Department of Applied Mathematics
Faculty of Electrical Engineering, Mathematics and Computer Sciences
Dr. K. Smetana, L.J. Corbijn van Willenswaard

February 6, 2019

Practical assignments accompanying

Scientific Computing

2018/2019 — Practical Assignment 1

Due: March 20, 2019, after practical class

How to hand in this assignment: Put all programs and the report (pdf-file) in one folder naming the latter Name1_ Name2_ Assignment1 and send the compressed folder to `k.smetana@utwente.nl` and `l.j.corbijnvanwillenswaard@utwente.nl` after the practical class on March 20, 2019. The report file should contain the names of the two students who did the assignment together and the respective student numbers. Include email addresses (Cc:) of both students who worked on the assignment and put Assignment 1 and the names of both students in the email subject. The programs should work exactly as stated in the assignment.

Task 1 (LU decomposition)
Write a Matlab function `lu_decomp` which can be used as `[L,U] = lu_decomp(A)` for matrices $A \in \mathbb{R}^{n\times n}$ that do *not* require pivoting. The program does *not* need to check whether the matrix requires pivoting.

- The m-file should start with a comment stating what the program is doing and what are the requirements for the matrix considered.
- The algorithm should check whether the matrix is square and return an error otherwise.
- The function should return a unit lower triangular matrix L and an upper triangular matrix U such that $A = LU$.
- The Matlab function `lu` must not be used.
- Use comments in your code explaining what you are doing in the respective step.

(a) Think of a test matrix which does not require pivoting and for which you can do Gauss elimination by hand in an acceptable time frame (for instance choose a 4 by 4 matrix). Give the test matrix in the report and explain why you chose that matrix (mention for instance why it does not require pivoting).

(b) Compare the output of the algorithm in each iteration with the result you obtain in each iteration if you do Gauss elimination by hand. In the final version of the code handed in both L and U should be printed to the screen in every step. Moreover, include in the report L and U in each iteration of your calculations by hand.

Task 2 (LU decomposition with pivoting)
Write a Matlab function `lu_decomp_pivot` which can be used as
`[L,U,P] = lu_decomp_pivot(A)` for matrices $A \in \mathbb{R}^{n\times n}$.

- The m-file should start with a comment stating what the program is doing and what are the requirements for the matrix considered.
- The algorithm should check whether the matrix is square and return an error otherwise.
- The function should return a unit lower triangular matrix L, an upper triangular matrix U, and a permutation matrix P such that $PA = LU$.
- The Matlab function `lu` must not be used.
- Use comments in your code explaining what you are doing in the respective step.
- Needless to say that the code from the previous task can be reused.

(a) Think of a new test matrix to assess the new capabilities of the algorithm, i.e. a test matrix that requires pivoting, for which you can do Gauss elimination by hand. Give the test matrix in the report and explain why you chose that matrix.

(b) Compare the output of the algorithm in each iteration with the result you obtain in each iteration if you do Gauss elimination with partial pivoting by hand. Does the algorithm select the correct row during partial pivoting? In the final version of the code handed in the permutation matrix P_j should be printed to the screen in every step. Note that we do not refer here to P which would be the product of P_j, $j = 1, \ldots, n-1$. Moreover, include in the report the index of the row selected during partial pivoting in each step.

Task 3 (Conjugate gradient method)
Write a Matlab function `cg_method` for symmetric, positive definite matrices $A \in \mathbb{R}^{n\times n}$ that can be used as

```
[x,flag,relres,iter,resvec] = cg_method(A,b,tol,maxit)
```

Here,

- x is the solution of the linear systems of equations $Ax = b$
- `tol` is the prescribed tolerance
- `maxit` is the maximal number of iterations
- `relres` is the relative residual $\|Ax - b\|_2 / \|b\|_2$
- `flag` is a convergence flag which should be

`0` if `cg_method` converged to the desired tolerance `tol`, that means that `relres` $\leq$ `tol`, within `maxit` iterations

`1` if `cg_method` iterated `maxit` times but did not converge to the prescribed tolerance `tol`

- `iter` is the iteration at which `cg_method` stopped
- `resvec` is the vector whose k-th component is the relative residual $\|Ax_k - b\|_2/\|b\|_2$ at the k-th step of the iteration, while x_1 is the initial guess.

The program does *not* need to check whether the matrix is positive definite.

- The m-file should start with a comment stating what the program is doing and what are the requirements for the matrix considered.
- The algorithm should check whether the matrix is symmetric and return an error otherwise.
- The Matlab function `pcg` as well as other Matlab functions that solve linear systems of equations must not be used.
- Use comments in your code explaining what you are doing in the respective step.

(a) Develop a strategy in order to validate your code, i.e. ensure that the code does not contain any mistakes. This strategy should rely on test matrices. Explain your strategy in the report in full detail. This means that if you used for instance a certain test matrix then you should provide this test matrix in the report and justify the choice of this test matrix.

(b) Consider the matrix $A \in \mathbb{R}^{100\times 100}$ whose diagonal entries are 2 and the minor diagonals are -1. So $A(1:4, 1:4)$ should look as follows

$$\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}.$$

Morever, consider the right-hand side vector $b \in \mathbb{R}^{100}$, whose entries all equal 1. Run your code with `tol`$=10^{-8}$ and `maxit`$=100$. Include a plot of `resvec` in your report and explain the convergence behavior. Hint: You might want to have a look at the condition of the matrix.