MCS 320 Project Three : Markov Chains with MATLAB due Wednesday 30 April 2003, at 2PM.

Most matrices which occur in practical applications have a special structure. For example, a matrix which consists of nonnegative numbers and where the sum of the entries in each column sums up to one is called *a stochastic matrix.* Stochastic matrices occur in situations where the outcome of each trial in an experiment falls into several specified outcomes (each possible with a certain probability), and where the outcome of one trial depends only on the immediately preceding trial. In such situation, the next outcome vector is obtained by multiplying the previous vector with a stochastic matrix. The sequence of these outcome vectors is called a *Markov chain.*

0. Modeling the Outcomes of Elections

Suppose studies have shown that of those who voted in a previous election, 70% will again vote democratic, 20% will vote republican, and the remaining 10% will turn libertarian. Similarly, of those who voted republican, 10% will vote democratic, 80% will stay republican, while the remaining 10% turns libertarian. Finally, 30% of the libertarian voting public will turn democratic in the next election, 30% will vote republican, while the remaining 40% will vote again libertarian. The numbers in these three sentences are the columns in the stochastic matrix A:

$$A = \begin{bmatrix} 0.70 & 0.10 & 0.30 \\ 0.20 & 0.80 & 0.30 \\ 0.10 & 0.10 & 0.40 \end{bmatrix} \quad \text{and with} \quad x = \begin{bmatrix} \%\text{voting democratic} \\ \%\text{voting republican} \\ \%\text{voting libertarian} \end{bmatrix}$$

we see that the mathematical translation of the first three sentences in this section is the matrix-vector product Ax.

A similar application concerns a car rental agency with three different locations. Customers can rent and return a car at any location. Given observed typical traffic patterns, we are interested in the distribution of cars over the three locations.

The stochastic matrix A defines a Markov chain:

$$x^{(k)} = Ax^{(k-1)}, \quad \text{or} \quad x^{(k)} = A^k x^{(0)}, \quad k = 1, 2, \ldots$$

Suppose we initially start with voting percentages of 80%, 10%, and 10%. Then with the sequence of commands

```
>> A = [.7  .1  .3;  .2  .8  .3;  .1  .1  .4 ]                        enter
>> x0 = [.8;  .1;  .1]                      % start vector            enter
>> A^10 * x0                                % after 10 elections      enter
```

we can model the outcome after any number of elections.

1. Assignment One: Computing Trajectories

Our first question is to see how the outcomes evolve in time. With the A and $x0$, we can make a trajectory T of 10 outcomes typing

```
>> T = [x0]               % initialization     enter
>> for i = 1 : 10         % loop of 10 steps   enter
     T = [T  A * T(:, i)] % extend T           enter
end                                            enter
```

At the end we see T as a 3-by-11 matrix, with in its columns the coordinates of the points along the trajectory.

The goal of assignment one is to put the above commands in an m-file, called `trajectory.m`. The prototype should be

```
function X = trajectory(A,x0,n)
%
%  Returns in the columns of the matrix X the coordinates of
%  n+1 points along the trajectory defined by X(k+1) = A*X(k),
%  starting at X(1) = x0.
%
```

so that the trajectory T from above can be simply created by one command:

```
>> trajectory(A, x0, 10)     enter
```

