Course number: 02635
Date: 24-10-2018
Student number: s185185

Assignment 1

Code structure

Forward Substitution Function

The forward substitution function was implemented in a relatively simple manner, using the provided template. For the purpose of calculating b_1 a separate line of code was use, while the rest of the algorithm was implemented for-loops, looping through each element of R and b

Finally, numerical failure was checked by looping through each element of x, using the *isfinite()* function and returning -1 if any element returned FALSE.

Triangular Sylvester Equation

The Sylvester Equation was implemented in a similar manner to the forward substitution function, but with a few amendments.

First of all, the header file "matrix.h" was included to define the matrix_t structures. The forward substitution function was also included in the source file for the Sylvester Equation to simplify the second step of the algorithm. Furthermore, the type of the Forward Substitution function was changed from *int* to *double**, as the function would return a pointer to the Sylvester function.

The first part of the code checks for NULL pointers and incorrect dimensions. The NULL pointers were checked first to avoid runtime errors in case there are in fact NULL pointers and the dimensions according to *m* and *n* being equal as well as ensuring that they are above 0.

Similar to the substitution function, the first row was calculated first while for the remaining rows, for-loops were utilized to calculate the two steps in the algorithm. Finally, a numerical failure check was run, similar to the forward substitution function.

Numerical test[1]

A simple test was run, implementing the functions in a main program and printing the solution to the console, with the following input, α, *R*, *b* and *C* (with α, *b* being input for the forward substitution function and C being input for the Sylvester function):

$$\alpha = 2.5 \qquad R = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 7 & 8 & 9 & 10 \\ 0 & 0 & 13 & 14 & 15 \\ 0 & 0 & 0 & 19 & 20 \\ 0 & 0 & 0 & 0 & 25 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 0 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}$$

Producing the following solutions for x and X, respectively:

$$x = \begin{bmatrix} 0.2857 \\ 0.1504 \\ 0.0606 \\ 0.0305 \\ 0.0200 \end{bmatrix} \qquad X = \begin{bmatrix} 0.500 & 0.125 & 0.036 & 0.019 & 0.013 \\ 0.625 & 0.3929 & 0.1455 & 0.034 & 0.002 \\ 0.321 & 0.392 & 0.293 & 0.148 & 0.054 \\ 0.194 & 0.273 & 0.302 & 0.239 & 0.140 \\ 0.137 & 0.183 & 0.238 & 0.251 & 0.204 \end{bmatrix}$$

[1] Numerical tests in Appendices. Similar tests were run for checking numerical error and dimension/NULL errors (not included in the appendices).