

## Homework 3: LU Factorization

Due: *Thursday, September 18, 2006*

1. Obtain from the course webpage copies of the files: `arrays.hpp`, `lu1.c`, and `input` if you are programming in C; for Fortran download `lu1.f` and `input`.
2. Pivoting. Revise the routines `LUfactor1()` and `LUsolve1()` in `lu1.c` or `lu1.f` to include partial pivoting. Your routines should use the declarations

```
int LUfactor(double **A, int *P, int n);
```

and

```
int LUsolve(double **A, int *P, double *b, int n);
```

and analogous ones in Fortran, where `P[]` is a vector of integers that will store the row-pivot information.

Hints:

- (a) `LUfactor()` should begin by initializing `P[j]=j`; for  $j = 0, 1, 2, \dots, n - 1$ .
- (b) You should avoid “physically” swapping row data in the matrix for pivoting. Instead, you should always refer to the matrix entries through the pivot index, as `A[P[i]][j]`, then you can “swap rows” by just exchanging values of `P[]` entries.
- (c) Within the  $j$  loop in `LUfactor` find the row with the largest possible absolute value of `A[P[i]][j]` and use pivoting to make that the  $A_{jj}$ .
- (d) The same pivoting/permutation must be applied to vector `b[P[i]]` in `LUsolve()`. You will need to allocate memory for or declare an extra vector in order to re-arrange `b[P[i]]` into its proper order `b[i]`.
- (e) Make sure that you can solve the test problem,

$$\begin{pmatrix} 0 & 1 & 2 \\ 3 & 2 & 1 \\ 1 & 5 & 7 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 8 \\ 10 \\ 32 \end{pmatrix}$$

with solution  $x = 1, y = 2, z = 3$ .

- The matrix and right-side vector should be read from an input file formatted as follows: the first entry is a number  $n$ , the next  $n^2$  numbers are the matrix entries, ordered major row-wise, and the last  $n$  numbers are the vector entries. A sample input file `input` for the above problem can be downloaded from the web. Output the solution vector to the screen.
- Name your program `lu_lastname.c` or `lu_lastname.f`, where `lastname` is your last name. Email your program to `alayton@math.duke.edu` (just `lu_lastname.c` is sufficient, no need to include `arrays.hpp`). Make sure to also include your name in a comment box in your program file. Your program will be tested using input files with the format mentioned above, so it is important to get the I/O right. Also, make sure that your program is well documented, as your grade will take into account the readability of your program.

3. LU factorization for series of  $A_k$ . There are families of linear systems  $A_k x = b$  in which  $A_k$  changes in some simple way into a matrix  $A_{k+1}$ , and it may then be simpler to find the LU factorization of  $A_{k+1}$  by modifying that of  $A_k$ . As an example that arises in the simplex method for linear programming, let  $A_1 = [a_1, \dots, a_n]$  and  $A_2 = [a_2, \dots, a_{n+1}]$ , with all  $a_j \in \mathbb{R}^n$ . Suppose  $A_1 = L_1 U_1$  is known, with  $L_1$  lower triangular and  $U_1$  upper triangular. Find a simple way to obtain the LU factorization  $A_2 = L_2 U_2$  from that for  $A_1$ , assuming pivoting is not needed.

Hint: Using  $L_1 u_i = a_i$ ,  $1 \leq i \leq n$ , write

$$A_2 = L_1 [u_2, u_3, \dots, L_1^{-1} a_{n+1}] \equiv L_1 \tilde{U}$$

Show that  $\tilde{U}$  can be simply modified into an upper triangular form  $U_2$ , and that this corresponds to the conversion of  $L_1$  into the desired  $L_2$ . More precisely,  $U_2 = Z \tilde{U}$ ,  $L_2 = L_1 Z^{-1}$ . Find the operation cost for obtaining  $A_2 = L_2 U_2$  this way, i.e., find  $k$  such that the operation cost is  $\mathcal{O}(n^k)$ .