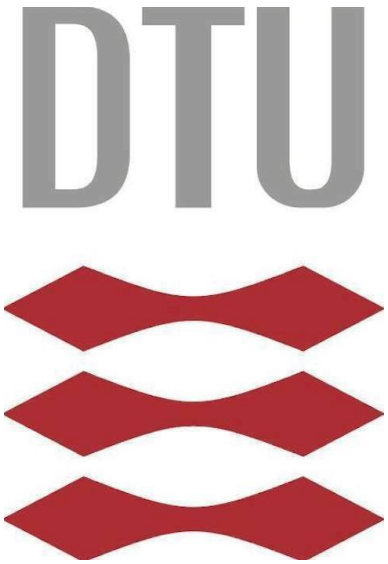
Technical University of Denmark



Lukas Leindals
s183920

02635 - Programming of Mathematical Software

02635 - Assignment 1

October, 2020
Technical University of Denmark

All code was tested using codejudge.

Part 1: Forward Substitution

To perform forward substitution the given formula is used

$$b_k \leftarrow \left(b_k - \sum_{i=1}^{k-1} b_i R_{ik} \right) / \left(\alpha + R_{kk} \right)$$

To translate this formula into a function

```
int fwdsub(unsigned long n, double alpha, double **R, double *b)
```

a for loop going from 0 to $n-1$ is initialised to get each value of k. To check for numerical errors, the divisor $\alpha + R_{kk}$ is first calculated and stored for later. If this is 0, the function returns -1, as it would result in a numerical error.
Now only the sum is needed to finish the calculation. This is done with a for loop with the variable i going from 0 to $k-1$. In the loop our indices k and i from the for loops are used to access the correct elements of b and R and add the product to the sum. Finnally, this can be used to overwrite the kth value of b with the solution.

Part 2: Triangular Sylvester equation

To implement a function

```
int tri_sylvester_solve(const matrix_t *R, matrix_t *C)
```

that can solve a system of equations on the form $R^T X + XR = C$, two steps have to be performed

1. $\mathbf{c}_k \leftarrow \mathbf{c}_k - \sum_{j=1}^{k-1} \mathbf{c}_j R_{jk}$
2. $\mathbf{c}_k \leftarrow \left(R_{kk} I + R^T \right)^{-1} \mathbf{c}_k$

First the input matrices R and C are checked for invalid inputs. If either of their respective matrices A are `NULL` -2 is returned. Furthermore, it is checked whether the input dimensions are all equal, meaning $m_R = n_R = m_C = n_C$, if this is not the case -2 is returned.
A for loop is now initialised with k as the variable, going from 0 to $m_C - 1$, as we want to update each row of C. Now a helping variable c_k of the given `struct vector` is allocated, if there is not enough memory for the pointer to the array, -2 is returned.
To fill this variable, a help function

```
double* step1(vector_t c_k, const matrix_t *R, matrix_t * C, unsigned long k)
```

is created. This function calculates step 1 by looping through the length of the input vector c_k with the variable i. First, the initial values of the kth row in C is assigned to the vector and the sum $\sum_{j=1}^{k-1} \mathbf{c}_j R_{jk}$ is then subtracted. The sum is calculated by starting a new for loop going from 0 to $k-1$ with the variable j and using the ith element of the jth row in C as $\mathbf{c}_j$ and the kth element of the jth row in R as R_{jk} and adding the product of these to the sum.
After c_k is filled using the function `step1`, the second step can be performed. Here we use the previously implemented `fwdsub` with the number of rows in C as n, R_{kk} as α, the R matrix as R and c_k as b. If this results in a numerical error -1 is returned, otherwise c_k is updated with the solution and the kth row of C can now be overwritten with c_k. Finally, the memory used by c_k is freed and the outer loop can continue the next row.