Assignment #10

Due Monday, 30 November, 2015 at the start of class

Please read Lectures 20, 21, 22, 23, 24, and 25 in Trefethen & Bau.

P19. This will get you started on the "in place" idea, which problem **P20** continues. Here, no more memory is used to store L and U than is already used to store A.

(a) Write an algorithm, and implement it as a function `mylu.m` in MATLAB, which takes as input a square $m \times m$ matrix A and computes $A = LU$ by Algorithm 20.1. The code in `mylu.m` will compute "in place", that is, it will only modify entries of A, and not create separate matrices L and U. It will take A as input and produce a new matrix Z which has all numbers l_{jk} and u_{jk} in the corresponding locations. Thus, you will be able to use it this way to recover matrices L and U:

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

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

(b) Now write another function `mybslash.m` which solves square systems $Ax = b$. It calls `mylu.m` to compute the in-place LU factorization Z. Then `mybslash.m` solves the system *without forming* L or U, and *without using* MATLAB's *backslash operation*. Thus `mybslash.m` will have loops which implement forward- and backward-substitution (= Algorithm 17.1) using the entries of Z. Show it works by comparing to "\" on some nontrivial 10×10 system $Ax = b$:

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

P20. **(a)** Analogous to problem **P19**, write an in-place implementation of Householder QR, namely Algorithm 10.1: `Z = iphouse(A)`. You may assume A is square. Note, however, that Z *must* be slightly larger than A; explain why and quantify. Explain how to recover R from Z; recovering Q is more subtle.

(b) Now write another function `qrbslash.m` which calls `iphouse.m` to get Z and then implements Algorithm 16.1 in place *without forming Q or R, and without* MATLAB's *backslash operation*. Thus `qrbslash.m` will have loops which implement Algorithms 10.2 and 17.1. Show it works just as you did in problem **P19(b)**.

P21. Set up a convincing example of the conclusion of Theorem 16.3, once using MATLAB's built-in `qr` command and once using your `qrbslash` from problem **P20**. I would suggest building an invertible 5×5 matrix A with integer entries, including several distinct prime numbers. Make sure $\kappa(A)$ is at least 100, so you can tell the difference between numerical errors of sizes $\kappa(A)\epsilon_{\text{machine}}$ and $\epsilon_{\text{machine}}$. Then choose x with integer entries, and compute $b = Ax$ by multiplication; check that there are no (rounding) errors. Thus you have a linear system for which you really know an *exact* solution x. Compute $\tilde{x}$ from each of the two QR implementations and see if the bound implied by (16.7) is true. Is that bound pessimistic? Why or why not?

Exercise 20.1 in Lecture 20.

Exercise 20.3a in Lecture 20.

Exercise 20.4 in Lecture 20.