Course number: 02635 Student number: s185185
Date: 21-11-2018

Code structure

Both parts of the assignment have been structured similarly. First of all, the matrix_io.h header and source file matrix_io.c has been used as required by CodeJudge. The test case for the first part has been separated into the source file for the function call_dgesv.c and a main function for testing, call_dgesv_test.c. A makefile has been included to link the files as well as dgesv_ from LAPACK and run the program. The second part has been structured similarly, linking call_dgesv.c, solve.c as well as the matrix_io.c source and matrix_io.h header files.

Numerical considerations and error handling

Numerical assessments are mostly left to LAPACK's dgesv_ function, although the function has been thoroughly tested to ensure that the function properly handles the following numerical considerations: 1) Non-numeric input such as INFINITY and NaN values and 2) ill conditioned matrices.

The first scenario is included as an addition in the call_dgesv function, as the dgesv_ function itself did not seem to deal properly with this scenario. The exact test is elaborated further under "Test cases".

Ill conditioned matrices are handled through the dgesv_ function. Hence, if dgesv_ returns a value (through the info argument) above 0, the matrix is said to be exactly or near singular, which may result from an ill conditioned matrix.

Errors are returned as requested for NULL pointers for A and b, a non-square matrix A, incompatible dimensions between A and b and finally memory allocation errors. The remainder, except the above described part of non-numeric input, is left for the dgesv_ function but considered as described above as well as tested, which will be elaborated next.

Test cases[1]

In the first part, two test cases were performed. One with pseudo-randomly generated numbers and one with a value replaced by INF in matrix A to check the call_dgesv function's return value (screenshot of result from command line in appendix – Invalid_input_A.png).

Three test cases were performed with different inputs of matrix A for the second part. One with a well-conditioned matrix, an ill-conditioned matrix and a matrix containing numerical errors (INF and NaN values) to test the programs (attached in appendix as Case 1, 2 and 3, respectively). All matrices are attached in the appendix and the solution x for the well-conditioned matrix is included as well

The ill-conditioned matrix, correctly, returns the exactly/near singular error message from dgesv_. The matrix containing numerical errors is handled correctly in the first part, but in the second part, the read_matrix function does not appear to deal with "incorrect" input in the same way the read_vector function does, and e.g. INF and NaN values are converted to 0's (returning in an "incorrect" result in x.txt, as seen from Case 2 in the appendix). The read function could be amended to check the input and return NULL in case of invalid content in the file. An example with an INF value in vector b is also included in the appendix (attached as Case 4).

The well-conditioned matrix returns a quite accurate result too (based on x.txt), and the program therefore appears to work as expected, given a proper input.

[1] All functions, header and Makefile used for test cases are included in the appendix as .zip files under names "Part 1" and "Part 2", respectively.