CS 322: Assignment A4

Due: Tuesday, April 6, 2004, 4PM

Do not submit work unless you have adhered to the principles of academic integrity as descibed on the course website:

http://www.cs.cornell.edu/Courses/cs322/2004sp/

Points will be deducted for poorly commented code, redundant computation that seriously effects efficiency, and failure to use features of MATLAB that are part of the course syllabus. In particular, use vector operations whenever possible. Pay attention to the course website for news that relates to this assignment.

Problem A (6 pts) Generalized Sylvester Equations

Suppose $R, T \in \mathbb{R}^{n\times n}$ and $S, Q \in \mathbb{R}^{m\times m}$ are given upper triangular matrices. Given $B \in \mathbb{R}^{m\times n}$ we wish to compute $X \in \mathbb{R}^{m\times n}$ so that

$$SXR - QXT = B$$

Analogous to the discussion in P6.1.3, develop a recipe for $X(:,k)$ assuming that $X(:,1),\ldots,X(:,k-1)$ is known. Using the algorithm that you discover, implement the following function so that it performs as specified

```
    function X = GenSylvester(S,R,Q,T,B)
%   S and Q are  m-by-m upper triangular matrices
%   R and T are  n-by-n upper triangular matrices
%   B is an m-by-n matrix
%   X solves the linear system SXR - QXT = B.
%   It is assumed that this system is nonsingular.
```

Submit your implementation of `GenSylvester`. You may use the \ operator to solve triangular systems. We'll test it with the script `A4A` which is available on the website.

Problem B (7 pts) A Markov Model for Plasmid Replication

Let n be a positive integer. Extrachromosomal DNA elements called *plasmids* are found in many types of bacteria. Assume that in a particular species there is a plasmid P and that exactly n copies of it appear in every cell. Sometimes the plasmid appears in two slightly different forms. These may differ at just a few points in their DNA. For example, one type might have a gene that codes for resistance of the cell to the antibiotic ampicillin, and the other could have a gene that codes for resistance to tetracycline. Let's call the two variations of the plasmid A and B.

Assume that the cells in the population reproduce in unison. Here is what happens at that time. The cell first replicates its DNA matter. Thus, if a cell has one type A plasmid and three type B plasmids, then it now has two type A plasmids and six type B plasmids. After replication, the cell divides. The two daughter cells will each receive four of the eight plasmids. There are several possibilities and to describe them we adopt a handy notation. We say that a cell is (i_A, i_B) if it has i_A type A plasmids and i_B type B plasmids. (Note that $i_A + i_B = n$.) So if the parent is $(1, 3)$, then its daughter will be either $(0, 4)$, $(1, 3)$, or $(2, 2)$.

The probability that a daughter cell is (i'_A, i'_B) given its parent is (i_A, i_B) requires the use of *binomial coefficients* [1]: and is given by

$$P_{i'_A,i_A} = \frac{\begin{pmatrix} 2i_A \\ i'_A \end{pmatrix}\begin{pmatrix} 2i_B \\ i'_B \end{pmatrix}}{\begin{pmatrix} 2n \\ n \end{pmatrix}}.$$

In the numerator you see the number of ways we can partition the parent's replicated DNA so that the daughter is (i'_A, i'_B). The denominator is the total number of ways we can select n plasmids from the replicated set of $2n$ plasmids.

[1]Recall that $\begin{pmatrix} q \\ r \end{pmatrix} = \frac{q!}{r!(q-r)!}$. The value defaults to one if $q = 0$ or $r = 0$ or $q < r$.

