CS 4220: Assignment 2

Due: Friday, February 19, 2010 (In Class or in Upson 5153 by 4pm)

Scoring for each problem is on a 0-to-5 scale (5 = complete success, 4 = overlooked a small detail, 3 = good start, 2 = right idea, 1 = germ of the right idea, 0 = missed the point of the problem.) One point will be deducted for insufficiently commented code. Unless otherwise stated, you are expected to utilize fully MATLAB'S vectorizing capability subject to the constraint of being flop-efficient. Test drivers and related material are posted on the course website `http://www.cs.cornell.edu/courses/cs4220/2010sp/`. For each problem submit output and a listing of all scripts/functions that *you* had to write in order to produce the output. You are allowed to discuss *background* issues with other students, but the codes you submit must be your own.

P1. (A Triangular Matrix Equation)

Suppose $S \in \mathbb{R}^{m\times m}$ and $T \in \mathbb{R}^{n\times n}$ are upper triangular and $B \in \mathbb{R}^{m\times n}$. The goal is to compute $X \in \mathbb{R}^{m\times n}$ so that $SXT + X = B$. Comparing k-th columns we see that

$$S\left(\sum_{j=1}^{k} T(j,k)X(:,j)\right) \;+\; X(:,k) \;=\; B(:,k)$$

Manipulate this equation to derive a formula for $X(:,k)$ assuming that $X(:,1{:}k-1)$ is known. By using this recipe, implement the following function so that it performs as specified:

```
  function X = MatTriSol(S,T,B)
% S (m-by-m) and T (n-by-n) are upper triangular and B is an m-by-n.
% X is an m-by-n matrix such that SXT +  X = B.
```

Your implementation should be vectorized and flop-efficient. It is OK to use \ to solve triangular systems.

Submit a listing of your implementation. It should include a comment that characterizes when the system has a solution. (Hint. Consider the diagonal entries of S and T.) Test your implementation with the script `P1` and submit output.

P2. (Effect of Condition on Relative Error)

Here is the 5-by-5 Pascal matrix:

$$P_5 \;=\; \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 1 & 3 & 3 & 1 & 0 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 & 6 \\ 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \;=\; \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 6 & 10 & 15 \\ 1 & 4 & 10 & 20 & 35 \\ 1 & 5 & 15 & 35 & 70 \end{bmatrix}$$

The built-in function `pascal(n)` is handy for generating these matrices. Here are two facts about P_n:

Fact 1. The (n,n) entry of P_n is given by the binomial coefficient

$$\begin{pmatrix} 2(n-1) \\ n-1 \end{pmatrix}$$

which is a very large number even for modestly-sized n.

Fact 2. If one is subtracted from the (n,n) entry of P_n, then the resulting matrix is exactly singular.

Recall that if A is an ill-conditioned matrix, then there is a matrix E with the property that $A+E$ is singular and $\| E \|/\| A \|$ is small. It follows from Facts 1 and 2 that the Pascal matrices are very ill-conditioned for large n.

For a given n, define $b^{(n)} = P_n * \text{ones(n,1)}$. Write a script `P2` that for $n = 4{:}4{:}16$ produces a computed solution $\hat{x}$ to $P_n x = b^{(n)}$ obtained by making explicit use of the factorization `[L,U,P] = lu(Pascal(n))`. For each n, your script should display (a) the value of n, (b) the 1-norm condition of P_n obtained via `cond`, (c) the relative 1-norm error in $\hat{x}$, (d) the relative error heuristic `eps*cond(A,1)`, and (e) $\hat{x}$. Submit `P2` and the output it produces. Display $\hat{x}$ through the 16th decimal place. Include a brief comment in `P2` that explains why Fact 2 is true.

P3. (Hessenburg LU)

We say that $H \in \mathbb{R}^{n\times n}$ is *upper Hessenberg* if $a_{ij} = 0$ whenever $i > j+1$. If we apply Gaussian elimination with partial pivoting to such a matrix then permutation matrices $P_1, \ldots, P_{n-1}$ and multiplier matrices $M_1, \ldots, M_{n-1}$ are generated so that

$$M_{n-1}P_{n-1}\cdots M_2P_2M_1P_1A = U$$

is upper triangular. Note that P_k is either I_n exactly or I_n with rows k and $k+1$ interchanged. The multiplier matrices are particularly simple in that there is only one multiplier below the diagonal, e.g.,

$$M_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -\alpha_2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad n = 4,\ k = 2.$$

Write a MATLAB function `[alpha,piv,U] = HessLU(A)` that takes an upper Hessenberg matrix A and returns an encoding of the multiplier matrices in row vector `alpha(1:n-1)` and an encoding of the permutation matrices in row vector `piv(1:n-1)`. In particular, α_k should be housed in `alpha(k)` and `piv(k)` should be zero if $P_k = I_n$ and one if not. In addition, write a function `x = HessSolve(alpha,piv,U,b)` takes the output of `HessLU` and uses it to solve the Hessenberg system $Ax = b$. It may be assumed that A is nonsingular.

Submit a listing of `HessLU` and `HessSolve` together with the output that is produced when the script `P3` is run. Do not forget to specify clearly `HessLU` and `HessSolve`.

