DANMARKS TEKNINSKE UNIVERSITET

MATHEMATICAL SOFTWARE PROGRAMMING
Course number: 02635

Assignment 1

Student:
Mads Esben Hansen, s174434

Group member:
Karl Emil Takeuchi-Storm, s130377

November 17, 2020

"call_dgesv" was structured such that the first thing we did was checking if the input was correct. This was done by:

$$\begin{aligned} \nexists A \quad \wedge \quad \nexists b \quad &\rightarrow \quad return - 9 \\ dim(A) = m \times n, \quad m \neq n \quad &\rightarrow \quad return - 10 \\ dim(A) = m \times m,\ len(b) = n, \quad m \neq n \quad &\rightarrow \quad return - 11 \end{aligned}$$

Then two vectors are dynamically allocated, one of length n and one of length n^2. If either of the allocation fail, the function returns -12.
Then "A" is copied into the vector of length n^2. Finally the "dgesv_" function is called, both the dynamically allocated vectors are freed, and "info" (the parameter given by "dgesv_") is returned.

"solve.c" was structured such that 4 checks are done. The first one is if "argc" is not equal to 4, if so return with "EXIT_FAILURE".
Then "A" is read calling "read_matrix", if it read to NULL return with "EXIT_FAILURE".
Now "b" is read calling "read_vector", if it read to NULL return with "EXIT_FAILURE".
Finally "call_dgesv" is called, if "info" does not equal 0, return with "EXIT_FAILURE". Otherwise the function calls "write_vector" to write the output and return with "EXIT_SUCCESS".

We did not do a lot of numerical considerations in our implementation since we do not perform any computations per se. We did however test linear equations that were somewhat hard to solve. We saw that in some cases, "dgesv_" is not able to recognize that "A" is singular, and in turn returns an incorrect answer. We discussed if we should implement some extra checking to ensure this would not happen, but agreed that this was beyond the scope of this course.

First we generated linear equations of different sizes using matlab. These equations were both singular and non-singular. Then if the functions was able to compute the correct result (in some cases it was not, as described earlier). Then we tried inputting matrices and vectors are incorrect sizes, and input NULL in stead of a correct input, to check if the initial error handling was done correctly. Finally I checked for memory leaks using "valgrind", which is a tool for just that. I also made sure that every time something was allocated dynamically, it would always be freed again.