2. Assignment Two: Plotting Several Trajectories

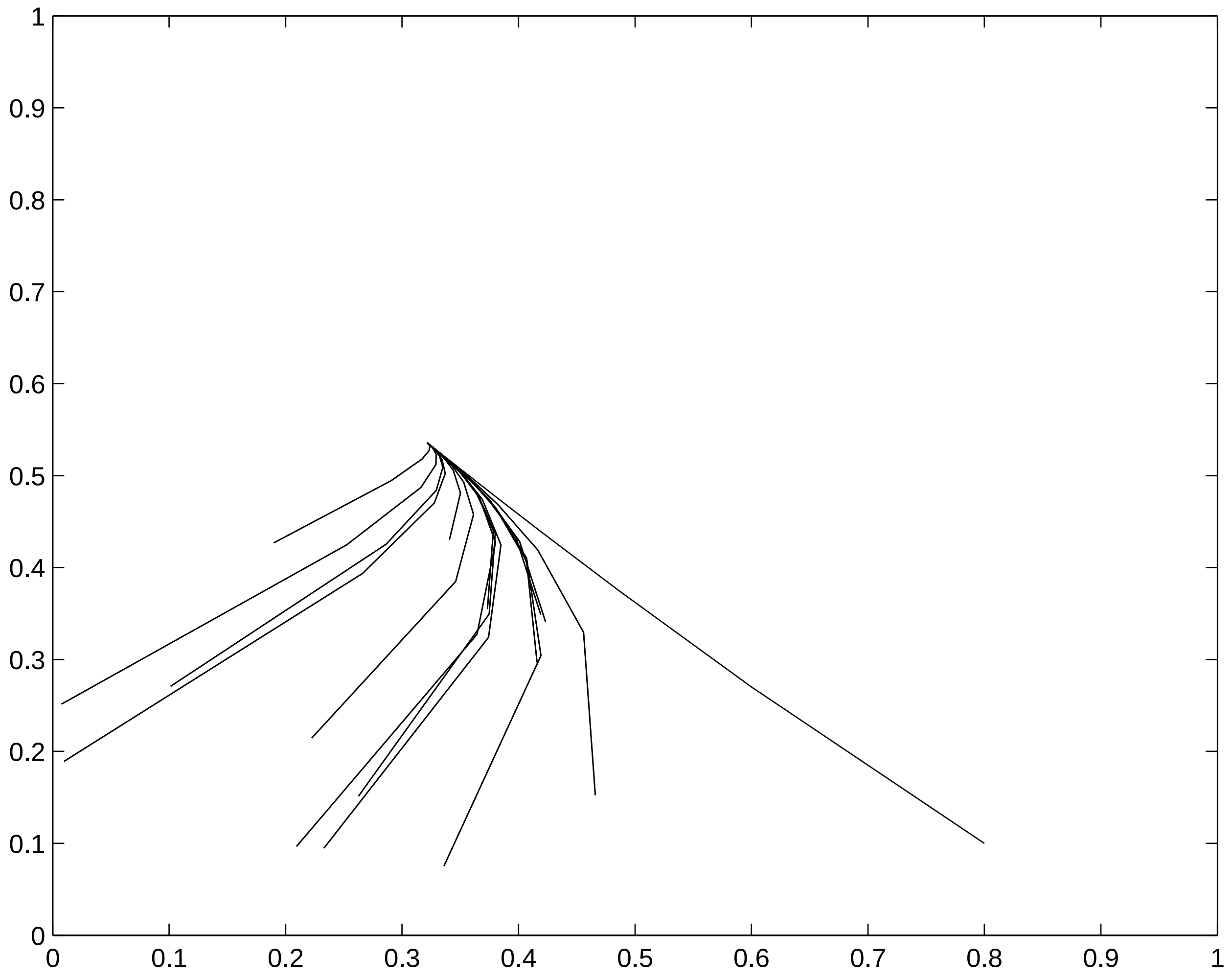
Now we can generate trajectories easily, we can visualize them. Since the sum of the coordinates of every point equals one, all our points lie in the same plane $x+y+z=1$ we consider only two dimensional plots. Suppose for the trajectory T, we wish to see the relation between the first with the second coordinate (democratic versus republican votes), then we execute the commands

```
>> axis([0, 1, 0, 1])       % fix coordinate system    enter
>> plot(T(1, :), T(2, :))   % select 1st and 2nd row   enter
```

Now we want to plot many trajectories, for various start vectors. First we generate 15 random start points:

```
>> X0 = [rand(1, 15)/2; rand(1, 15)/2; ones(1, 15)]   % rand draws in [0, 1]   enter
>> X0(3, :) = X0(3, :) − X0(1, :) − X0(2, :)          % sum must be one        enter
```

Using $X0$, we repeatedly call *trajectory* to generate trajectories for plotting:



The goal of assignment two is to put the above command in an m-file, called `trajplots.m`, with prototype:

```
function trajplots(A,x0,n,i1,i2)
%
%  Plots several trajectories A^k*x0, k=1:n, i1 versus i2.
%
%  On entry:
%    A        stochastic matrix defining the Markov chain
%    x0       matrix with in its columns various start points
%    n        number of stages in each trajectory
%    i1       first horizontal coordinate of each trajectory
%    i2       second vertical coordinate of each trajectory
%
```

To make a figure like the one above, we then simply type

$$>> trajplots(A, X0, 40, 1, 2) \quad \textbf{enter}$$

This command shows the first two coordinates of all trajectories starting at the columns in $X0$, using 40 points on each trajectory.

3. Assignment Three: Calculating Steady-State Vectors

From the plots we have seen that all trajectories converge to the same point, this is the so-called *steady-state vector*. So independently of the starting points $x0$, every trajectory will converge to the steady-state vector. So this steady-state vector only depends on the transition matrix A.

In assignment three we will investigate how to calculate the steady-state vector, given the matrix A. Basically, we solve the linear system $Ax = x$ and there are two methods to do this:

from the output of $[v, d] = eig(A)$: The steady-state vector is parallel to the eigenvector which corresponds to the eigenvalue one. This eigenvector can be found in the columns of the output **v**.

from the output of $null(A - eye(3))$: We can directly compute the eigenvector with eigenvalue one by computing the null space of the matrix $A - eye(3)$ ($eye(3)$ is the MATLAB notation for a 3-by-3 identity matrix).

In both methods we need to multiply the vector we find with a scalar to have the sum of the components in the vector to equal one.

4. The deadline is Wednesday 30 April 2003

Bring to class the printout of your MATLAB session (also GNU Octave works fine to solve this project) which illustrate the good working of your m files. With the command **type** you can bring the listing of an m-file in your session, i.e.: execute as *type trajectory.m* and *type trajplots.m* to show your m-files.

It is good practice to use one m-file which generates all answers to the three assignments.

For assignment two, you also must provide the printout of the plots, for coordinates 1-2, 1-3, and 2-3.

If you have questions, comments, or difficulties, feel free to come to my office for help.

Mark your calendars for the following event:

FINAL EXAM is in Taft Hall 0219

on Tuesday 6 May 2003 from 1:00 till 3:00PM.