Workout 1: System of Linear Equations

September 13, 2019

1 Mandatory exercises

Consider the system $Ax = b$ where

$$A = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 0 & 3 \\ 4 & -3 & -1 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} -3 \\ 1 \\ -6 \end{bmatrix} \quad (1)$$


2. Use $L$, $U$, and $P$ to solve the system $Ax = b$.

3. Suppose that you have represented the system of equations $Ax = b$ by storing variables $A$ and $b$.

   1. Write the Matlab/Python command that solves the system using Matlab’s `backslash` operator / Python function scipy.linalg.solve(), and stores the result in the variable $x$.
   2. Write the Matlab/Python command that $LU$-factorizes the matrix. Then, write the commands that use the result of the $LU$-factorization to solve $Ax = b$.

1.1 If you have time

4. Perform the $LU$-factorization as in Exercise 1 above. However, instead of the permutation matrix $P$, you should use a vector $\text{piv}$ to keep track of the pivoting information. Write a script that uses $L$, $U$ and $\text{piv}$ to solve $Ax = b$.

5. It is important that computational algorithms are efficient with regard to both (1) floating point operations and (2) memory requirements, and that they (3) are stable. For each of these aspects, mention some feature(s) of the algorithm used in Exercise 1 relating to that aspect.

6. True or false? (Give arguments to support your claim):

   1. You can improve the conditioning of a coefficient matrix by increasing the precision in the floating point number representation.
   2. Pivoting improves the condition number of the coefficient matrix.
   3. Pivoting improves the accuracy of the solution obtained.
2 Non-mandatory exercises

7. Suppose that you are going to compute the forces in a truss structure. This leads to a system of linear equations, where the right-hand side contains the external forces acting on the truss. We consider a case where the coefficient matrix has condition number \( \text{cond}_1(A) \approx 10^3 \) and the right-hand side is as follows (only two external forces acting on the truss):

\[
b = \begin{bmatrix}
0 \\
\vdots \\
0 \\
5000 \\
0 \\
6000 \\
0 \\
\vdots \\
0
\end{bmatrix}
\]

1. The forces are measured in Newton (N), with an accuracy of \( \pm 0.005 \text{N} \). What is the upper bound of the relative error in the right-hand side, in the 1-norm?
2. Suppose that you want the solution to be computed with a relative error no larger than 1\%, using the 1-norm. Can that be guaranteed in this case?
3. What would be the upper bound of the relative error in the solution if the condition number was ca. \( 10^6 \)?:

8. In what situations will you get a shorter execution time by using LU-factorization? Express the time gain as a formula in terms of suitable parameters.

9. The lab session for this course module contains an example concerning a diving board. In that application, the mathematical model after discretization is a system of linear equations where the coefficient matrix has a band structure. In addition, the matrix has further properties that make pivoting unnecessary. This is a very common situation in computer simulations. In such cases, the system of equations can be solved much faster if the band structure is taken advantage of. You should now investigate this in detail for the special case of a tridiagonal matrix \( A \), assuming that pivoting is not required. An efficient implementation of Gaussian elimination for this case should make use of the fact that most of the elements in the lower triangle of \( A \) are already zero. Sketch a pseudo code for how to LU-factorize \( A \) when the tri-diagonal structure is taken into account. You can follow the lecture note or look up for the Thomas algorithm.

10. Show that by using the algorithm from the previous exercise, you will only need ca. \( 3n \) floating point operations to LU-factorize the tri-diagonal matrix \( A \).

11. We now consider the general case, where \( A \) is a full \( n \times n \) matrix. Make your own, efficient implementation of LU-factorization with partial pivoting, in such a way that no explicit swapping of rows takes place. Use the integer vector \( \text{piv} \) to keep track of the ordering of the rows. Sketch your algorithm in pseudo code.