Assignment #9

Due Friday 21 November, at the start of class

Please read Lectures 16, 17, 20, 21, and 22 in the textbook *Numerical Linear Algebra*, SIAM Press 1997, by Trefethen and Bau. We are outright skipping Lectures 18 and 19!

DO THE FOLLOWING EXERCISES FROM THE TEXTBOOK:

- **Exercise 15.2**
- **Exercise 17.1**
- **Exercise 17.2** *Exactly what do Theorem 17.1 and Theorem 15.1 imply . . .*
- **Exercise 20.3** *Do part* (a) *only.*
- **Exercise 20.4**

DO THE FOLLOWING ADDITIONAL PROBLEMS:

P21. Implement Algorithm 20.1 in MATLAB etc. as a function with signature `[L, U] = mylu(A)`. Demonstrate that your implementation works by reproducing the stages of the calculation on pages 148–149, starting from the matrix given in equation (20.3).

P22. Consider the "two strokes of luck" in Lecture 20. Write a short MATLAB code which generates random L_k matrices and confirms the "strokes of luck" in the $m = 4$ case. Specifically, generate random matrices L_1, L_2, L_3 which are 4×4 matrices of the pattern shown in the middle of page 150. Note that the entries ℓ_{jk} for $j = k + 1, \dots, m$ are just random numbers your code generates; they do *not* come from ratios x_{jk}/x_{kk}. Then compute $L_1^{-1} L_2^{-1} L_3^{-1}$ and confirm that it comes out as in equation (20.7).

P23. An *in-place* Gauss elimination algorithm re-uses the memory in which A is stored, to store L and U. This is mentioned in the sentence after Algorithm 20.1.

(a) Write a function with signature `Z = iplu(A)` which takes as input a square $m \times m$ matrix A and computes $A = LU$ by Algorithm 20.1. It will not create separate matrices L and U. It will produce a matrix Z which has the numbers l_{jk} and u_{jk} in the corresponding locations. You will be able to recover matrices L and U as follows:

```
>> Z = iplu(A);
>> U = triu(Z),  L = tril(Z,-1) + diag(ones(m,1))
```

Demonstrate that `iplu(A)` works by applying it to the matrix A in (20.3) and recovering the factors in (20.5).

(b) Now write another function with signature `x = bslash(A,b)` which solves square systems $Ax = b$. It must call `iplu(A)` to compute the in-place LU factorization. Then it solves the system from Z *without* forming L or U.[1] It will have loops which implement forward- and backward-substitution (Alg. 17.1) using the entries of Z. Show it works by comparing to "\" on randomly-generated linear systems $Ax = b$:

```
>> x1 = bslash(A,b);
>> x2 = A \ b;
>> norm(x1 - x2) / norm(x2)
```

(c) Why is your `x = bslash(A,b)` solver not recommended for general use? Sketch how you might modify it to add partial pivoting. (*No code is needed here.*)

(Extra Credit) Figure out how to build an in-place Householder QR based solver. That is, solve $Ax = b$ for square and invertible A, based on Algorithms 10.1, 10.2, and 17.1, which is to say based on Algorithm 16.1, but using **only slightly more memory** than needed to store A and b. (*Hint: Where can you put the v vectors for the Householder reflectors, equivalently the matrix W mentioned in Exercise 10.2, as you generate zeros below the diagonal?*) One point extra credit for sketching how to do it. Two more points for a demonstrated working implementation; feel free to start from codes I have written on old homework solutions etc.

[1]And, of course, without using MATLAB's backslash operation!