Let $P(n)$ be the $(n+1)$-by-$(n+1)$ matrix whose (i'_A, i_A) entry is given by $P_{i'_A,i_A}$. (Note subscripting from zero.) If $n = 4$, then

$$P = \frac{1}{\binom{8}{4}} \begin{bmatrix} \binom{0}{0}\binom{8}{4} & \binom{2}{0}\binom{6}{4} & \binom{4}{0}\binom{4}{4} & \binom{6}{0}\binom{2}{4} & \binom{8}{0}\binom{0}{4} \\ \binom{0}{1}\binom{8}{3} & \binom{2}{1}\binom{6}{3} & \binom{4}{1}\binom{4}{3} & \binom{6}{1}\binom{2}{3} & \binom{8}{1}\binom{0}{3} \\ \binom{0}{2}\binom{8}{2} & \binom{2}{2}\binom{6}{2} & \binom{4}{2}\binom{4}{2} & \binom{6}{2}\binom{2}{2} & \binom{8}{2}\binom{0}{2} \\ \binom{0}{3}\binom{8}{1} & \binom{2}{3}\binom{6}{1} & \binom{4}{3}\binom{4}{1} & \binom{6}{3}\binom{2}{1} & \binom{8}{3}\binom{0}{1} \\ \binom{0}{4}\binom{8}{0} & \binom{2}{4}\binom{6}{0} & \binom{4}{4}\binom{4}{0} & \binom{6}{4}\binom{2}{0} & \binom{8}{4}\binom{0}{0} \end{bmatrix}.$$

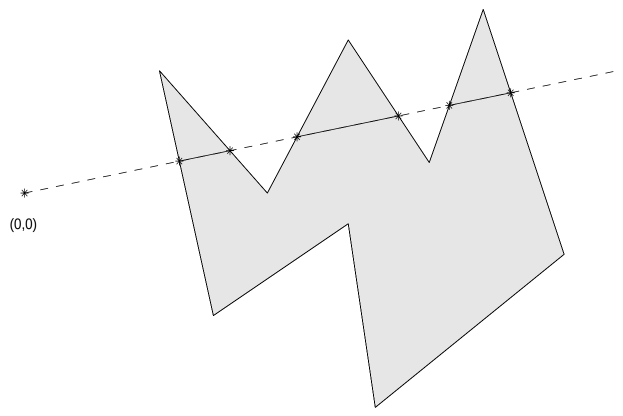
Call this the plasmid transition matrix. Suppose we have a large population of cells and that $v \in \mathbb{R}^{n+1}$ has the property that v_i is the fraction of cells that have i type A plasmids, $i = 0{:}n$. We'll call such a vector the "plasmid make-up" vector. Suppose each cell in the population divides producing two daughter cells. If $P = P(n)$ and $z = Pv$ then z_i will be the fraction of daughter cells that have i type A plasmids. More generally, if $z = P^k v$, then u specifies the plasmid make-up in the k-th generation. (I.e., if $k = 3$ we're looking at the set of great-granddaughters.) Going in the other direction, suppose z satisfies the linear system, $P^k z = v$. It follows that z is the plasmid make-up vector for a cell population that would produce the given population after k divisions. Thus, given the plasmid make-up of the current population we can see how it evolves via matrix-vector multiplication and we can see where it came from via linear equation solving. Complete the following MATLAB function so that it performs as specified:

```
    function Z = Evolve(v,k)
  % v is a column vector and k is a nonzero integer.
  % Define n = length(v)-1 and let P be the (n+1)-by-(n+1) plasmid transition matrix.
  % Z is a (n+1)-by-|k| matrix
  % If k is positive then Z(:,i) = (P^i)v, i = 1:k.
  % If k is negative, then (P^i)Z(:,i) = v, i = 1:|k|.
```

Exploit structure when you set up P. It should only involve $O(n^2)$ flops. Make effective use of the MATLAB function `[L,U,P] = lu(P)`. Submit your implementation of `Evolve`. We'll test it with the script `A4B` which is available on the website.

Problem C (7 pts) A Ray Through a Polygon

Here is a ray "going through" a polygon:



We would like to compute the length of that part of the ray that is "inside" the polygon, i.e., the sum of the lengths of the solid line segments. To do this we need to compute the intersection points of the ray with the polygon edges and sum the appropriate point-to-point distances. It turns out that this involves solving a bunch of 2-by-2 linear systems, one for each edge of the polygon. The central part of the problem is to vectorize this process so that all these systems are solved "at the same time".

