MATH 449, HOMEWORK 4

DUE OCTOBER 9, 2015

Part I. Theory

Problem 1. Suppose $A \in \mathbb{R}^{n\times n}$ is nonsingular and has a decomposition $A = LU$. Prove that the decomposition is unique. That is, if there is another decomposition $A = \widetilde{L}\widetilde{U}$, then $L = \widetilde{L}$ and $U = \widetilde{U}$.

Hint: Begin by multiplying both sides of $LU = \widetilde{L}\widetilde{U}$ by $\widetilde{L}^{-1}$ on the left and U^{-1} on the right. (Be sure to justify why these inverses exist!) Recall the properties of products and inverses of (unit) triangular matrices.

Part II. Programming

Instructions. For the programming portion of this assignment, you will be running and modifying the code in the provided file `hw4.py`. Hand in a printed copy of the modified file, as well as a printout of the IPython terminal session(s) containing the commands and output you used to get your answers.

In class, we discussed *Doolittle's method* for computing the LU decomposition of a matrix. For $k = 1, \dots, n$, we calculate the kth row of U and kth column of A as follows:

$$u_{kj} = a_{kj} - \sum_{r=1}^{k-1} \ell_{kr} u_{rj}, \qquad j = k, \dots, n$$

$$\ell_{ik} = \frac{1}{u_{kk}} \left(a_{ik} - \sum_{r=1}^{k-1} \ell_{ir} u_{rk} \right), \qquad i = k+1, \dots, n.$$

In addition to its memory/cache efficiency, this loop is also easy to implement using Python's slicing notation:

```
for k in range(n):
    U[k,k:] = A[k,k:] - dot(L[k,:k], U[:k,k:])
    L[k+1:,k] = (A[k+1:,k] - dot(L[k+1:,:k], U[:k,k]))/U[k,k]
```

(This uses the slicing expression `k:` for $j = k, \dots, n$, `k+1:` for $i = k+1, \dots, n$, `:k` for $r = 1, \dots, k-1$, and `dot` to compute the sum as a dot product.)

This is implemented as the function `lu` in `hw4.py`. Note that the function returns *two* arrays, L and U. (In Python terminology, the data structure containing the ordered pair L, U is called a *tuple*. More generally, a tuple can be used to represent any ordered n-tuple, hence the name.) The two return values can be obtained by calling the function like `L,U = lu(A)`.

Problem 2. Consider the LU decomposition of $A = \begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix}$. As discussed in class, for ϵ sufficiently small, floating point error will cause the product LU to be computed as $\begin{pmatrix} \epsilon & 1 \\ 1 & 0 \end{pmatrix} \neq A$.

a. What is the smallest $k \in \mathbb{N}$ where this problem occurs for $\epsilon = 10^{-k}$?

b. Verify that this does not occur for the pivoted matrix $PA = \begin{pmatrix} 1 & 1 \\ \epsilon & 1 \end{pmatrix}$, using the value of ϵ obtained in part a.

Problem 3. In `hw4.py`, the function `uSolve` solves the upper triangular system $Ux = b$ using back substitution. (The only tricky part here is iterating backwards, which is done using the `reversed` function. As with the function `lu`, notice the use of `dot` to compute the sum as a dot product.)

Similarly the function `lSolve` is meant to solve the lower triangular system $Lx = b$ using forward substitution—but it is only partly implemented. Finish the implementation by replacing the dummy line

```
x[i] = 0 # replace this line
```

in the main loop with the appropriate expression for `x[i]`. (Do not assume that L is *unit* lower triangular.) To test your implementation, run the following commands and print your output:

```
seed(449)
L = tril(rand(3,3))
b = rand(3)
lSolve(L,b)
```

(This takes L and b to be "random." However, the `seed` command begins by seeding the random number generator with the number 449, so everyone should get the same answer.)

Problem 4. Create a function `luSolve(L,U,b)` which solves the system $LUx = b$ using forward and back substitution. To test your implementation, run the following commands and print your output:

```
seed(449)
A = rand(3,3)
b = rand(3)
L,U = lu(A)
luSolve(L,U,b)
```