Introduction to Computational Physics SS2019

Lecturers: Rainer Spurzem & Ralf Klessen
Tutors: Branislav Avramov, Raul Dominguez, Robin Tress, Yun Kiyun
Exercise 4 from May 15, 2019
Return by noon of May 24, 2019

1 Numerical linear algebra methods

- Consider the following matrix equation:

$$\begin{pmatrix} \epsilon & 1/2 \\ 1/2 & 1/2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1/2 \\ 1/4 \end{pmatrix}$$

  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 $(x, y)^T$ into the above equation and checking if you get the correct right-hand-side, i.e. $(1/2, 1/4)^T$.
  - 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 any other 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. Recall that some programming languages use (row,column) order while others use (column,row) order.

2 Tridiagonal matrices (homework)

Consider the following tridiagonal $N \times N$ matrix equation

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_{N-2} & b_{N-2} & c_{N-2} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_N & b_N \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-2} \\ x_{N-1} \\ x_N \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{N-2} \\ y_{N-1} \\ y_N \end{pmatrix}$$

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 $y_1 \cdots y_N$, 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 y 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 $y_1 \cdots y_N$. Is this satisfactory? (2 points)