GEOSCI 561: Homework #3 (Newton's Method in N dimensions)

Due: Fri., Feb. 6

1. Consider a linear system of equations

\[ \begin{align*}
    A x &= b \\
    &\begin{bmatrix}
        1 & 2 & 3 \\
        3 & 1 & 2 \\
        2 & 3 & 1 \\
    \end{bmatrix} \begin{bmatrix}
        x \\
        y \\
        z \\
    \end{bmatrix} = 
    \begin{bmatrix}
        10 \\
        13 \\
        13 \\
    \end{bmatrix}
\end{align*} \]

Solve this system both analytically (i.e., by hand) and numerically, using any standard matrix solver. One possibility is the pair of routines LUDCMP and LUBKSB from the book Numerical Recipes. See call sequence on next page. I have also copied the two relevant subroutines into the class website: http://www.geosc.psu.edu/~kasting/Geosci_561/Geosci561. You should be able to download them from there. I’ve made them available in both F77 (.f) and F90 (.f90) formats, as we had trouble with the F90 files last time, for some reason. Both subroutines are in the same file, ludlub.f or ludlub.f90.

2. Consider the nonlinear system of equations

\[ \begin{align*}
    1) \quad x + y + z - 4 &= 0 \\
    2) \quad xy + z^2 - 2 &= 0 \\
    3) \quad xyz - 1 &= 0
\end{align*} \]

Find one root of this system using Newton’s method in three dimensions. Start from \((x,y,z) = (2.7, 0.4, 1.1)\) and use the matrix solver from problem 1. Compare this to the analytic solution which has the root \(x = ?, y = ?, z = 1.\) Can you find the other roots, i.e. the other values of \(z\), for this system of equations? How many iterations did it take to find the answer?

As part of this assignment, it will be useful to develop a convergence test. Suppose that we insist that \(|\Delta x_i/x_i| < 10^{-5}\) for each element of the vector \(x = (x,y,z)\). You can accomplish this by putting the following piece of code within your Newton loop:

```plaintext
errmax = 0. 
do i = 1,3 
    err = abs(dx(i)/x(i)) 
    errmax = amax1(errmax, err) 
if (errmax < 1.e-5) exit 
end do
```
LU Decomposition Method

One fast, stable way of solving a linear system of equations
\[ A \mathbf{x} = \mathbf{b} \]

is to factor the matrix A into the production of a lower triangular matrix L and an upper triangular matrix U. For a 3 x 3 matrix, this looks like

\[
A = \begin{pmatrix} x & 0 & 0 \\ x & x & 0 \\ x & x & x \end{pmatrix} \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \end{pmatrix}
\]

For reasons that become obvious if one looks into the arithmetic, it is easy to solve the original matrix equation by a method called back substitution once this factoring has been performed.

Routines that use this method or a variety of other methods are available from any standard FORTRAN library (e.g. IMSL, LINPACK, NAG). They are also from the book *Numerical Recipes* by Press et al. Two routines are required: LUDCMP factors the matrix, and LUBKSB solves the linear system using back substitution

The call sequences are:

CALL LUDCMP (A, N, NP, INDX, D)

CALL LUBKSB (A, N, NP, INDX, B)

where

A =  N x N matrix with physical dimension NP. N can be less than or equal to NP. Note that FORTRAN stores matrices by columns, i.e., it treats them as a vector with elements (col 1, col 2, ..., col N).

INDX = integer vector of length N that holds info about partial pivoting (output only). Must be dimensioned!

D = ± 1., depending on the number of row interchanges (output only)

B = vector of length N representing the right-hand side of the matrix equation (on input)

= solution vector x (on output)