Introduction to Computational Physics SS2017

Lecturers: Rainer Spurzem & Ralf Klessen
Tutors: Sebastian Bustamante, Caroline Heneka, Ondrej Jaura, & Viviana Niro
Exercise 5 from May 17, 2017
Return by noon of May 26, 2017

1 Numerical linear algebra methods

- Consider the following matrix equation:

$$\begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}$$

  where ϵ is a small number, say, $\epsilon = 10^{-6}$.

  - Solve the above system *numerically* by hand (or write a small program that does this) using either the Gauß-Jordan method or the Gaußian elimination and backsubstitution technique (your choice), but without pivoting. Use single precision, and take $\epsilon = 10^{-6}$ (if you prefer to take double precision, then use $\epsilon = 10^{-12}$). Check the result by back-substituting (x, y) into the above equation and checking if you get the correct right-hand-side, i.e. $(1, 0.5)$.
  - Do the same, but now with row-wise pivoting. What do you notice, compared to the previous attempt? How small can you make ϵ without running into precision problems?

- Solve the above equations using the Numerical Recipes routines `ludcmp` and `lubksb` (or equivalent subroutines for LU-decomposition and back-substitution from a library of your choice), and check if the same results are obtained.

- This matrix is symmetric. But it also works for non-symmetric matrices. Try one out, and check that you use the correct order of indices for the matrix: Some programming languages use (row,column) order while others use (column,row) order, and it can also depend on the implementation of the linear algebra software!

2 Tridiagonal matrices (homework)

Consider the following tridiagonal matrix equation

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_{n-2} & b_{n-2} & c_{n-2} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_{n-2} \\ r_{n-1} \\ r_n \end{pmatrix}$$

1. Derive the iterative expressions for Gaußian elimination, in a form that can be directly implemented as a numerical subroutine. Do *not* apply pivoting here, because in the *special case* of tridiagonal matrix equations pivoting is rarely necessary in practice. (3 points)

2. Derive the iterative expressions for backward substitution, also for implementation as a numerical subroutine. (3 points)

3. Program a subroutine that, given the values $a_2 \cdots a_n$, $b_1 \cdots b_n$, $c_1 \cdots c_{n-1}$ and $r_1 \cdots r_n$, finds the solution vector given by $x_1 \cdots x_n$. (10 points)

4. Take $n = 10$, and set all a values to -1, all b values to 2, all c values to -1 and all r values to 0.1. What is the solution for the $x_1 \cdots x_n$? (2 points)

5. Put your solution $x_1 \cdots x_n$ back into the original matrix equation above and find how much the result deviates from the original right-hand-side $r_1 \cdots r_n$. Is this satisfactory? (2 points)