DTU Fall 2016, d. 23 November 02635 - Mathematical Software Programming
Student: **Mattia Paterna, 163143**

ASSIGNMENT 2 REPORT

OUTLINE

The main code (*solve.c*) is divided into several sections. Each section is well separated and commented within the code. Generally speaking, a reading section opens the code and it is followed by the algebraic routine call. A final section is meant to write the code onto an output file. A preliminary section, named *0*, takes care of prompting the user about the program usage if the number of arguments is incorrect.
Each main operation in the script is preceded by a brief explanation in form of a comment line. All the files for testing are provided inside the folder: the user is encouraged to experiment with them.

ERROR HANDLING

The program takes into account for several possible errors that may come from the user's incorrect input. A general review is given in the foregoing list:

I. the file is not existing (and/or incorrect name);
II. the provided matrix is not square:
III. the length of the vector is not equal to the order of the matrix, i.e. its number of rows.

The user is always prompted information about the error, and the program is terminated returning a specific exit code depending on it, which is provided inside the *linalg* library.

Moreover, the program transposes automatically the input matrix calling the BLAS routine *dgemm* that multiplies the input matrix with the neutral (i.e. *identity*) matrix. Therefore, there's no need for the user to take care of the ordering of the elements.
The program also returns whether the algebraic routine has been successful or not. In the latter case, depending on the exit code provided by the *dgesv_* routine itself, the program is able to inform the user either when a solution does not exists because of a non-invertible matrix (*exit code < 0*) or when a specific routine input argument is incorrect (*exit code = incorrect n-th argument*).
Lastly, several tests for the correctness of the program have been made using different strategies:

I. defining macro, such as DEBUG and PRINT for the tester to be prompted messages regarding every code stage and the solution vector eventually;
II. using assertions in the early stage of design;
III. printing error messages if I/O operations get wrong.

TIME TESTING

A separate scripts named *time_test.c* has been created to check the CPU time for different sizes. The CPU time is checked for these operations:

I. matrix allocation;
II. vector allocation;
III. *dgesv_* routine (including matrix transposition and all ancillary stuff).

Moreover, a specific script for these operations has been created where each of them is computed in number of 10.000 for each *n*. A MATLAB® script imports data and compute the mean for each *n*.
As shown in Appendix, the time complexity seems to be *O(log n)* for matrix allocation and the BLAS routine (linear slope in the double logarithmic plot), whilst the vector allocation has *constant* time complexity (it doesn't seem to change over *n*).

APPENDIX

A. double logarithmic plot for the main operations in *solve.c*