We start with a review of parametric equations for rays and line segments. A ray that "leaves" the origin making angle θ with the positive x-axis can be specified as follows:

$$\{\ (x(t), y(t)) \mid x(t) = \cos(\theta)t,\ y(t) = \sin(\theta)t,\ 0 \le t\ \}\ .$$

Likewise, a line segment that connects the points (α_1, β_1) and (α_2, β_2) can be specified as follows:

$$\{\ (x(t), y(t)) \mid x(t) = \alpha_1 + (\alpha_2 - \alpha_1)t,\ y(t) = \beta_1 + (\beta_2 - \beta_1)t,\ 0 \le t \le 1\ \}\ .$$

If we can find t_1 and t_2 that satisfy $0 \le t_1$ and $0 \le t_2 \le 1$ so that

$$\begin{aligned} \cos(\theta)t_1 &= \alpha_1 + (\alpha_2 - \alpha_1)t_2 \\ \sin(\theta)t_1 &= \beta_1 + (\beta_2 - \beta_1)t_2 \end{aligned}$$

then the ray and the line segment intersect. To that end we simply solve the 2-by-2 linear system

$$\begin{bmatrix} \cos(\theta) & (\alpha_1 - \alpha_2) \\ \sin(\theta) & (\beta_1 - \beta_2) \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix}$$

and check to see if $0 \le t_1$ and $0 \le t_2 \le 1$. If these conditions hold, then the ray and the line segment intersect and the point of intersection is $(\cos(\theta)t_1, \sin(\theta)t_1)$.

Returning to the ray-through polygon problem, suppose $(x_1, y_1), \ldots, (x_n, y_n)$ are the polygon's vertices and that the ray makes an angle of θ radians with the positive x-axis. Let $d(\theta)$ be the length of that part of the ray that is inside the polygon. Of course, if the ray fails to intersect the polygon then $d(\theta) = 0$. Otherwise, we need to determine the points where the ray intersects the polygon's edges. Suppose $(u_1, v_1), \ldots, (u_m, v_m)$ are these points of intersection, indexed in order of increasing distance from the origin. It follows that

$$d(\theta) = \sum_{i=1}^{m/2} \sqrt{(u_{2i-1} - u_{2i})^2 + (v_{2i-1} - v_{2i})^2}$$

Note that m, the number of intersection points, must be even because each time the ray "enters" the polygon it must "leave" the polygon.

Write a fully vectorized MATLAB function `[d,u,v] = InsideDist(x,y,theta)` that takes column n-vectors `x` and `y` that define the vertices of the given polygon and returns in `d` the value of $d(\theta)$ where θ is the value of the scalar `theta`. Here is what `InsideDist` can assume about the polygon:

- $n \geq 3$.
- The origin is not inside the polygon.
- Every ray from the origin intersects no more than one polygon vertex. This guarantees that all the 2-by-2 systems are nonsingular.
- The polygons edges only meet at the vertices, i.e., no crossing edges.

If the ray and the polygon intersect, then output parameters `u` and `v` should be column vectors that return the coordinates of the intersection points, indexed in order of increasing distance from the origin. If the ray and the polygon fail to intersect, then `u` and `v` should be empty vectors.

For each polygon edge a 2-by-2 linear system must be solved to see if it intersects with the ray. These systems must be solved using the method of Gaussian elimination with partial pivoting. No loops are necessary for this part of the calculation as this process can be vectorized completely.

Run the script `A4C` to test your implementation of `InsideDist`. This script is available on the website.

We mention that many image processing problems involve the solution of lots of small linear systems "at the same time". Vectorization in the style of this problem is often essential for high performance. It must also be said that a "production version" of `InsideDist` would have to deal gracefully with ill-conditioning in the 2-by-2 systems.