c` as well as basic data structures and routines for working with matrices (see `matrix_io.h`). There is also a makefile template that can be used on Windows, Mac, and the DTU Unix system. You are encouraged to use the routines defined in `matrix_io.h` (and implemented in `matrix_io.c`) to simplify your program (e.g., you can use `read_vector()` and `read_matrix()` to read the problem data A and b, and `write_vector()` can be used to write the solution x to a file).

The zip file also contains two text files (`A.txt` and `b.txt`) that you may use as a test case. The solution to the corresponding least-squares problem is

$$x \approx \begin{bmatrix} 1.0516 \\ -2.0416 \\ 1.0429 \end{bmatrix}.$$

Windows users: Please note that Windows does not ship with BLAS/LAPACK libraries, so you will need to obtain these. A precompiled version of OpenBLAS (which contains both BLAS and LAPACK) can be downloaded from CampusNet. This library is linked against `gfortran` which is a Fortran runtime library, so if you get a linking error related to `gfortran`, rerun the TDM-GCC installer in order to add the GCC Fortran compiler to your GCC installation.

Report

Write a short report (see requirements under "Practical information") in which you briefly address the following questions:

1. Function/program structure: how did you organize your code?
2. Briefly explain how your program works: what does the user need to know in order to use your program? Are there any limitations?
3. How did you debug and/or test your program? What tests did you perform to ensure correctness? What happens if the user specifies data files with incompatible dimensions (e.g., if A is if size $m \times n$ and b is a vector of length $k \neq m$)? What happens if `dgels` fails (e.g., if A does not have full rank)?