P4. (Computing the Stationary Vector of a Markov Process)

Suppose a large number of fleas are placed on a runway with n indexed tiles, e.g.,

1	2	3	4	5	6	7	8	9

$n = 9.$

Assume q_{ij} is the probability that a flea on tile j hops to tile i during a given time step and let $f^{(k)} \in \mathbb{R}^n$ have the property that $f_i^{(k)}$ is the number of fleas that reside on tile i after the k-th time step. Note that with this model,

$$f^{(k+1)} = Qf^{(k)}$$

where $Q = (q_{ij})$ is the n-by-n matrix of transition probabilities. The matrix Q has unit column sums because a flea on tile j must hop somewhere with probability 1.

Now suppose that $0 < \alpha < 1$ and define the transition probabilities by

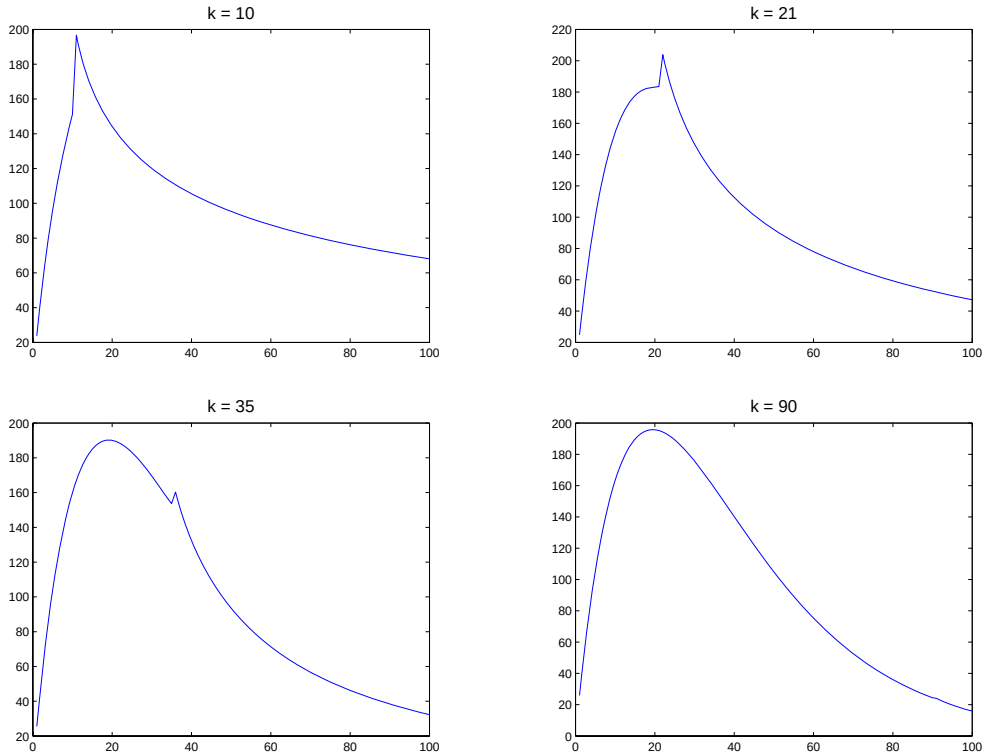
$$q_{ij} = \begin{cases} \alpha & \text{if } i = j+1 \\ (1-\alpha)/j & \text{if } i \le j < n \\ 1/n & \text{if } j = n \\ 0 & \text{Otherwise} \end{cases} \tag{1}$$

For example, if $n = 4$ and $\beta = 1-\alpha$ then

$$Q = \begin{bmatrix} \beta & \beta/2 & \beta/3 & 1/4 \\ \alpha & \beta/2 & \beta/3 & 1/4 \\ 0 & \alpha & \beta/3 & 1/4 \\ 0 & 0 & \alpha & 1/4 \end{bmatrix}$$

This means that a flea on tile j hops one tile to the right with probability α. Otherwise it hops with equal probability to one of tiles 1 through j.

Does the distribution of fleas reach a steady state after enough time steps? Here are some snapshots of $f^{(k)}$ for the case $n = 100$, $f^{(0)} = \text{ones(n,1)}$, and $\alpha = .95$:



These plots were more or less produced by the script

```
fk = 100*ones(100,1);   %10000 fleas uniformly distributed.
for k=1:100
   fk = Q*fk;
   plot(fk)
   pause
end
```

It turns that the $f^{(k)}$ converge to a vector f. Note that if $Qf = f$ then $(Q - I_n)f = 0$. In other words, the limit vector f (called the *stationary vector*) is in the nullspace of $Q - I_n$. We show that the vector f can be computed via the factorization $P(Q - I) = LU$. Since $Q - I$ is singular, then the same can be said about U. (L is nonsingular since it has ones on its diagonal.) Thus, $(P - I)f = 0$ iff $Uf = 0$. A singular upper triangular U must have at least one zero on its diagonal. For our problem, it is possible to show that $u_{11} \cdots u_{n-1,n-1} \neq 0$ and $u_{nn} = 0$. Using this fact, show how to compute a nonzero z so that $Uz = 0$ and $z_n = 1$. The final f can be obtained by scaling z.

Write a function `P4(alpha,n)` that produces a plot of the stationary vector for the n-by-n matrix Q defined by (1). The scaling of the stationary vector doesn't matter. (The plot of f and μf will look the same.) Submit a listing of `P4` and the plots for `P4(.9,100)`, `P4(.95,100)`, and `P4(.98,100)`. You may use the MATLAB `lu` function or your `HessLU` to produce U.